Traccc: GPU Track Reconstruction Library

Beomki Yeo

UC Berkeley and LBNL

On behalf of the traccc team

Motivation for Hardware Accelerator in HEP Computing

- Hard to sustain computing budget model with traditional CPUs in future experiments (e.g. High Luminosity LHC)

- Hardware accelerators such as GPU are expected to outperform the CPUs

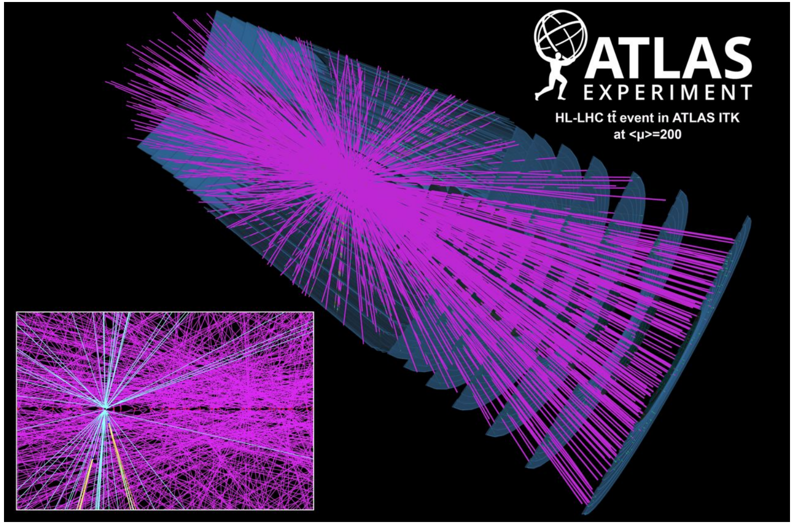


Image credit

What is GPU?

- Graphical Processing Unit
  - a.k.a graphics card
  - Major Vendors: NVIDIA, AMD, and Intel

- Invented for the visualization on PC screen
  - Now also used as a hardware accelerator

- Two types of GPU
  - Integrated in CPU
  - Discrete GPU

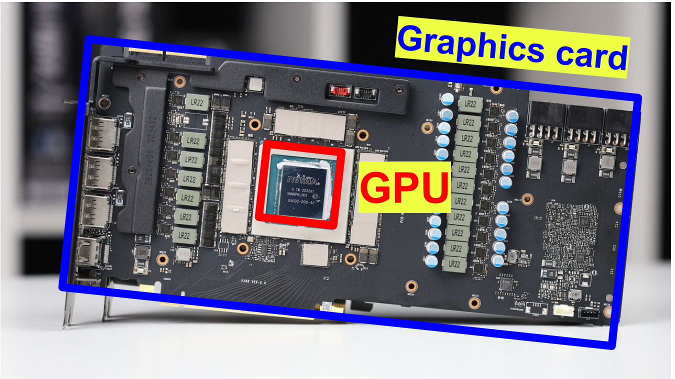
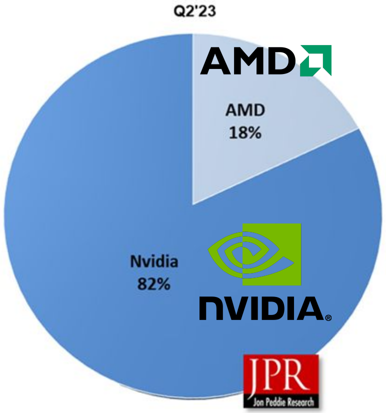
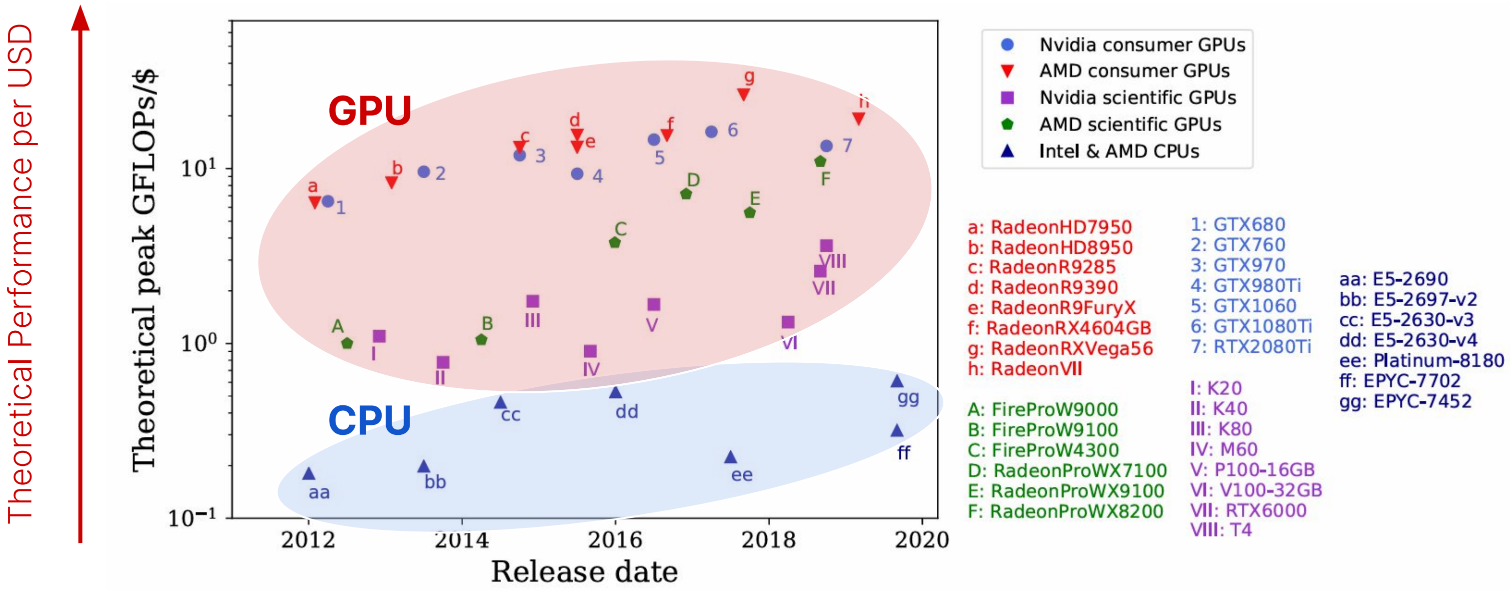


Image credit



Discrete GPU Market Share

Why GPU? (Performance per USD)



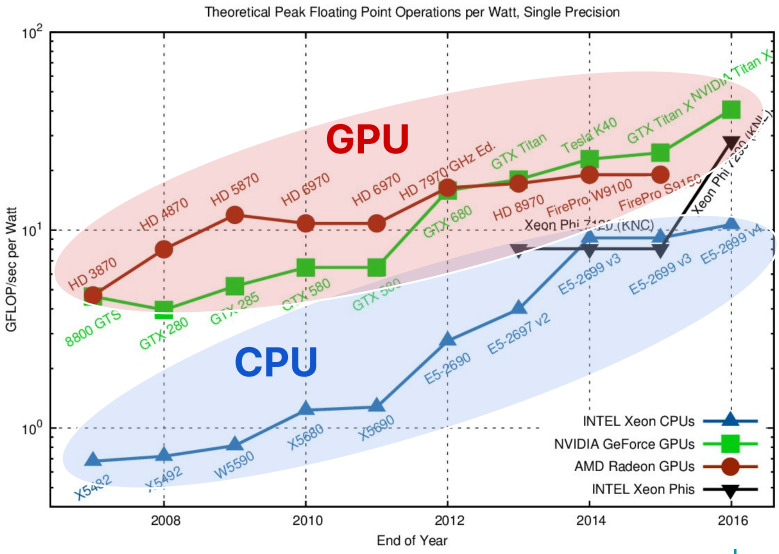
arXiv:2003.11491

Theoretically GPU is faster and more economical than CPU

Why GPU? (Performance per Watt)

Theoretical Performance per Watt

Single (FP32)



Double (FP64)

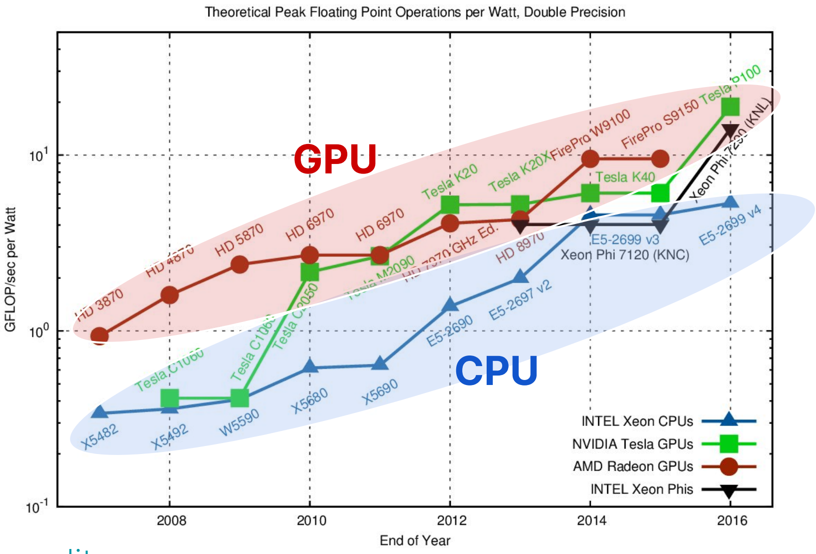
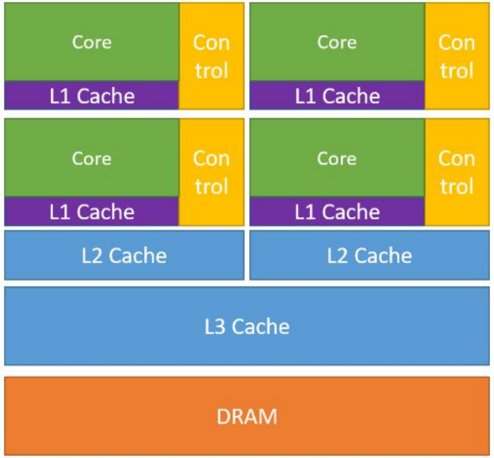


Image credit

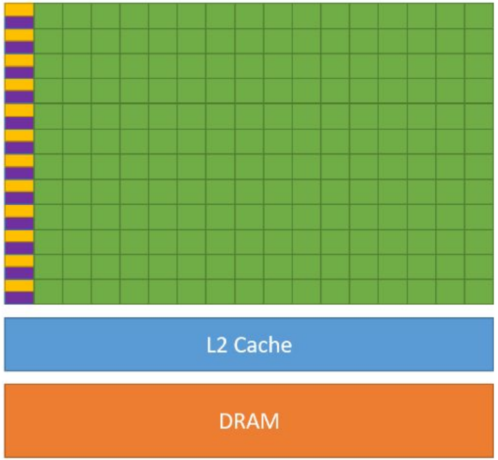
Theoretically GPU is faster and more economical than CPU

CPU vs GPU

- **CPU**
  - Small number of powerful cores (~10)
    - Branch prediction, register renaming, etc.
  - Large caches
- **GPU**
  - Many number of cores (≳1000)
    - *A lot* simpler
  - Small caches



CPU



GPU

Image credit

Practically GPU can outperform CPU with parallelizable and relatively simple algorithms

ACTS – A Common Tracking Software

- ACTS: C++ based track reconstruction toolkit for any experiment
  - General info about ACTS is covered in A. Salzburger's talk on Thursday
  - ATLAS Phase-II is the primary target (Offline & Online reconstruction)

- Most of development had happened with CPU
  - Also interested in implementing the CPU algorithms into hardware accelerators such as **GPU**

- Faced with following challenges during the initial attempts for GPU implementation:
  - Geometry and event data model designed with runtime polymorphism (*GPU-unfriendly*)
  - Difficult to recycle the original CPU algorithms for GPU

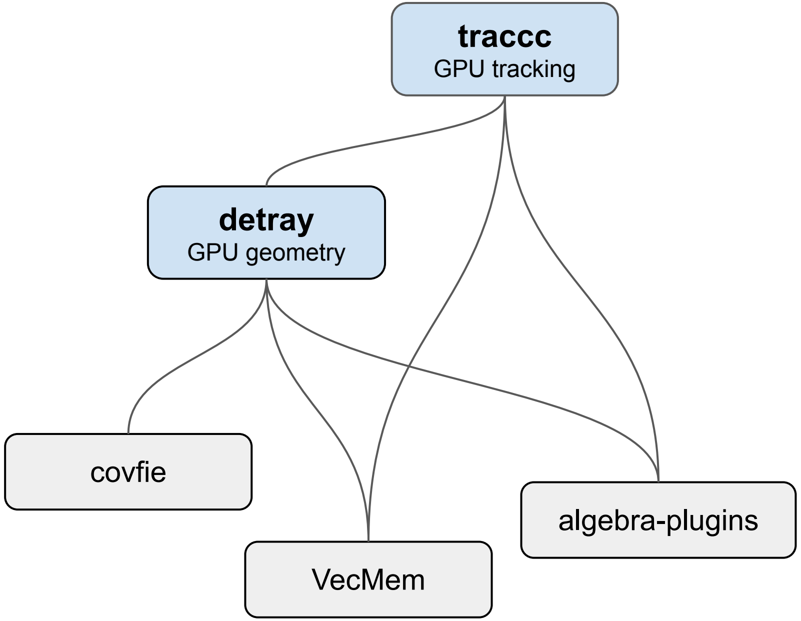
- Such challenges were the motivation for dedicated GPU R&D projects

ACTS GPU R&D Projects

- **traccc**
  - Library for the GPU tracking demonstrator
  - Candidate for ATLAS Online reconstruction

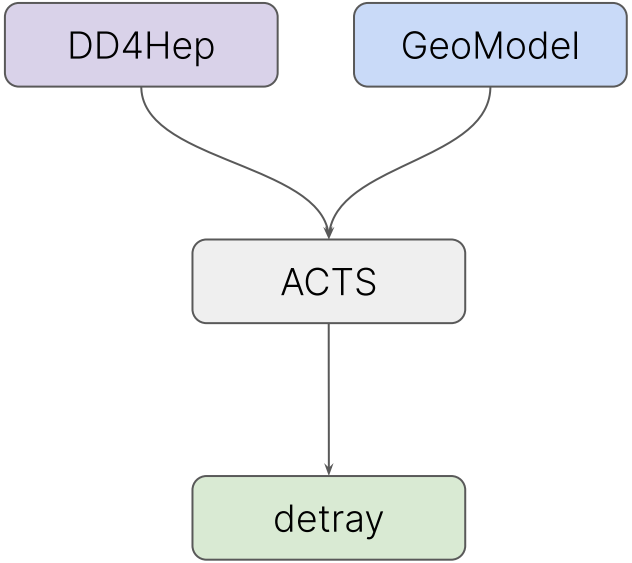
- **detray**
  - Library for the GPU tracking geometry
  - Compile-time polymorphism design

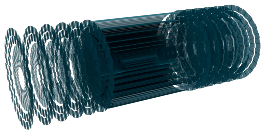
- There are also other cool R&D libraries (VecMem, covfie and algebra-plugins) but not covered in the today's presentation



Detray – GPU Friendly Geometry

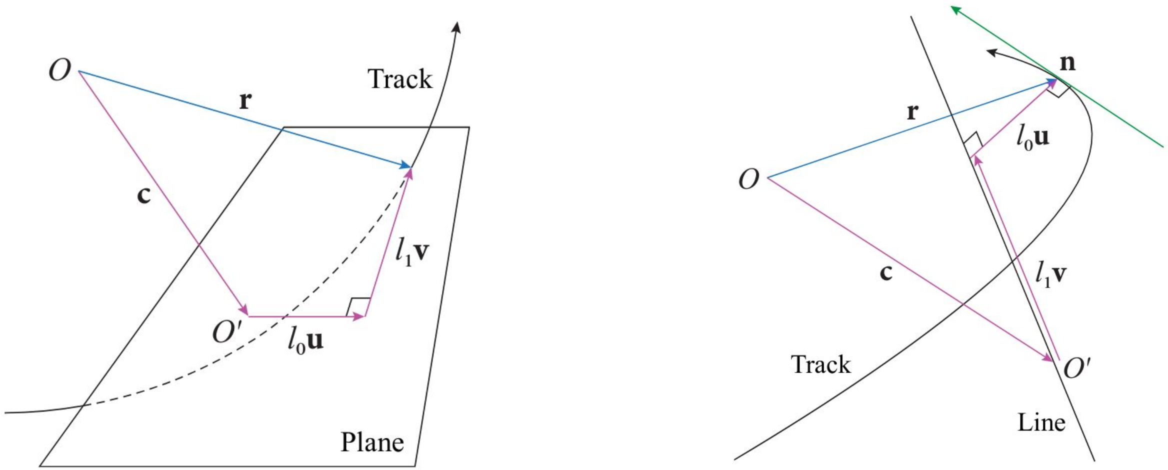
- Tracking geometry is very necessary for track finding and fitting
- **detray** moves out of the runtime polymorphism to adopt the compile-time polymorphism
- Index-based link between geometrical objects (No pointers!)
- Can translate ACTS geometry to utilize the existing plugins for other geometry libraries
  - DD4Hep
  - GeoModel (ATLAS-specific)





Detray – Track Propagation

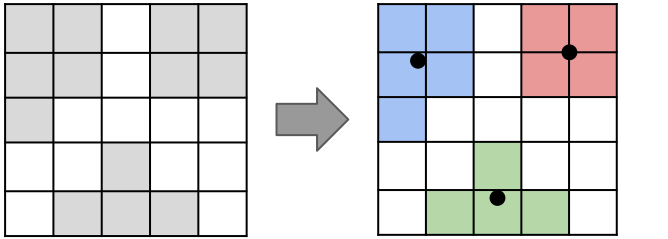
- Fourth order Runge-Kutta stepper inside the Inhomogeneous B field and material
- Material interaction: Bethe energy loss and multiple scattering
    - Non-gaussian noise (e.g. Brems) will be added in the near future
- Supports various surface types: Bound frame (e.g. pixels), Perigee frame (e.g. straw tubes and drift chamber)



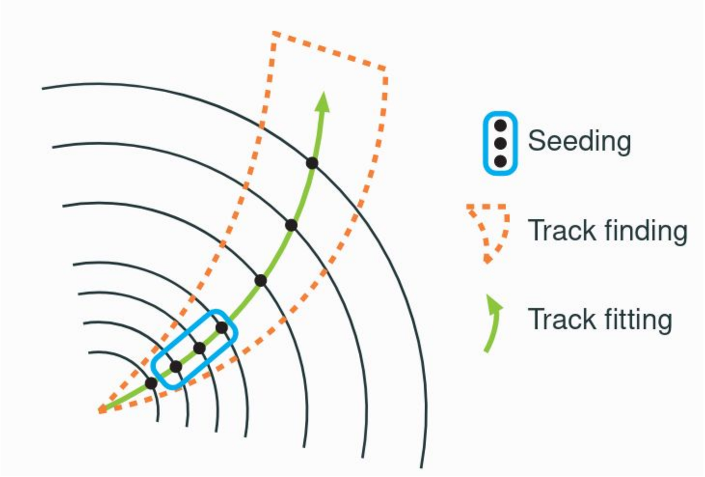
Bound (left) and perigee (right) frame with a track intersecting

Traccc – Algorithms

1. Hit clusterization
   - Creating measurements from pixel readouts
2. Seeding (Pattern recognition)
   - Finding *three* measurements of a single particle
3. Track finding (Pattern recognition)
   - Finding all measurements of a single particle
   - **Combinatorial Kalman Filter**
4. Track fitting
   - Fitting the measurements to a track
   - **Kalman Filter**



2D Hit clusterization for measurement creation

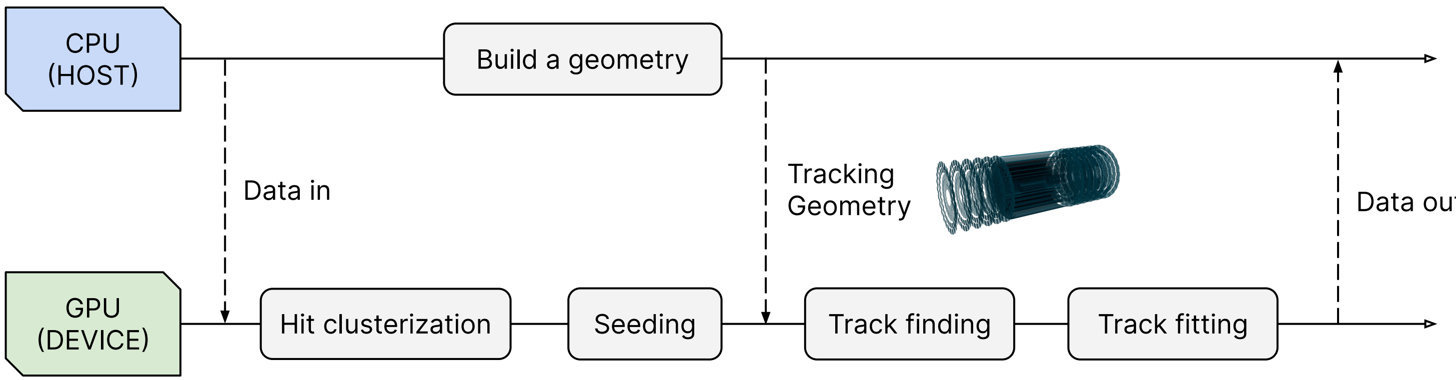


Pattern recognition and fitting
Tracking algorithm in a nutshell

Traccc – Workflow

- End-to-end analysis in the GPU
  - Minimize the memory transfer between CPU and GPU

- The tracking geometry is transferred to the GPU only once and the same geometry on the device is used for all events



Traccc – Development Status

- Mainly developed for **CUDA** (Nvidia exclusive) and **SYCL** (all vendors)
  - Ongoing development for portability solutions such as Alpaka, Kokkos and Futhark

- Supports both single (float) and double precisions

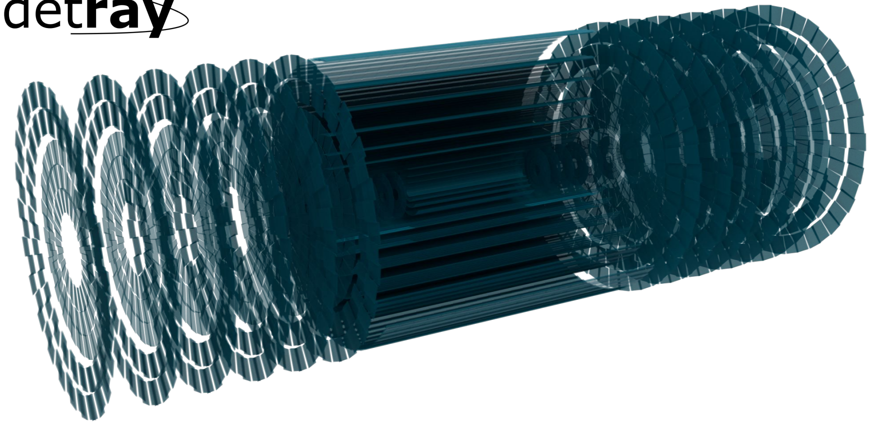
Category	Algorithms	CPU	CUDA	SYCL
Clusterization	CCL / FastSv / etc.	✅	✅	✅
	Measurement creation	✅	✅	✅
Seeding	Spacepoint formation	✅	✅	✅
	Spacepoint binning	✅	✅	✅
	Seed finding	✅	✅	✅
	Track param estimation	✅	✅	✅
Track finding	Combinatorial KF	✅	✅	🟡
Track fitting	KF	✅	✅	✅

Showcase: Open Data Detector

- Freely available open-source detector
  - LHC-like geometry
  - Developed by ACTS community for algorithm research and development

- Traccc team recently demonstrated full GPU tracking chain with the open data detector

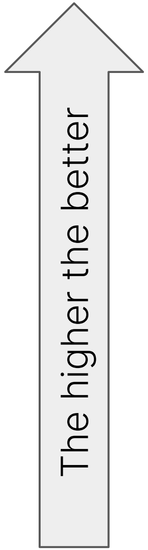
detray

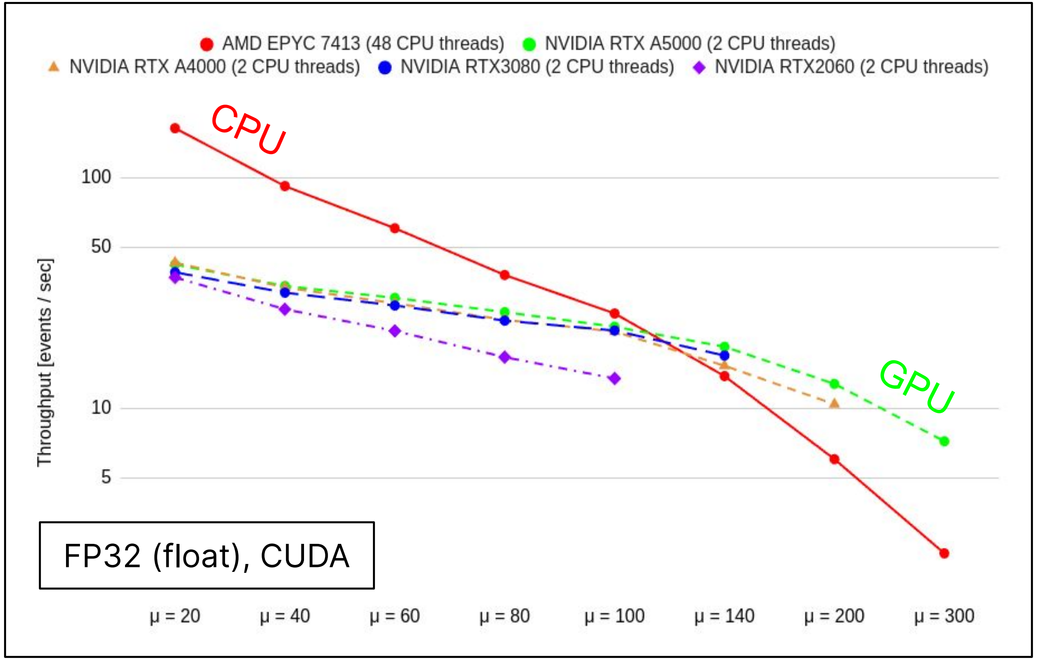


[HSF workshop, (May 16th 2024)]

Showcase: Open Data Detector (cont.)

- GPU throughput is higher than CPU for large size of data (pileup ≳ 140, corresponding to the ATLAS Run 4 data size)

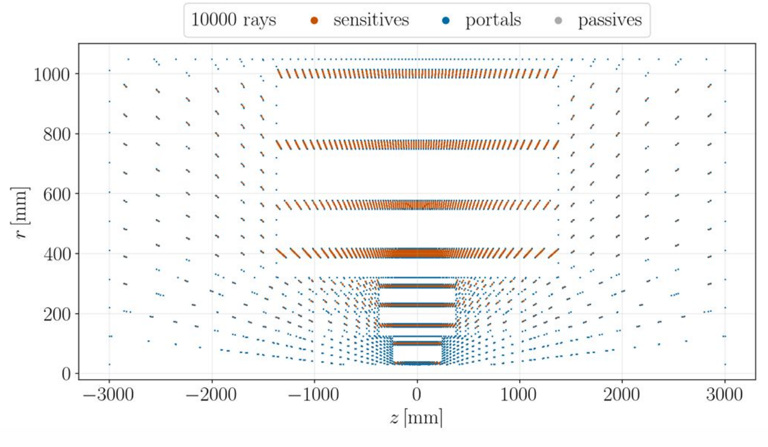




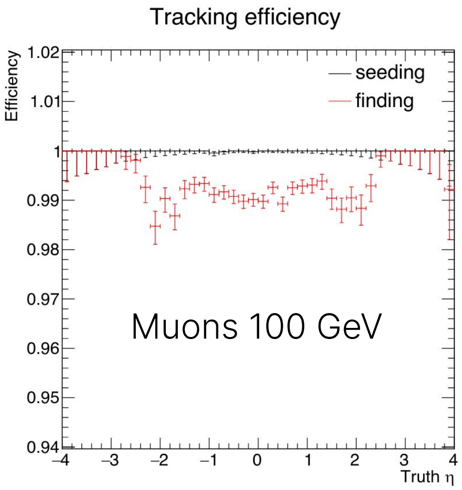
[HSF workshop, (May 16th 2024)]

Showcase: ATLAS Experiment

- High Level Trigger for High Luminosity LHC
- Has been Investigated only for physical performance
  - Once GeoModel is fully supported, GPU speed performance can be measured



Detray ray tracing in the ATLAS geometry
Courtesy of J. Niermann

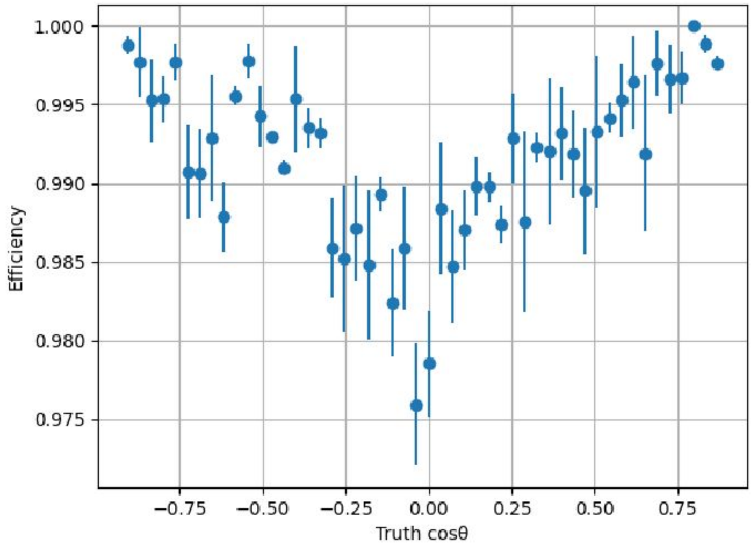


Particle reconstruction efficiency
of seeding and track finding
Courtesy of S. Shimizu

Showcase: CEPC

- Demonstrated seeding with the CEPC pixel geometry

- Initial performance looks good
  - See Y. Zhang's talk for detailed updates



Particle reconstruction efficiency of seeing algorithm with CEPC pixel geometry
ACTS Parallelization Meeting (Mar 8th 2024)

Summary

- traccc is the GPU track reconstruction library as an R&D project of ACTS

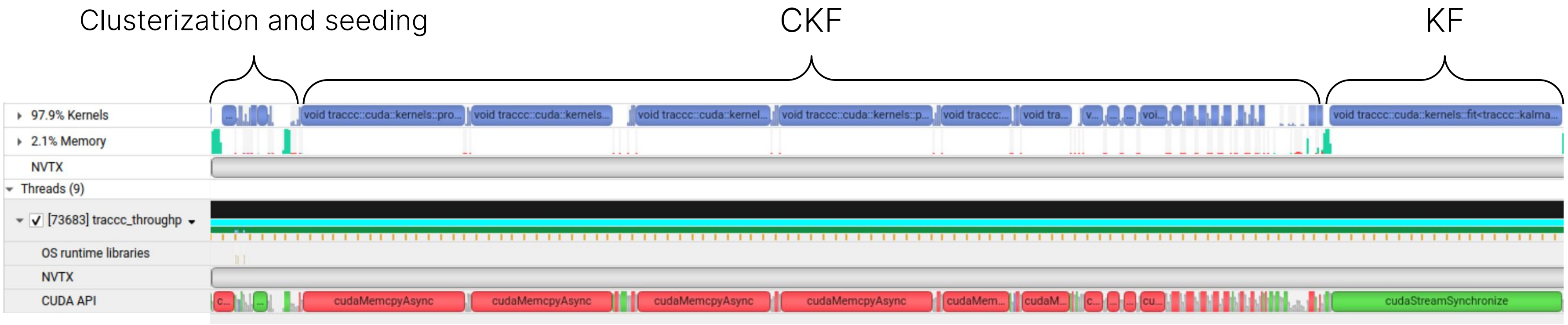
- Recently showed promising results of **full GPU tracking chain** with the Open Data Detector
  - Getting into the new R&D phase: performance optimization

- We welcome the participation of all experiments interested in GPU track reconstruction
  - ACTS Parallelization Meeting (Biweekly Friday 16:00 CET / 22:00 Shanghai)

Backups

GPU Profiling Result for an Event

- Most of time is spent in GPU Kernels
- Main bottlenecks: track finding (CKF) and track fitting (KF) kernels
  - **track propagation** (e.g. matrix operations) is expensive



One event timeline (~0.1 sec per event)