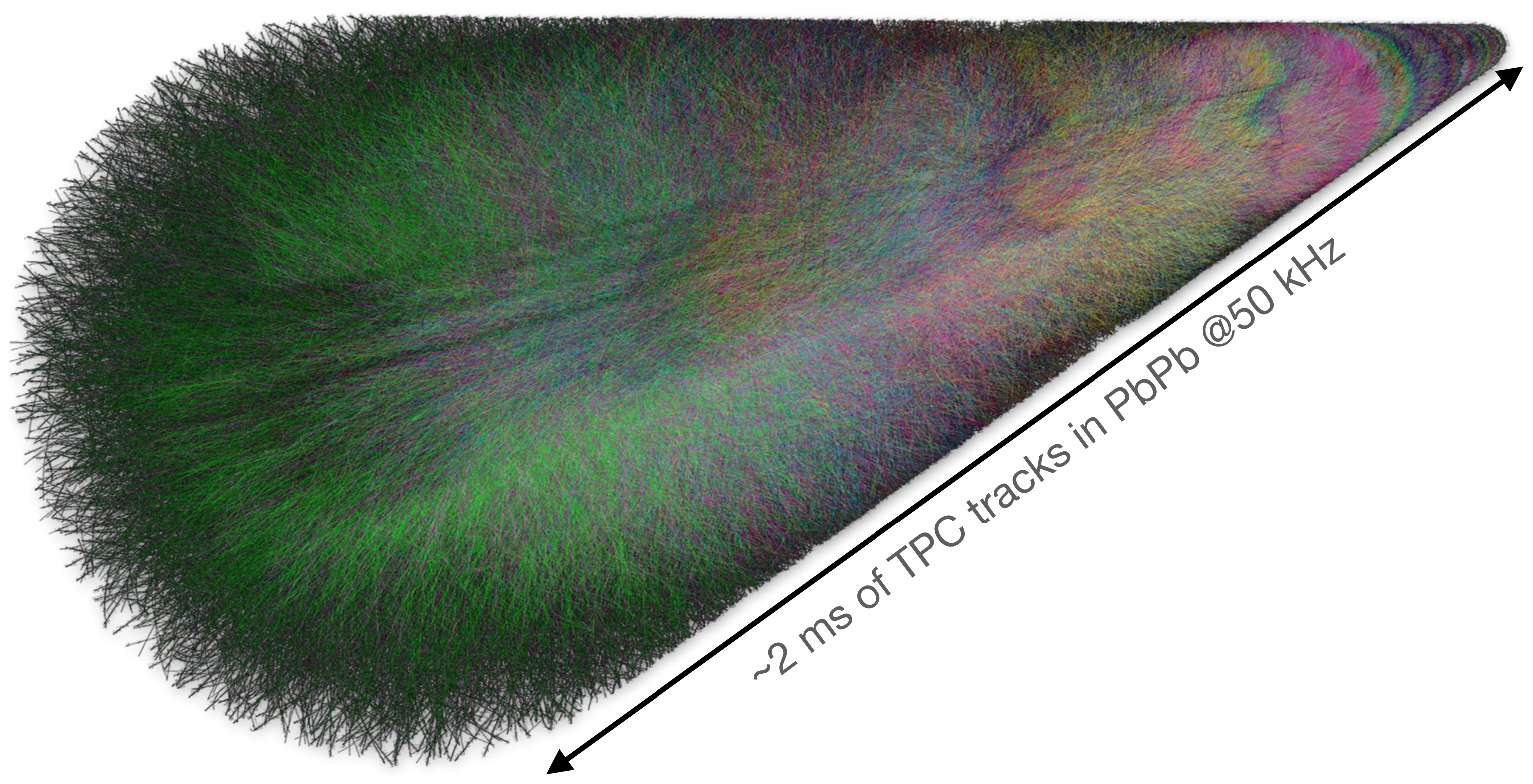
A vendor-unlocked GPU reconstruction for the ALICE Inner Tracking System

Matteo Concas, for the ALICE collaboration

CHEP 2023, May 9th

ALICE reconstruction using GPU in Run 3

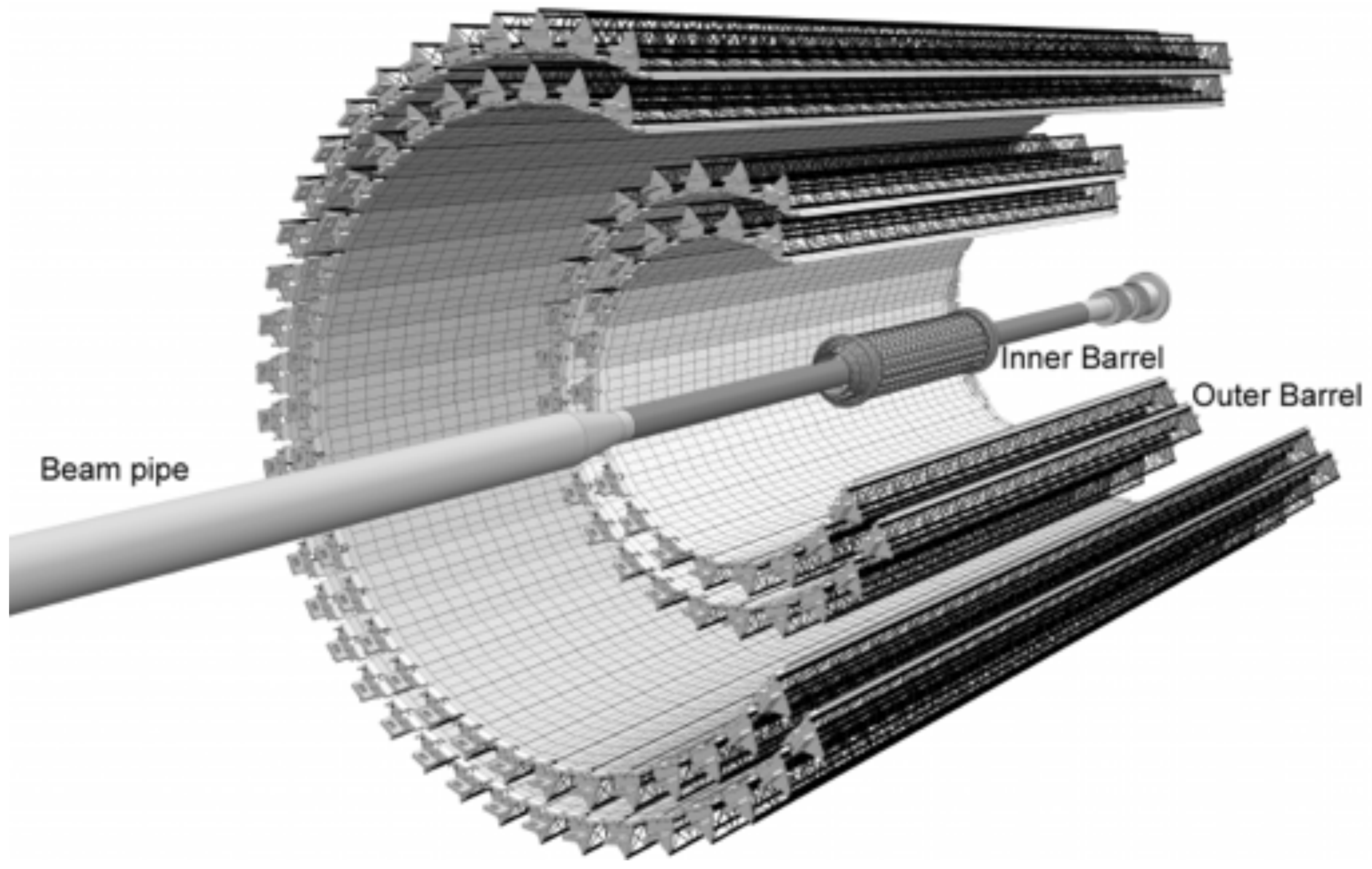
- Trigger-less acquisition: continuous readout
  - The stream of data is split into O(10ms) timeframes
  - $L_{int} > 10\ nb^{-1}$ of PbPb data at 50kHz: 50x more than Run 2
- Reconstruction is two-stepped
  - Synchronous phase (beam circulating): for calibration and data compression
  - Asynchronous phase (no beam): full processing of data staged on a temporary buffer
- ALICE uses GPUs to accelerate the process[1]
  - During the asynchronous reconstruction, the fraction of available GPU increases
  - Use those resources efficiently by offloading ITS reconstruction there



[1] "The O2 software framework and GPU usage in ALICE online and offline reconstruction in Run 3"

ITS reconstruction in Run 3

- A new upgraded Inner Tracking System
  - A cylindrical silicon detector with 12.5 billion pixels and 10 m^2 of sensitive area
  - Provide spatial information in the form of clusters of fired pixels
- Continuous readout: continuous track reconstruction
  - The atomic time unit is Readout Frames (ROF): ~4μs
- Standalone vertexing and tracking algorithm
  - During the synchronous phase, 1% of primary tracks are reconstructed
  - During the asynchronous phase is sensitive to secondaries and tracks lower p_T



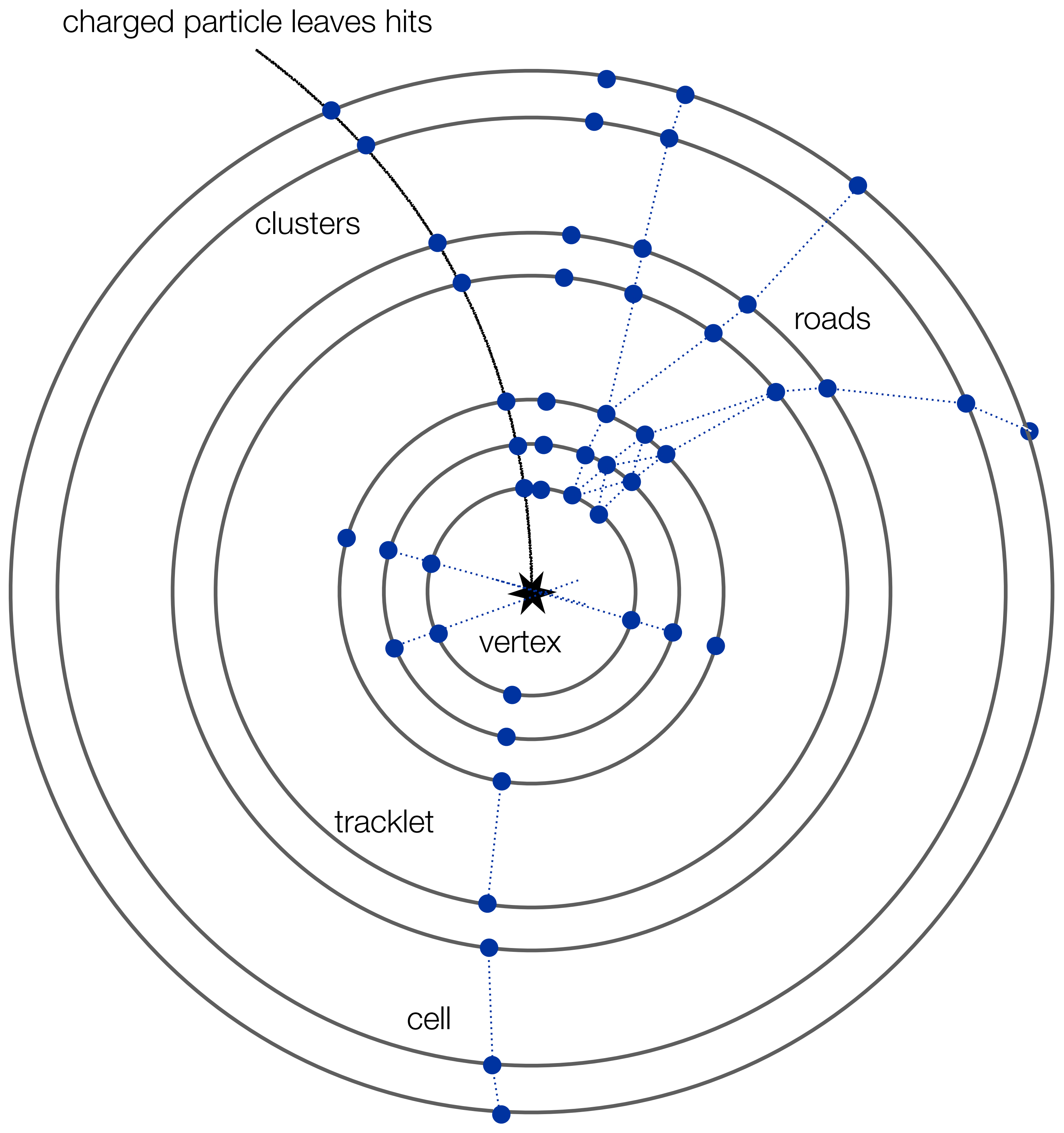
ITS tracking

Timeframe

ROF 0	ROF 1	ROF ...	ROF N
- clusters - vertices - tracklets - cells - roads **- tracks**	- clusters - vertices - tracklets - cells - roads **- tracks**	...	- clusters - vertices - tracklets - cells - roads **- tracks**

ITS vertexing and tracking

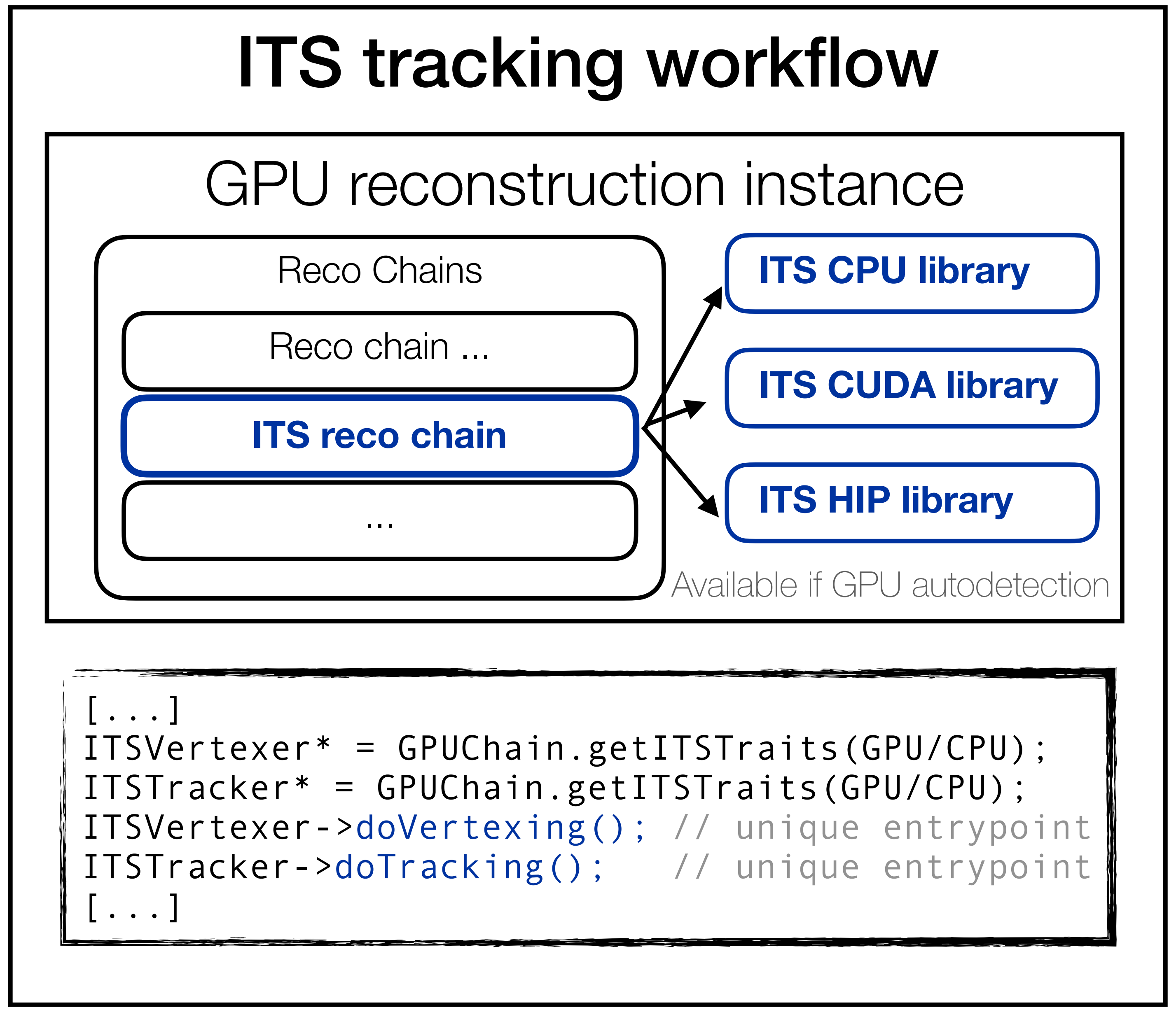
- **Primary vertex seeding**
  - Combinatorial matching followed by linear extrapolations of *tracklets*
  - Unsupervised clustering to find the collision point(s)
- **Track finding and track fitting**
  - It uses vertex position to reduce the combinatorics in matching the hits
  - Connect segments of tracks, the *cells*, into a tree of candidates: *roads*
  - Kalman filter to fit tracks from candidates
- **The algorithm is decomposable into multiple parallelisable steps**
  - Each ROF can be processed independently(*)
  - In-frame combinatorics can be processed simultaneously



(*) Information from adjacent ROFs can be used to recover from information splitting

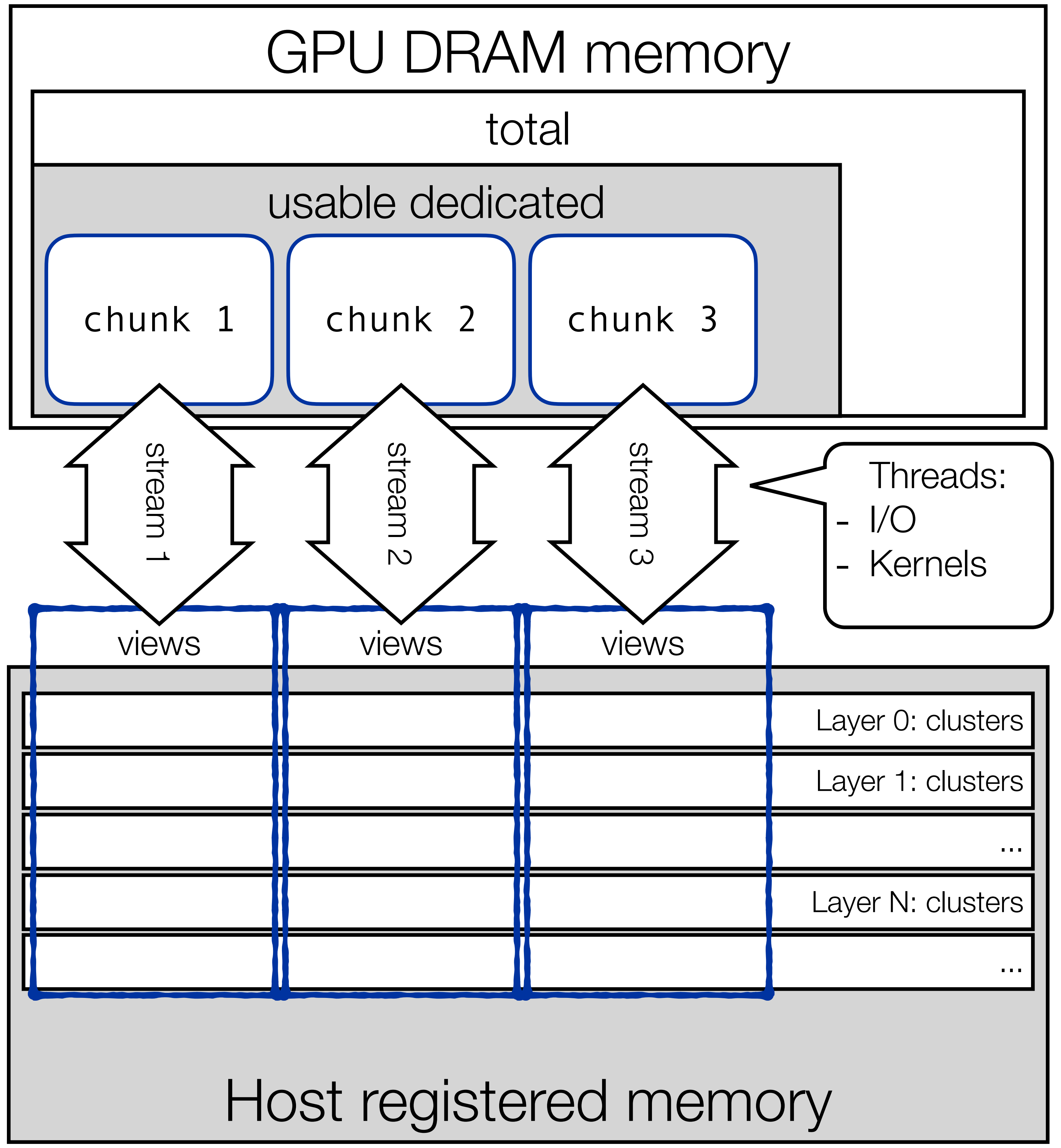
A parallel implementation using GPUs

- Yesterday: accelerate processing using parallel architectures
  - Promising porting of some routines based on CUDA and OpenCL in the past
- Today: operate a plug-in standalone GPU tracking for ITS
  - Mainstream reconstruction framework provides the interface for GPU lib loading
  - Supports CUDA and HIP with a single code base
- Tomorrow: build a GPU reconstruction chain, including ITS
  - Centrally manage GPU memory and kernel scheduling for deeper integration
  - Easier to later integrate additional steps like the ITS-TPC matching



Cornerstones of the GPU implementation

- Resource usage flexibility via configuration
  - The amount of usable memory is a parameter that is passed to the algorithm
  - All required chunk sizes are set as a fraction of the total available memory
- Multi-threaded streams process bunches of ROFs in parallel
  - Each POSIX thread manages a stream, and the full tracking is independent
  - I/O operations on one stream are hidden behind kernel executions
- Use case extensibility via a generic *N*-layers implementation
  - `TrackerGPU<NLayers>` offers native support for future use cases (ITS3/ALICE3)



Cross-platform on-the-fly code generation

ALICE

- The O2 compilation via CMake, provides
  - Platform autodetection and production of corresponding target libraries
  - Custom commands setting dependencies between targets
- HIP code is generated in place from CUDA sources
  - Build source of targets parsing CUDA files and generating HIP versions
  - Currently based on `hipify-perl`: is run on all `.cu` files to produce HIP
- Headers files are shared across both the compilations
  - Negligible boilerplate (<0.1% LoCs) to cope with some architectural differences

```
// CUDA code
cudaMalloc(&A_d, Nbytes);
cudaMalloc(&C_d, Nbytes);
cudaMemcpy(A_d, A_h, Nbytes, cudaMemcpyHostToDevice);

vector_square <<<512, 256>>> (C_d, A_d, N);
cudaMemcpy(C_h, C_d, Nbytes, cudaMemcpyDeviceToHost);

// HIP code, translated
hipMalloc(&A_d, Nbytes);
hipMalloc(&C_d, Nbytes);
hipMemcpy(A_d, A_h, Nbytes, hipMemcpyHostToDevice);

hipLaunchKernelGGL(vector_square, 512, 256, 0, 0, C_d, A_d, N);
hipMemcpy(C_h, C_d, Nbytes, hipMemcpyDeviceToHost);
```

State of the development and testing

- GPU implementations are more complex due to data organisation
  - Naming is shared when processing steps reach the same output
- The vertexing is fully operative in its GPU implementation
- The porting of tracking is being finalised
  - Road finder is under development: size and number of found roads are not static
  - Track fitting had a POC, which requires an in-depth review
- Tested on both Nvidia and AMD cards
  - First setup: workstation with AMD Ryzen™ 9 7950X CPU and Nvidia™ TITAN Xp
  - Second setup: EPN node with 2x AMD EPYC™ 7452 and AMD Instinct™ MI50

Vertexer	
Tracklet Finder	✅
Tracklet Selection	✅
Vertex Fitter	✅

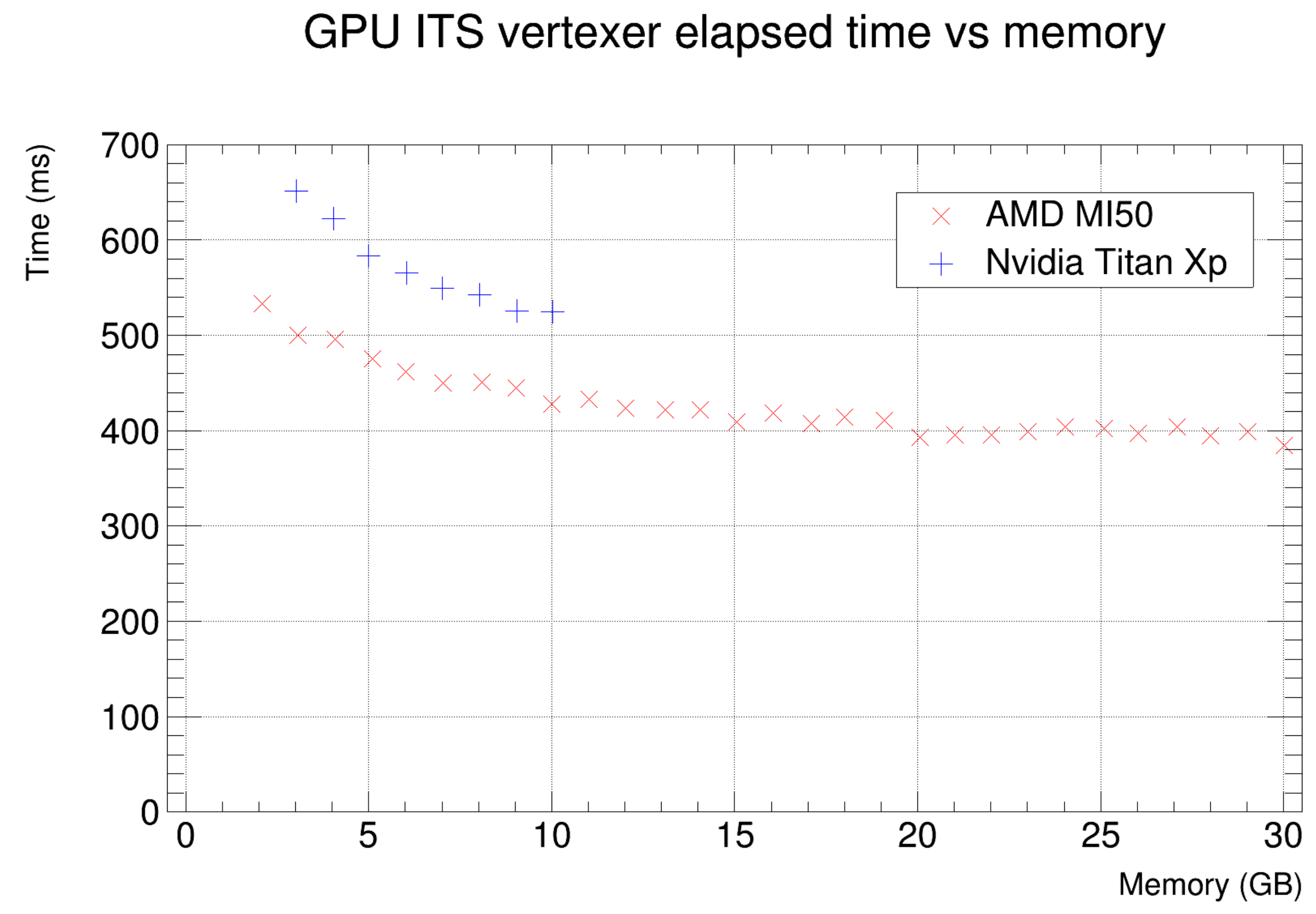
Tracker	
Tracklet Finder	✅
Trkl duplicate finder	✅
Cell finder	✅
Cell neighbour finder	✅
Road finder	🚧
Track fitting	✳️

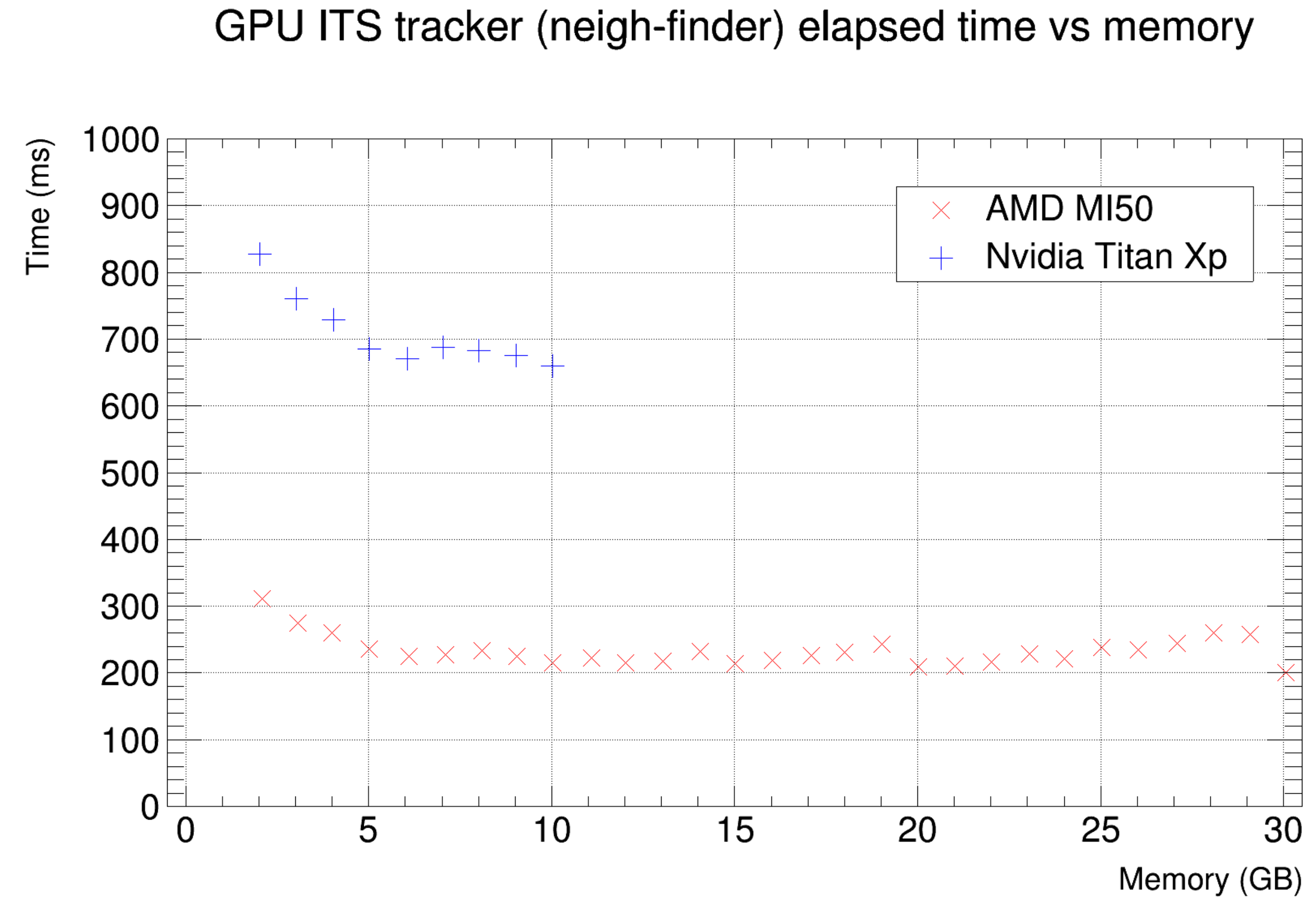
	Clock (GHz)	RAM (GB)
AMD Ryzen™ 9 7950X	4.5-5.7	128
Nvidia™ TITAN Xp	1.586	12
AMD EPYC™ 7452	2.35-3.25	512
AMD Instinct™ MI50	1.725	32

Preliminary performance

Elapsed Time [ms]	AMD EPYC™	AMD Ryzen™	AMD MI50	Nvidia™ TITAN Xp
Vertexer	2913±376	1416±183	291±38	478±64
Tracker (Neigh. Finder)	550±71	287±37	211±27	779±105
Tracker Full	13756±1780	6917±893	🚧	🚧

- Total timing measured on real data
  - A batch of 5 timeframes of *pp* collisions @500kHz
  - CPU is run in single thread configuration
- Considerations
  - The timing is promising if the primary goal is to trade GPUs for CPUs
  - The most time-consuming part is the track fitting, high rewards expected
  - Streaming chunks of a timeframe works successfully
  - Timing decreases with memory increasing, then reaches a plateau





Conclusions and outlook

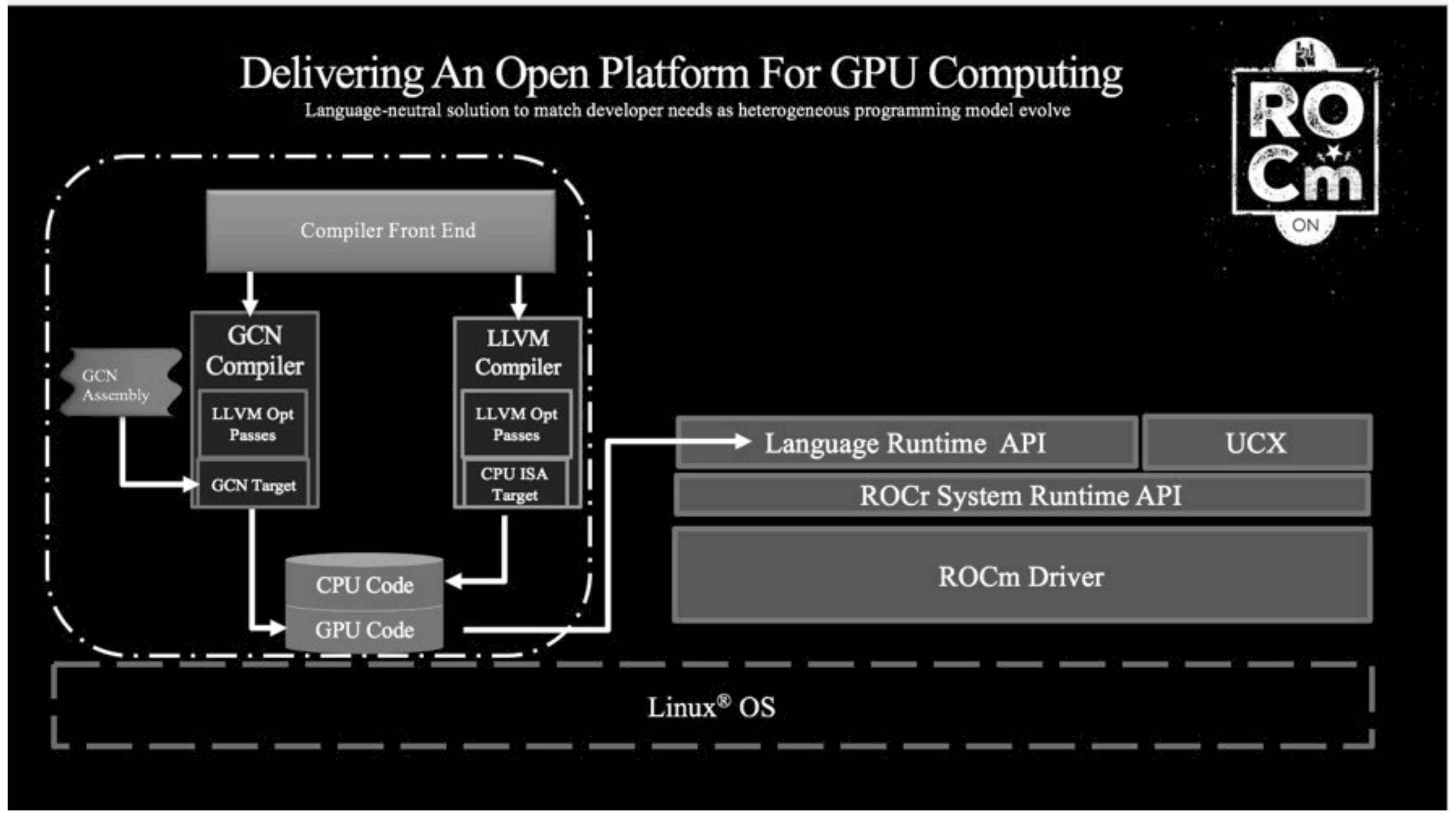
- ALICE plans to extend the coverage of GPU utilisation in the asynchronous reconstruction
  - The goal is to increase the efficiency in using the resources when TPC does not have the monopoly
- ITS is finalising the porting of the seeding vertexer and tracking
  - Road finding and track fitting, the last missing components, are under active development
  - Performance in *pp* collisions from actual data is not final but shows some promising margin
- Optimisation of the algorithms is to start after the finalisation of the porting
  - Tuning for GPU parameters can be performed with general-purpose tools for optimisation[1]
- GPU adoption in the ITS software chain can be further extended
  - Signal digitisation and Clusterisation part are good candidates that are being considered

[1] "A parameter optimisation toolchain for Monte Carlo detector simulation"

Backup

Heterogeneous-Compute Interface for Portability

- Support GPUs from two main vendors:
  - CUDA language and runtime for Nvidia
  - HIP language and ROCm runtime for AMD
- HIP: a C++ Runtime API and Kernel language
  - Portable AMD and NVIDIA applications from single source code
  - It is shaped around CUDA APIs to ease translation
  - CUDA libraries, like Thrust and CUB, have their HIP versions using ROCm
- ROCm has tools to translate CUDA to HIP automatically
  - `hipify-clang`: based on Clang, actual code translation
  - `hipify-perl`: script for line-by-line code conversion
- Strategy: maintain only the CUDA code and generate HIP



ALICE data processing for Run 3

- Online reconstruction and calibration for data compression
  - Synchronous: TPC full reconstruction and calibration
  - Asynchronous: all compressed data are reconstructed
  - Single computing framework for online-offline computing: O^2
- Operate part of the reconstruction on GPUs is *mandatory*
  - Minimise the cost/performance ratio for online farm
  - 250x Event Processing Nodes (EPNs), 8x AMD MI50 GPUs
- Efficient utilisation of available computing resources is desired
  - A larger fraction of GPUs available during the asynchronous phase