AdePT

Accelerated demonstrator of electromagnetic Particle Transport

25th Geant4 Collaboration Meeting

andrei.gheata@cern.ch

Background & motivation

- HEP simulation on accelerators: a hot topic
  - Several discussions, presentations, brainstorming meetings over the last 6 months
- Part of Geant4 simulation could be handed over to the attached GPU card
  - EM calorimeter simulation seems a natural candidate
  - If we can profit from massive parallelism at track level - EM showers
  - Expressing code as more simple/compact kernels, driven by a GPU-specific workflow
- Some prototypes developed to demonstrate specific functionality - not enough
  - Geometry on GPU (VecGeom based ray-tracing)
  - ‘Stateless’ Geant4 to allow easy extraction of tracks from the standard stepping workflow
- Consolidating the effort in a more coherent way in the R&D team
  - Work on a prototype demonstrating a realistic simulation workflow on GPU
  - Toy models as kernels first, evolving to realistic ones later-on

Prototype goals

Demonstrate a realistic complete simulation workflow on GPU

- Understand technical possibilities/limitations for GPU usage in full HEP simulation
- Assessment of feasibility, development effort and performance expectations of a large scale GPU simulation project targeting EM shower simulation confined to GPU
- 6 month - 1 yr time scale

Prototype

- Start with a basic “Fisher-Price” like workflow demonstrator
  - Single particle type carrying minimal state, 2 processes (energy loss and secondary generation), no scoring (or minimal energy deposits per cell)
  - Allowing to develop a framework controlling a dynamic track workflow
- Evolve as e+/e-/gamma simulator in simple calorimeter setup
  - Magnetic field
  - VecGeom-based transport manager as first implementation
  - Gradually evolved physics processes allowing to simulate EM showers
  - Simple pre-configured scoring as simulation result, transferred to host
- Maintain CPU compatibility for the entire simulation
  - Adding possibility to connect to a Geant4 simulation on the host CPU

Technical objectives

1. Development of a core GPU transport engine
    - Dynamic kernel scheduling, management of workflow and state data
2. Adapt/develop/optimize GPU-friendly transport components
    - Mockup -> realistic versions for physics models providing similar shower development as for CPU version
    - Realistic geometry optimized for GPU
3. Understand constraints/hard limits and find solutions for:
    - Data handling, memory management
    - Kernel scheduling, GPU performance
    - Single precision usage

Work areas: core

- Workflow management
  - Workflow design. Kernel scheduling and processing flow.
- Data management
  - Data model design. Handling of state data. Memory management.
- Infrastructure & core services
  - Programing model, math, random numbers, types, concurrency and synchronization.
- Reuse/inspire as much as possible from existing implementations and HEP experience
  - VecCore/VecGeom, ALICE GPU reconstruction, LHCb Allen framework, …
  - Performance portability libraries

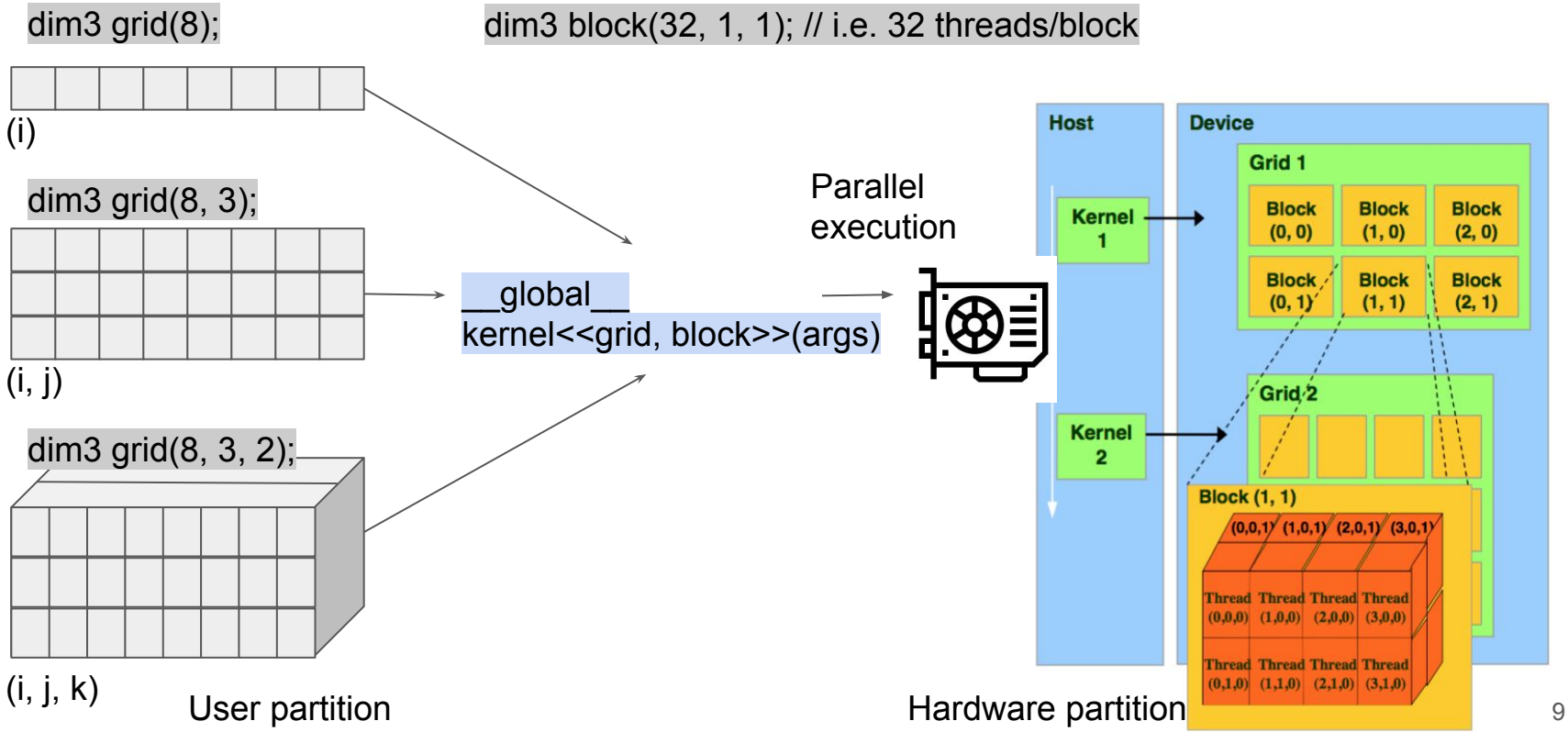
Work areas: components re-design

- Geometry
  - optimized GPU navigator
  - Investigation of alternatives: flattening and tessellation, Optix
- Field
  - Helix in a first phase -> R-K
- Physics models implementation and porting
  - Mock-up versions allowing to emulate shower-like behavior (short time scale)
  - A more compact version (GPU-friendly) of the EM physics implemented in GeantV
- Single precision handling for all the above

AdePT repo and contributions

- ‘Ideas and plans’ document
- Very recently picked a ‘code name’ and created a repository in GitHub
  - https://github.com/apt-sim/AdePT
- Bootstrap repo with basic household tools: facilitate collaboration and contributions
  - Apache 2.0 licence for code, Creative Commons CC-BY-4.0 for docs + checker action
  - © CERN copyright - should be OK for all HEP contributors
  - Contributors guide, Code of Conduct
- Started regular morning meetings in the RnD group
  - Coordination, general design for now
  - Mattermost channel, you are very welcome to join!
  - We expect more topical development meetings to come when the prototype gets more mature

Design sketching: static kernel scheduling not suitable



Simulation kernels & compute pipeline

Update kernels (U): modify track state

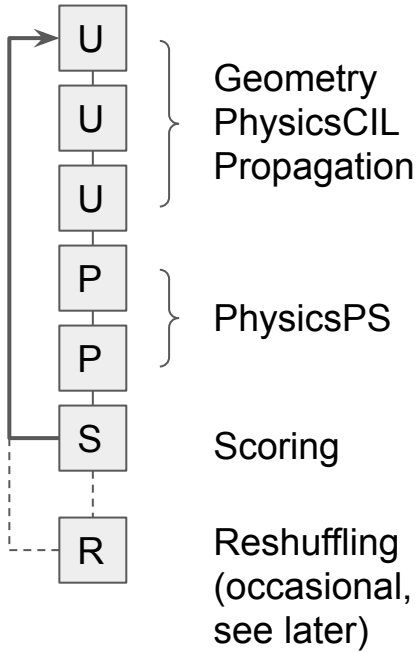
Producer kernels (P): modify + produce tracks

Scoring kernels (S): produce hits output

Reshuffle kernel (R):compact state data/cleanup/feed

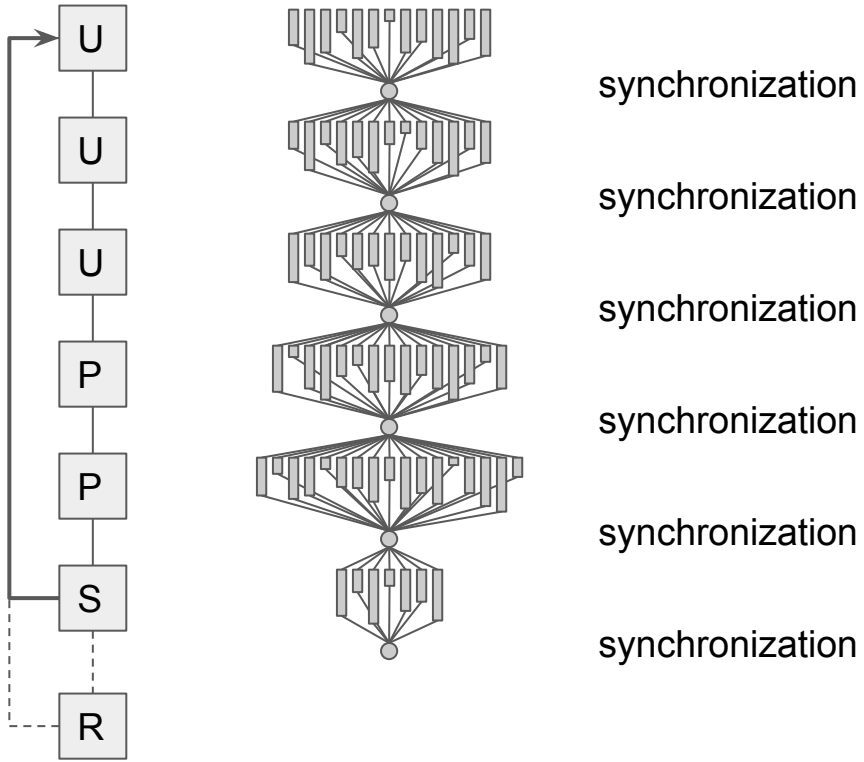
- Kernel execution order per track cannot be overlapped
- Each kernel to be executed for a (dynamic) population of tracks
- Most kernels may mark tracks as killed

Pipeline = stepping actions



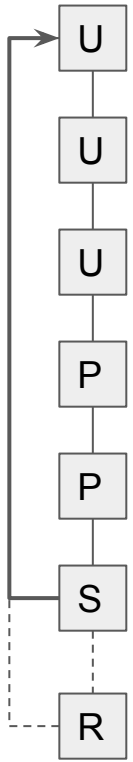
Parallelism

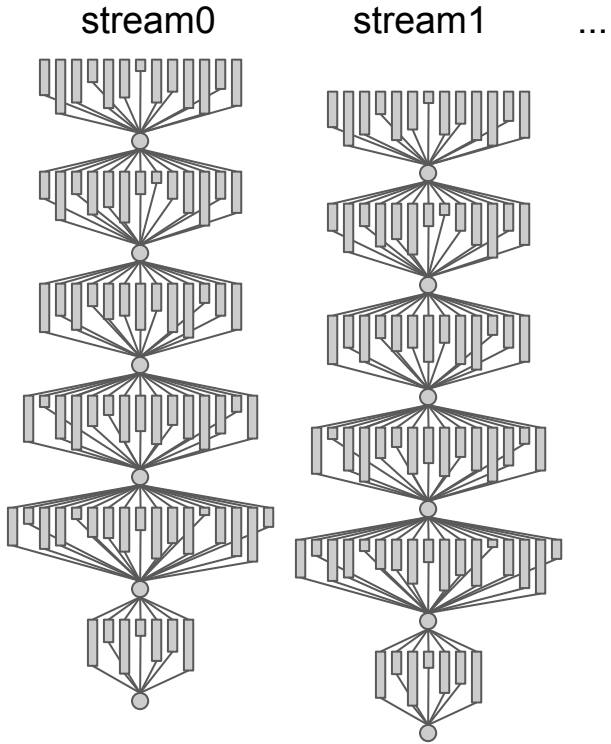
- Warp execution is coherent (SIMT)
  - waiting for slowest branch to finish
- Kernel execution in the same stream is coherent
  - Waiting for last scheduled warp to finish before next kernel is scheduled
- Potentially large inefficiency



Overlapping CUDA streams

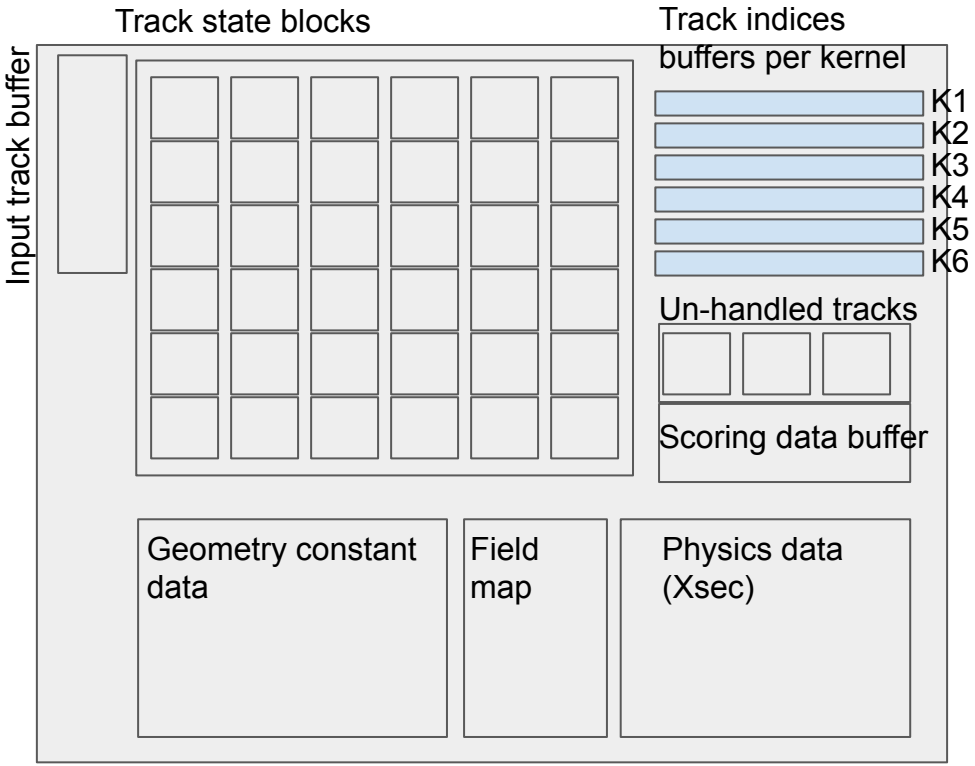
- CUDA stream ~ CPU process
- Work in separate streams can be overlapped
  - Keeping the device oversubscribed
  - Improving occupancy
- Requires splitting the track populations in several separate streams
  - Avoid using synchronized data structures for tracks





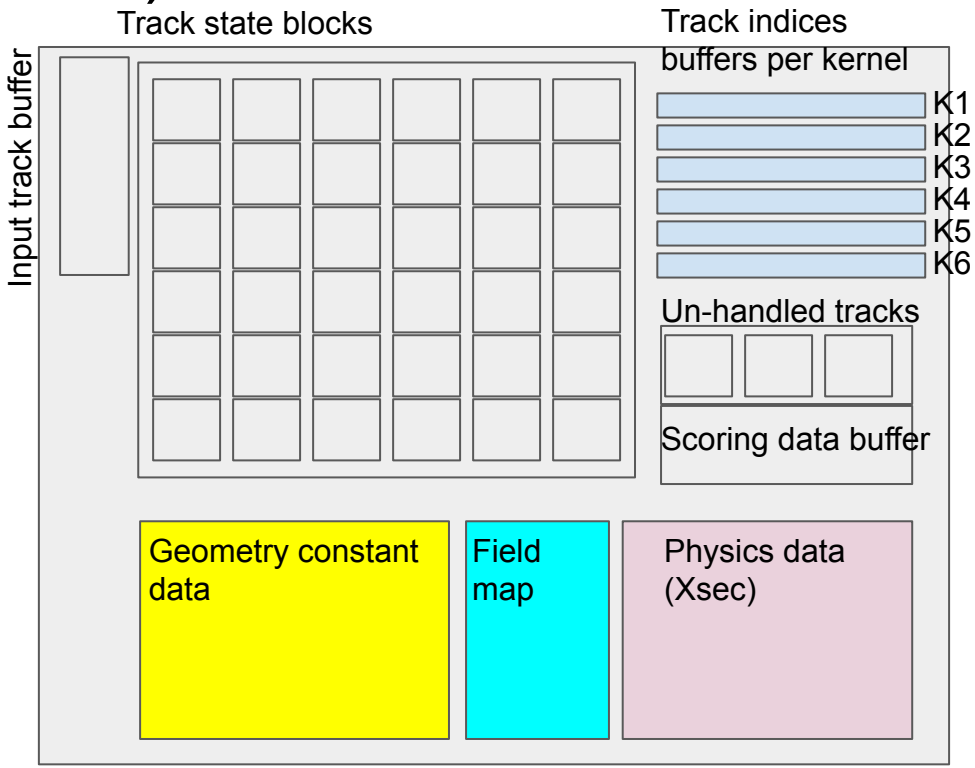
Memory usage

- Constant data
  - Geometry, field map, xsec
- State data (tracks)
  - Pre-allocated in contiguous blocks
  - Producer kernels get fresh new blocks for adding new tracks
  - Killed tracks create holes, blocks need to atomically account them
- Tracks may overspill back to CPU
  - Un-handled, excess
- Scoring data



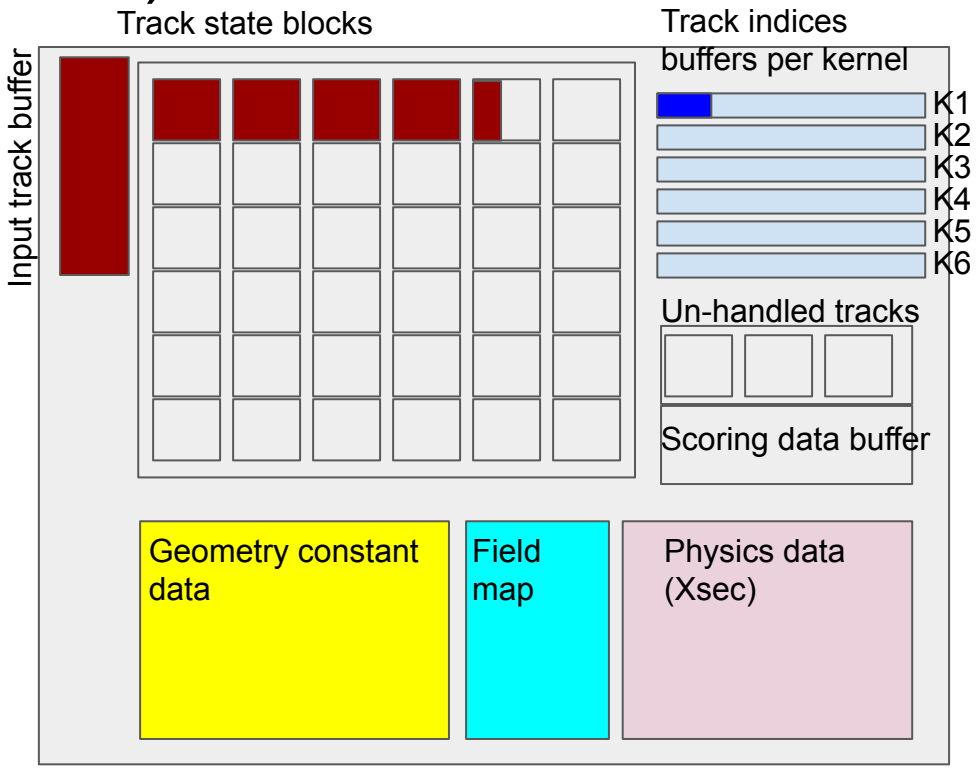
Workflow per stream (sketch)

- Populate constant data on the device



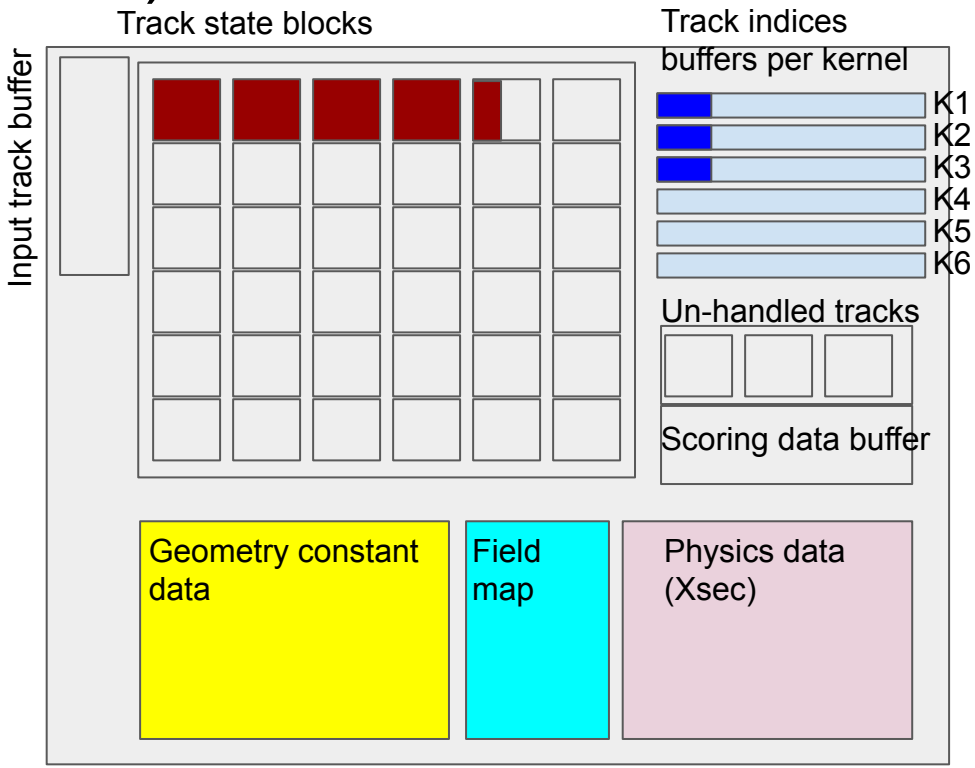
Workflow per stream (sketch)

- Populate constant data on the device
- Copy track states from the host
  - And their indices in the first kernel buffer



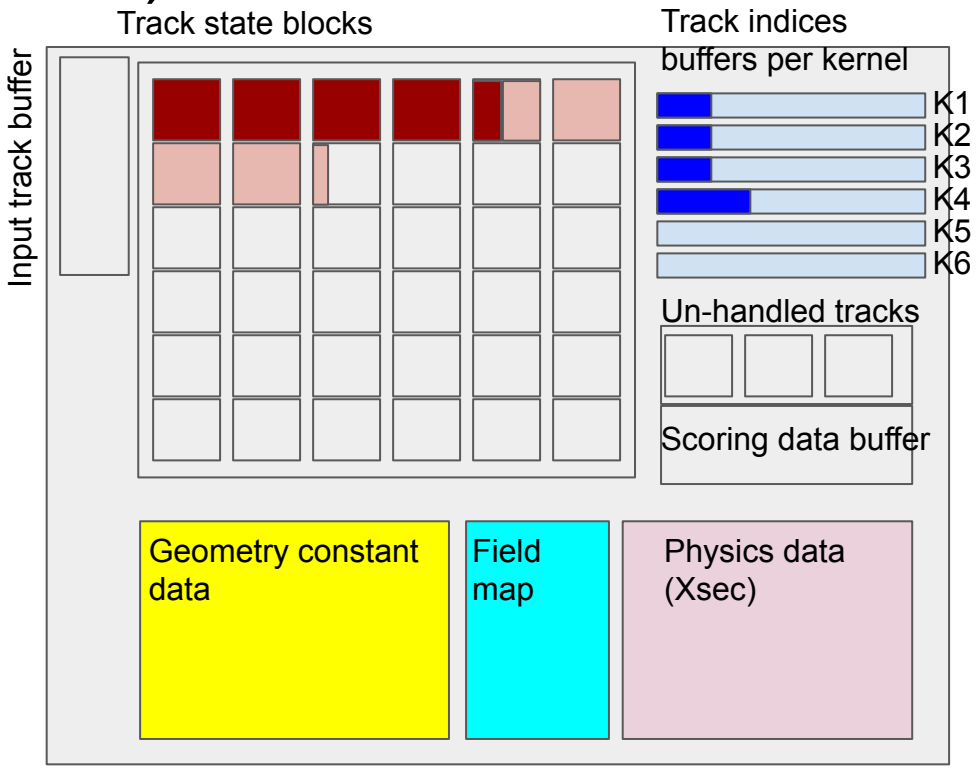
Workflow per stream (sketch)

- Populate constant data on the device
- Copy track states from the host
  - And their indices in the first kernel buffer
- Schedule the current kernel
  - U - copy indices to following buffer



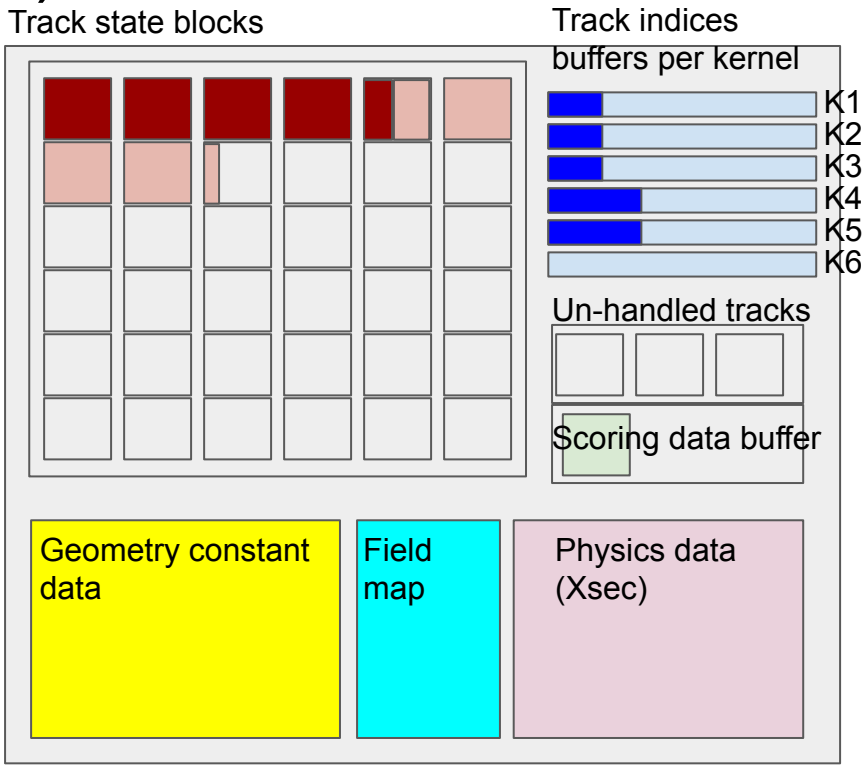
Workflow per stream (sketch)

- Populate constant data on the device
- Copy track states from the host
  - And their indices in the first kernel buffer
- Schedule the current kernel
  - U - copy indices to following buffer
  - P - Get ownership of a free track slot, then write new track to it



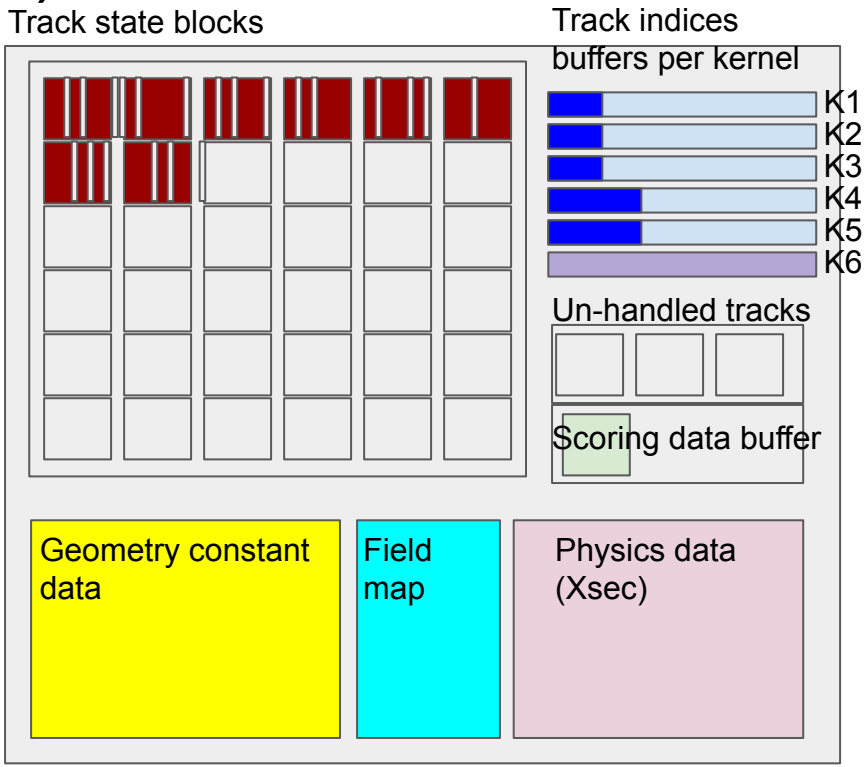
Workflow per stream (sketch)

- Populate constant data on the device
- Copy track states from the host
  - And their indices in the first kernel buffer
- Schedule the current kernel
  - U - copy indices to following buffer
  - P - Get ownership of a free track slot, then write new track to it
  - S - scoring kernels need to fill concurrently the hits data structure



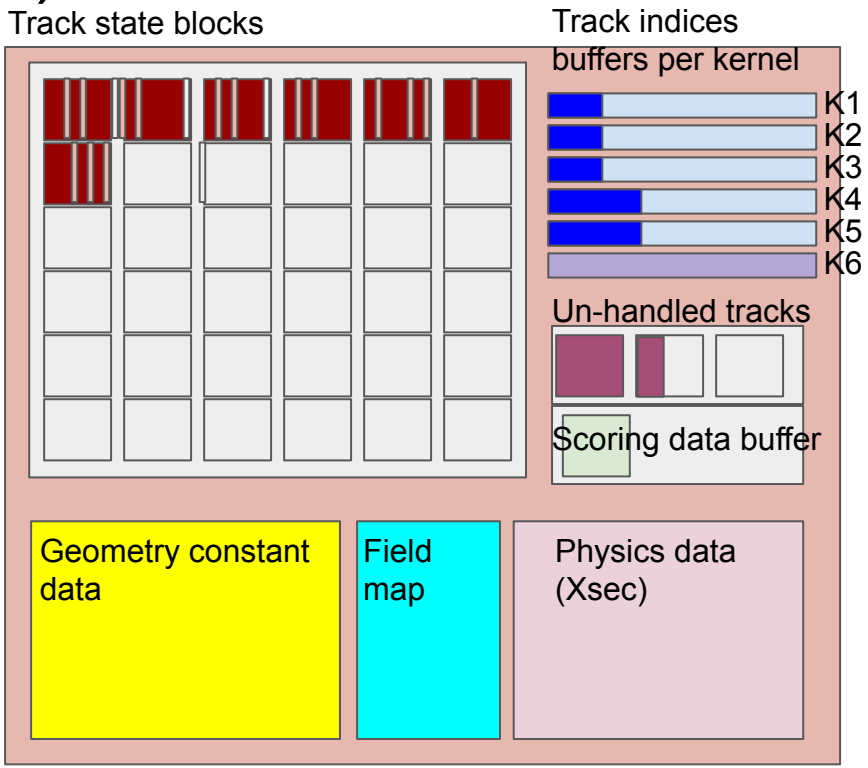
Workflow per stream (sketch)

- Tracks may be killed during the processing, creating holes
  - Need ‘hole management’ queues
- Reshuffling/cleanup/feeder kernel run occasionally
  - May need to refill leftover holes, feed new tracks, collect results



Workflow per stream (sketch)

- Tracks may be killed during the processing, creating holes
  - Need ‘hole management’ queues
- Reshuffling/cleanup/feeder kernel run occasionally
  - May need to refill leftover holes, feed new tracks, collect results
- Un-handled tracks may spill back to host
- Scoring data copied to host



Conclusions

- Bootstrapping a more coherent R&D effort to understand usability of GPUs in HEP simulation
  - AdePT prototype as playground to implement a GPU EM calorimeter simulation workflow, feeding in tracks and getting out hits
- Just the repo and some design ideas so far
  - Based on previous experience with ray-tracing on GPU and other prototypes
  - A fresh start, all developers who can/want to contribute are very welcome
  - Fisher-Price approach first to develop a simulation skeleton with the desired features
- Work on the different modules should become factorizable
  - Start with toy models, improved to reach performance standards to be defined