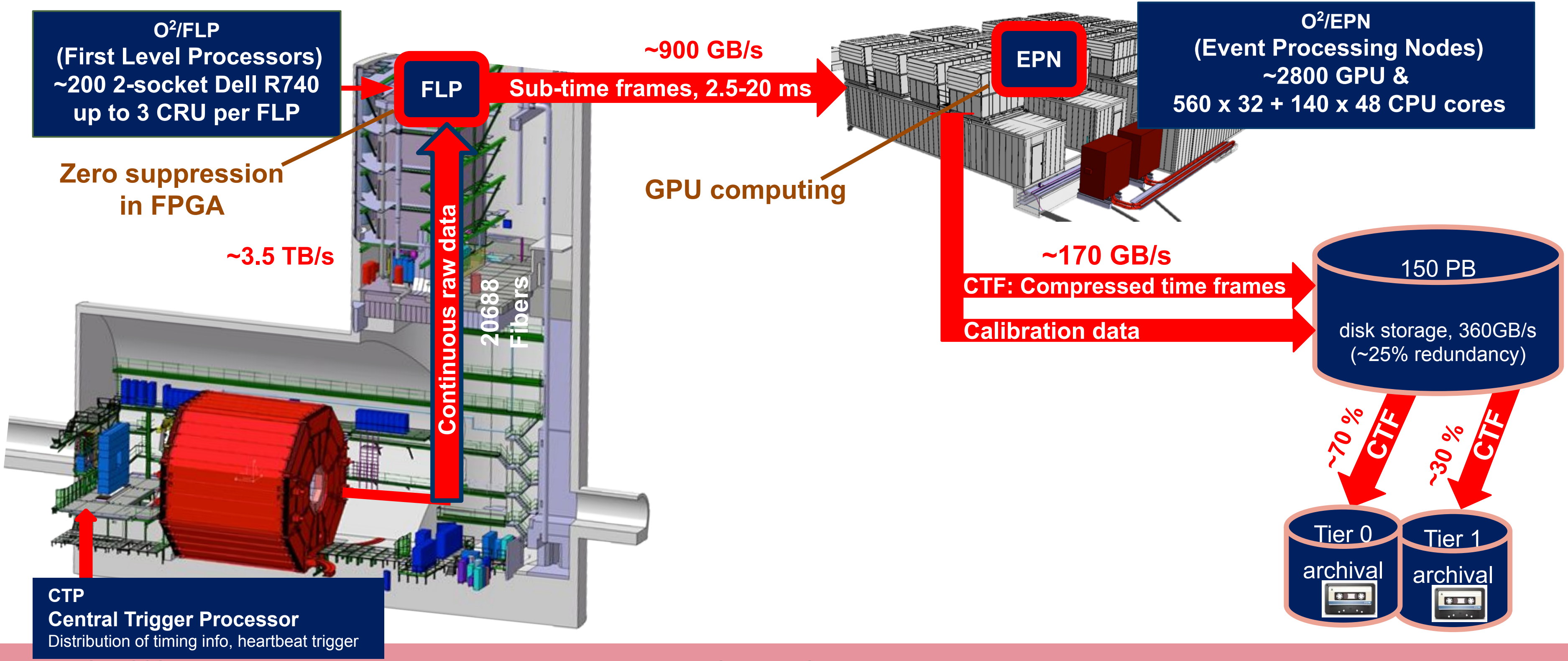
Porting on GPU: TPC Track-Model Clusters Decoding in ALICE

June 18th, 2024

Gabriele Cimador
(Università di Trieste and INFN Trieste)
for the ALICE collaboration

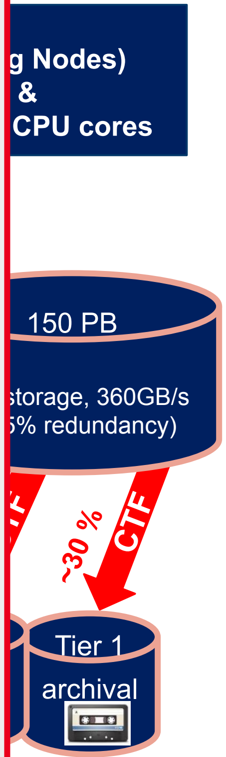
ALICE’s data flow - Synchronous Processing



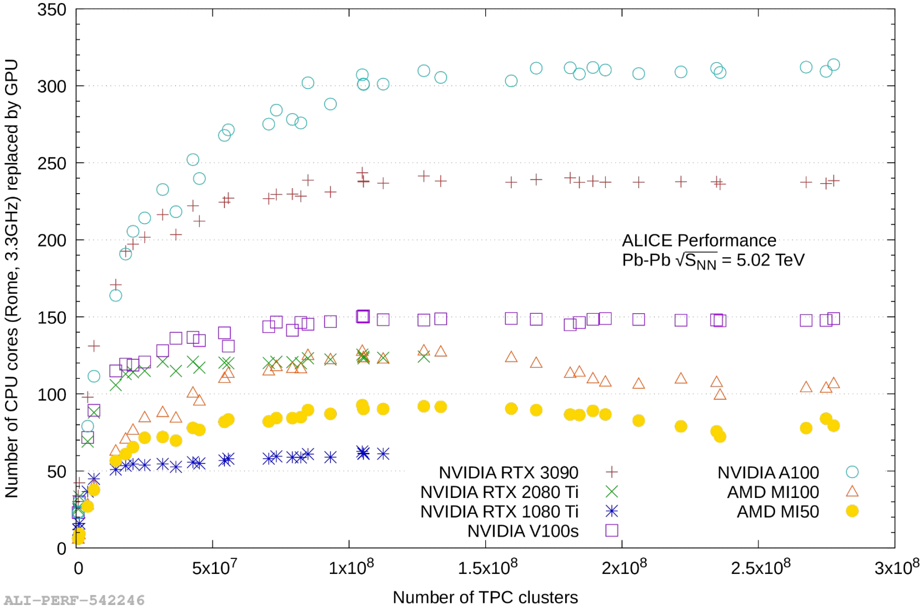
ALICE's data flow - Synchronous Processing

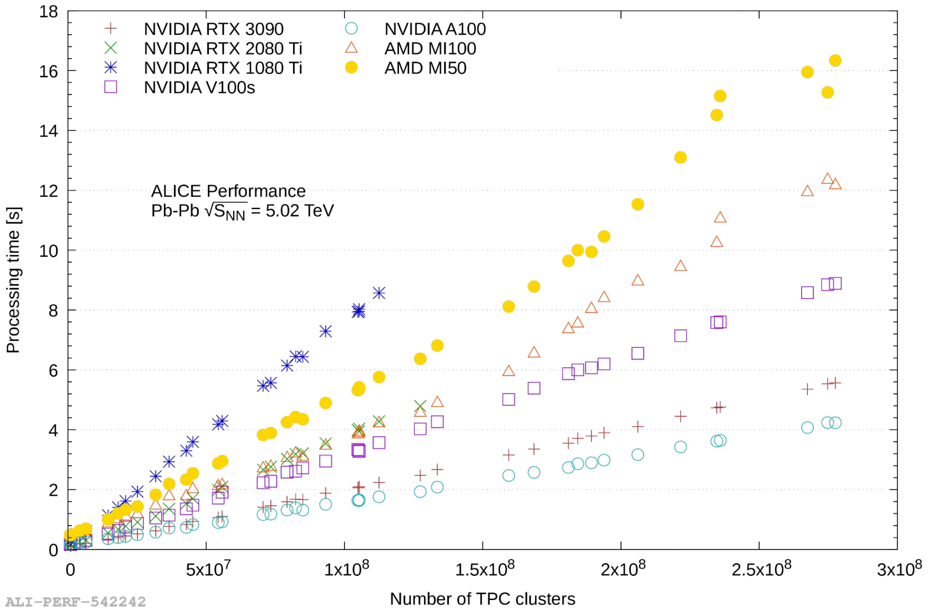
- TPC accounts for 90% of total data
- TPC compression needs **full online TPC track reconstruction for 100% of events**
- Hence **TPC processing takes 99%** of synchronous time
  - Clustering
  - TPC track reconstruction
  - Compression
- Moreover full barrel reconstruction for 1% of events for detector calibration

⟹ EPN tailored to run fastest TPC reconstruction possible **on GPUs**



TPC reconstruction on GPU





- GPUs can replace 50-300 CPU cores @ 3.3 GHz for ALICE TPC online processing (MI50 replaces ~80 CPU cores)
- Online processing time increases linearly with number of TPC clusters

EPN farm

Original EPN configuration

- 280 nodes
- 8xMI50 32 GB GPUs per node
- 2x32 physical cores AMD Rome 7452 CPUs
- 512 GB 3.2 GHz main memory

After 2022 Pb-Pb test extended the farm by 70 more nodes

- 2x48 physical cores
- 8xMI100 GPUs
- 1 TB main memory

CPU processing would need

> 2000 servers with related networking

$\Longrightarrow$ With GPUs, 350 servers with 30% processing margin

$\Longrightarrow$ GPUs most viable, feasible and easier to maintain solution within budget

ALICE data flow - Asynchronous Processing

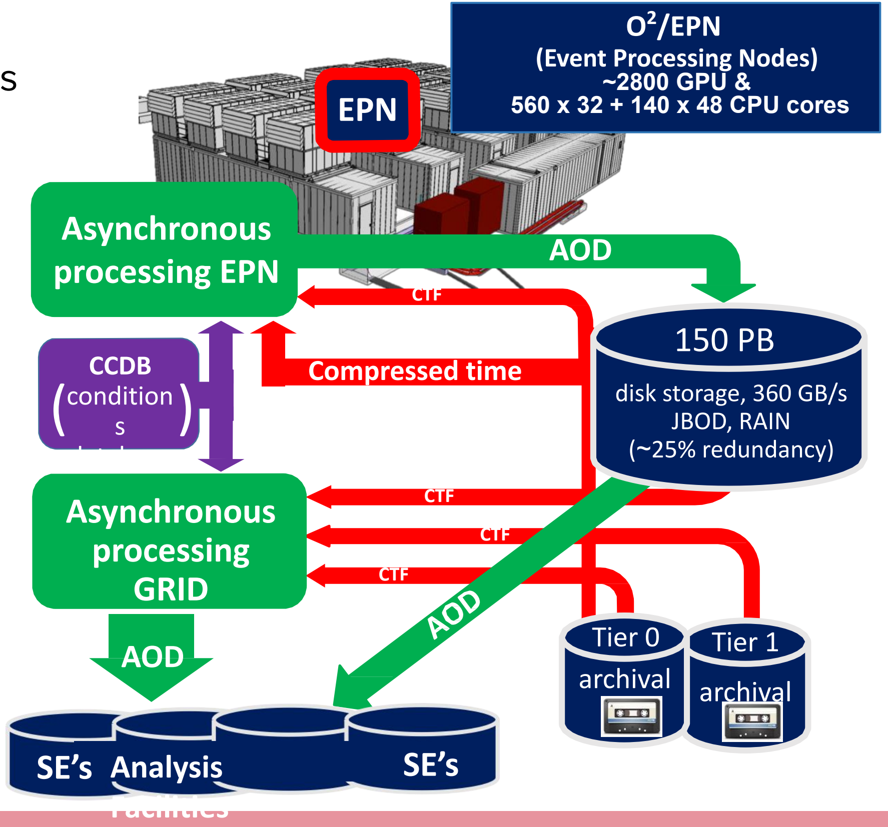
Full reconstruction with final calibration for all detectors

for all events

- Global tracks reconstructed
- Matching between detectors
- Primary and secondary vertices identification
- PID hypothesis

Computing model plans to run async processing

- ⅔ on GRID (CPUs only)
- ⅓ on EPN (CPUs + **GPUs**)



Sync processing vs async processing

Offline (async) processing:

- TPC: no clustering, no compression
- **Full** reconstruction for **all** collisions for **all** detectors

Different impact of TPC processing:

99.37 % online vs **61.41 % offline**

(linux CPU time)

ALICE employs single software framework for Online-Offline processing (O^2) $\Longrightarrow$ **Same algorithms for both reco**

- 60% of async reco already available on GPU (TPC)
- If we offload more async reco tasks to GPU:
  - Higher GPU load to become less CPU-bound
  - Consecutive tasks on GPU avoid memory transfer to CPU
  - Exploit as much as EPN resources as possible during async processing and reduce total processing time

Hence working on porting **full barrel tracking** on GPUs

Speedup in asynchronous reconstruction

- On EPNs 85% compute power is in the GPUs
- Reducing CPU time by 85% yields to 6.5x speedup
- Offloading more becomes GPU-bound ⟹ “speed of light” is 6.5x speedup
- At today 60% of async reco already on GPU ⟹ current **expected speedup of 2.5x**

- Optimistic scenario: 80% of async reco offloaded to GPU (full barrel tracking)
- **Expected speedup of 5x**

Asynchronous processing
650 kHz pp real data

Processing step	% of time
TPC Processing	61.41 %
ITS TPC Matching	6.13 %
MCH Clusterization	6.13 %
TPC Entropy Decoder	4.65 %
ITS Tracking	4.16 %
TOF Matching	4.12 %
TRD Tracking	3.95 %
Quality Control	4.00 %
Rest	5.22 %

Run on GPU in baseline scenario

Run on GPU in optimistic scenario

Asynchronous reconstruction benchmarks

- EPN nodes used as GRID nodes for async. reco.
- EPN divided in 2 NUMA domains, each with:
  - 64 virtual cores
  - 4 GPUs
- EPN split in different configurations for async. benchmark
  - 8 virtual cores and 16 virtual cores to test CPU performance
  - ⅛ of EPN: 16 virtual cores + 1 GPU
  - ½ of EPN: 64 virtual cores + 4 GPUs (entire NUMA region)

Configuration (2022 pp, 650 kHz)	**Time per TF (11 ms, 1 instance)**	**Time per TF (11 ms, full server)**
CPU 8 virtual cores	76.91 s	4.81 s
CPU 16 virtual cores	34.18 s	**4.27 s**
1 GPU + 16 CPU virtual cores	14.60 s	1.83 s
1 NUMA domain (4 GPUs + 64 virtual cores)	3.5 s	**1.70 s**

Asynchronous reconstruction benchmarks

- EPN nodes used as GRID nodes for async. reco.
- EPN divided in 2 NUMA domai
  - 64 virtual cores
  - 4 GPUs
- EPN split in different configurat
  - 8 virtual cores and 16 virtual cores t
  - ⅛ of EPN: 16 virtual cores + 1 GPU
  - ½ of EPN: 64 virtual cores + 4 GPUs (entire NUMA region)

Factor 2.51
Proves expected speedup of 2.5

Configuration (2022 pp, 650 kHz)	Time per TF (11 ms, 1 instance)	Time per TF (11 ms, full server)
CPU 8 virtual cores	76.91 s	4.81 s
CPU 16 virtual cores	34.18 s	**4.27 s**
1 GPU + 16 CPU virtual cores	14.60 s	1.83 s
1 NUMA domain (4 GPUs + 64 virtual cores)	3.5 s	**1.70 s**

TPC track-model coding / decoding

- TPC data compression via entropy encoding (Asymmetric Numeral Systems encoders family)
- Less entropy in data means a better compression factor
- Hence several steps in TPC compression aims at reducing entropy
- Part of cluster entropy reduction is the **track-model encoding**:
  1. Coordinates of clusters attached to tracks stored as residuals to extrapolated track model
  2. Unattached clusters sorted by coordinates, values saved as differences between consecutive clusters

Thus async reco needs to decode cluster coordinates for reconstruction:

- TPC track-model decoding implemented on CPU (baseline scenario)
- Offloaded **track-model decoding to GPU** as part of the optimistic scenario

GPU track-model decoding details

- Input: structures of arrays containing residuals and other properties of clusters
- Output: contiguous array of TPC cluster objects
  - TPC divided into rows of readout pads (152 per sector, 36 TPC total sectors)
  - Output array logically divided into 5472 chunks, each chunk contains clusters from same row
  - Not necessary to sort clusters inside chunk, but need sorting between chunks e.g. first row of first sector goes first
- Problem: cannot know a priori how many attached clusters per row (need to propagate track along TPC volume)

GPU solution:

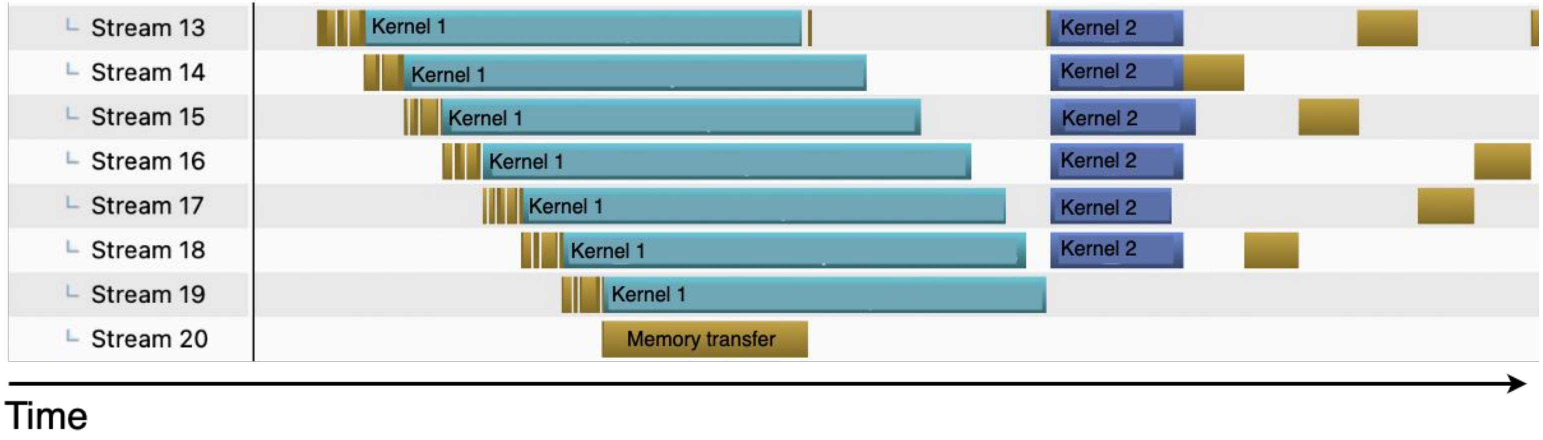
1. First GPU kernel decodes attached clusters
   a. Each GPU thread takes one track, propagates along TPC volume and decodes coordinates
   b. Cluster object stored in distinct temporary buffer for every row (5472 tmp buffers)
2. Second GPU kernel decodes unattached clusters
   a. Each GPU thread takes one row, copies related tmp buffer in output buffer
   b. Decodes unattached clusters of related row directly in final buffer

Improvements

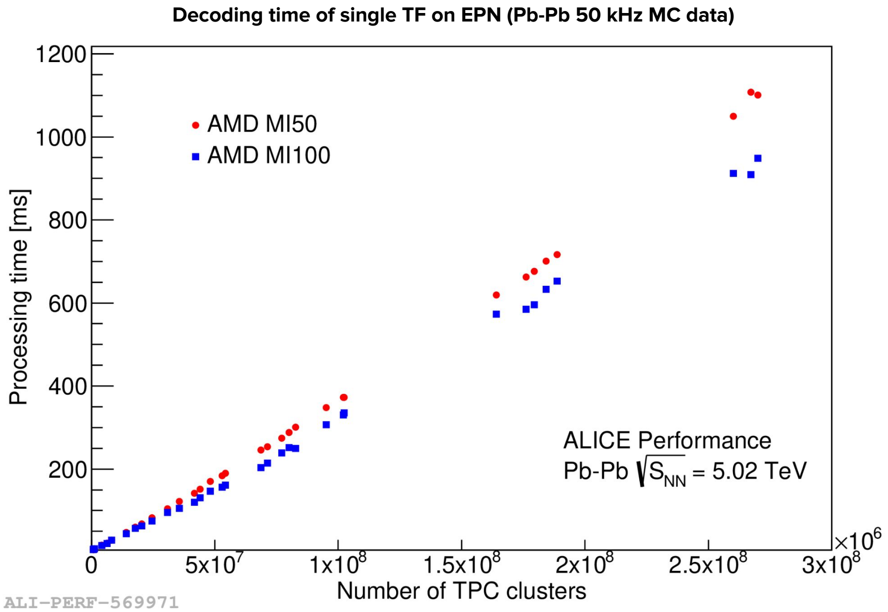
Made efforts to improve GPU solution:

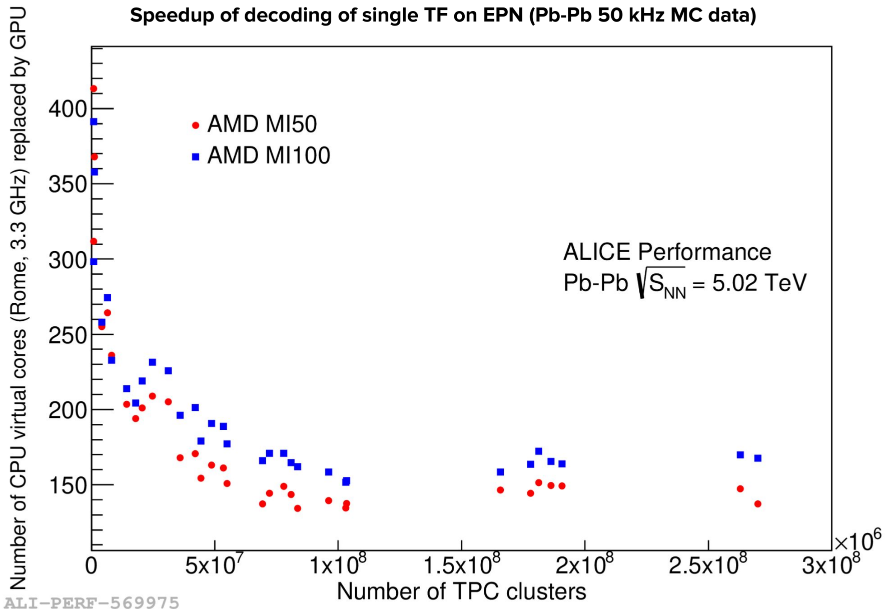
- Structure of arrays demands serial offset precomputation which suits CPU computing better
- Hide host-device memory transfer latency
  - Input divided into equal chunks and processed in different streams by first kernel
  - Second kernel also processed in different streams and transfer to host divided into equal chunks

- Memory transfer
- Kernel 1
- Kernel 2



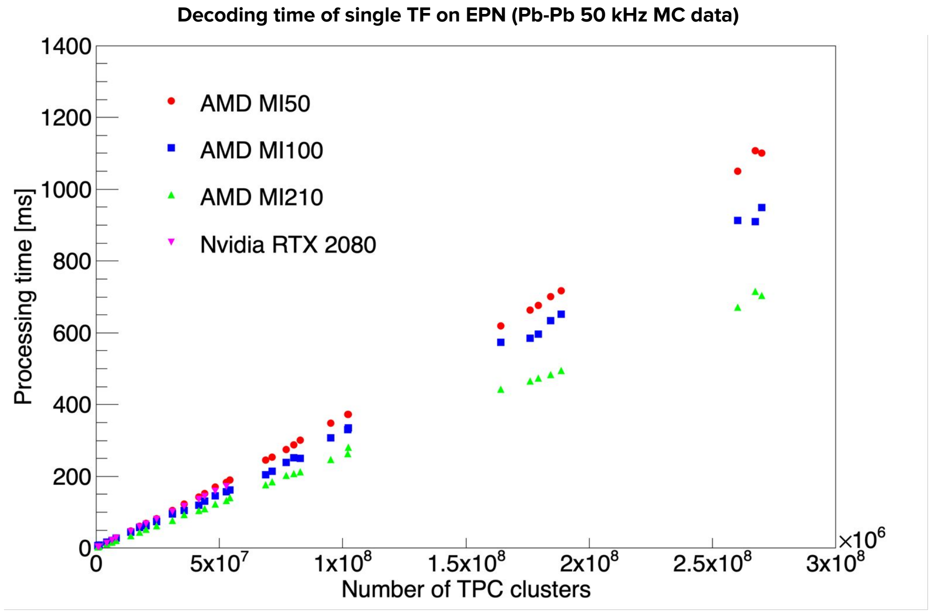
TPC track-model decoding performance on GPU





- No strong superlinear effect on GPU decoding
- Single GPU (MI50-MI100) replaces 140-170 CPU cores for decoding

TPC track-model decoding performance on GPU



AMD Instinct Accelerators				
	MI250	MI210	MI100	MI50
Compute Units	[illegible]	104	120	60
Matrix Cores	[illegible]	416	480	N/A
Boost Clock	[illegible]	1700MHz	1502MHz	1725MHz
FP64 Vector	[illegible]	22.6 TFLOPS	11.5 TFLOPS	6.6 TFLOPS
FP32 Vector	[illegible]	22.6 TFLOPS	23.1 TFLOPS	13.3 TFLOPS
FP64 Matrix	[illegible]	45.3 TFLOPS	11.5 TFLOPS	6.6 TFLOPS
FP32 Matrix	[illegible]	45.3 TFLOPS	46.1 TFLOPS	13.3 TFLOPS
FP16 Matrix	[illegible]	181 TFLOPS	184.6 TFLOPS	26.5 TFLOPS
INT8 Matrix	[illegible]	181 TOPS	184.6 TOPS	N/A
Memory Clock	[illegible]	3.2 Gbps HBM2E	2.4 Gbps HBM2	2.0 Gbps GDDR6
Memory Bus Width	[illegible]	4096-bit	4096-bit	4096-bit
Memory Bandwidth	[illegible]	1.6TBps	1.23TBps	1.02TBps
VRAM	128GB	64GB	32GB	[illegible] 32 GB

- Nvidia RTX 2080 with 8 GB of VRAM is limited to to timeframes with a small number of clusters
- MI210 (CNAF) proves to be significantly more performant

Impact on async reco performance

On EPN run two async reco, one per NUMA domain:

- Single GPU decoding can replace ~ 150 virtual cores
- Four GPUs per NUMA domain vs 64 virtual cores

TPC track-model decoding is only a small part of async reco

Async reconstruction with GPU decoding vs CPU decoding:

- ~ 2,8% faster for 2023 pp data
- ~ 1,2% faster for 2023 Pb-Pb data

ITS Clustering on GPU (Leonardo Cristella @ INFN BA)

Investigating feasibility of porting (part of) ITS clustering to **alpaka**

- alpaka: Abstraction Library for Parallel Kernel Acceleration
- Header-only C++17 abstraction library for accelerator development
- Platform independent, aims to provide performance portability across accelerators through the abstraction of the underlying levels of parallelism
- Supports the concurrent use of multiple devices such as the hosts CPU as well as attached accelerators (for instance Nvidia/AMD GPUs)
- Only **one implementation** of the user kernel is required
  - no need to write special CUDA, OpenMP or custom threading code

Summary

- GPUs proven fundamental for ALICE in Run 3
    - GPUs allow to collect 50 kHz Pb-Pb collisions in continuous readout mode within budget
    - Synchronous reconstruction 99% of processing time on GPU
    - Asynchronous reconstruction 60% on GPU and more to come (when running on EPN)
- When run on EPN, asynchronous reconstruction is 2.51x faster thanks to GPUs
    - **Asynchronous reco productions for physics** run on GRID (CPU) and on EPN (CPU + GPU) since January 2023
    - Successfully decoded TPC cluster data on GPUs
    - PR merged into the official ALICE code
    - Decoding on GPU is a step ahead towards the optimistic scenario
    - Working to reach expected speedup of 5x