Usage of GPUs for online and offline reconstruction in ALICE in Run 3

David Rohr for the ALICE Collaboration

ICHEP 2024

19.7.2024

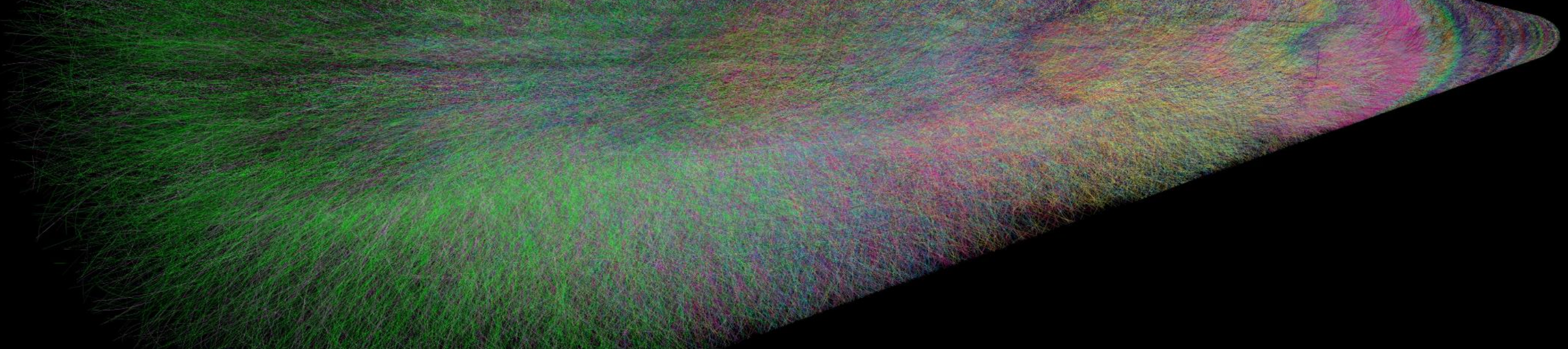
drohr@cern.ch

ALICE in Run 3

Targeting to record large minimum bias sample.

- All collisions stored for main detectors → no trigger
- Continuous readout → data in drift detectors overlap
- Recording time frames of continuous data, instead of events
- 100x more collisions, much more data
- Cannot store all raw data → online compression

→ Use GPUs to speed up online (and offline) processing



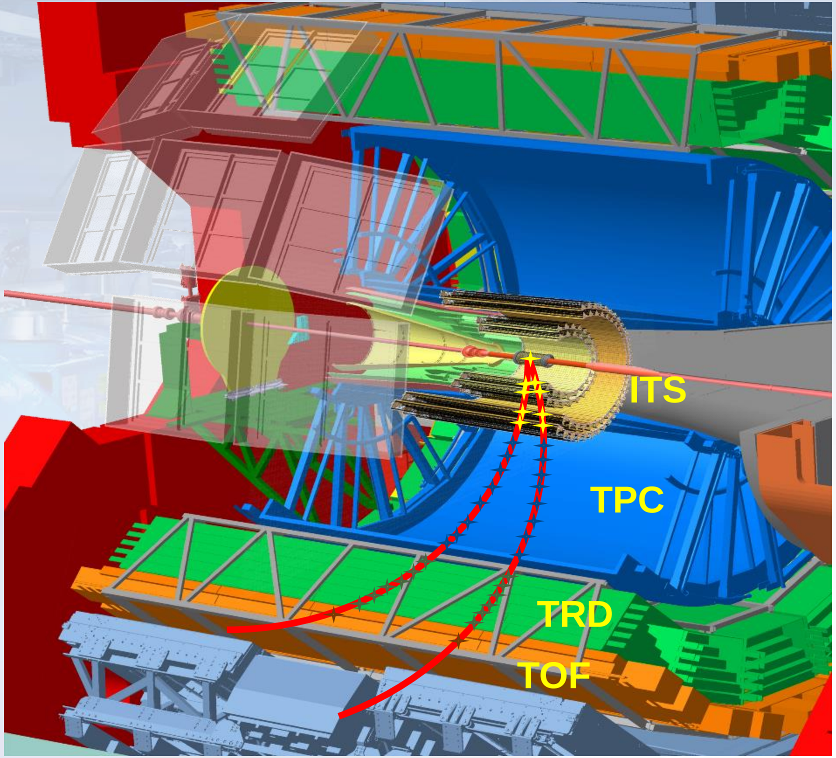
- Overlapping events in TPC with realistic bunch structure @ 50 kHz Pb-Pb.
- Timeframe of 2 ms shown (will be 10 – 20 ms in production).
- Tracks of different collisions shown in different colors.

David Rohr, drohr@cern.ch

The ALICE detector in Run 3

- **ALICE uses mainly 3 detectors for barrel tracking: ITS, TPC, TRD + (TOF)**
  - **7 layers ITS** (Inner Tracking System – silicon tracker)
  - **152 pad rows TPC** (Time Projection Chamber)
  - **6 layers TRD** (Transition Radiation Detector)
  - **1 layer TOF** (Time Of Flight Detector)

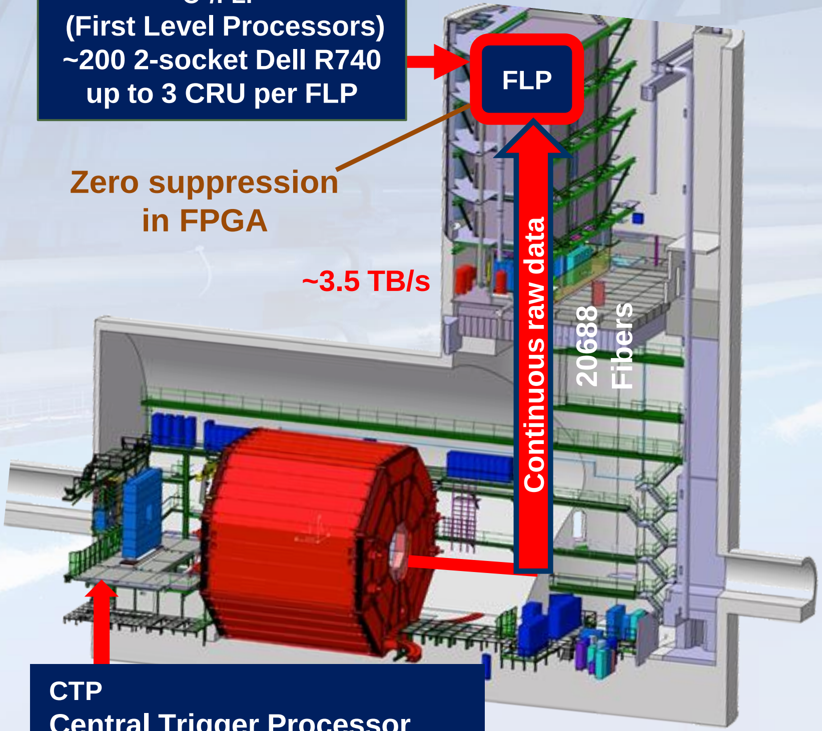
- **ALICE performs continuous readout.**
- **Native data unit is a time frame: all data from a configurable period of data up to 256 LHC orbits.**
  - Current default since 2023 is **~2.8 ms** (32 LHC orbits)



ALICE Raw Data Flow in Run 3

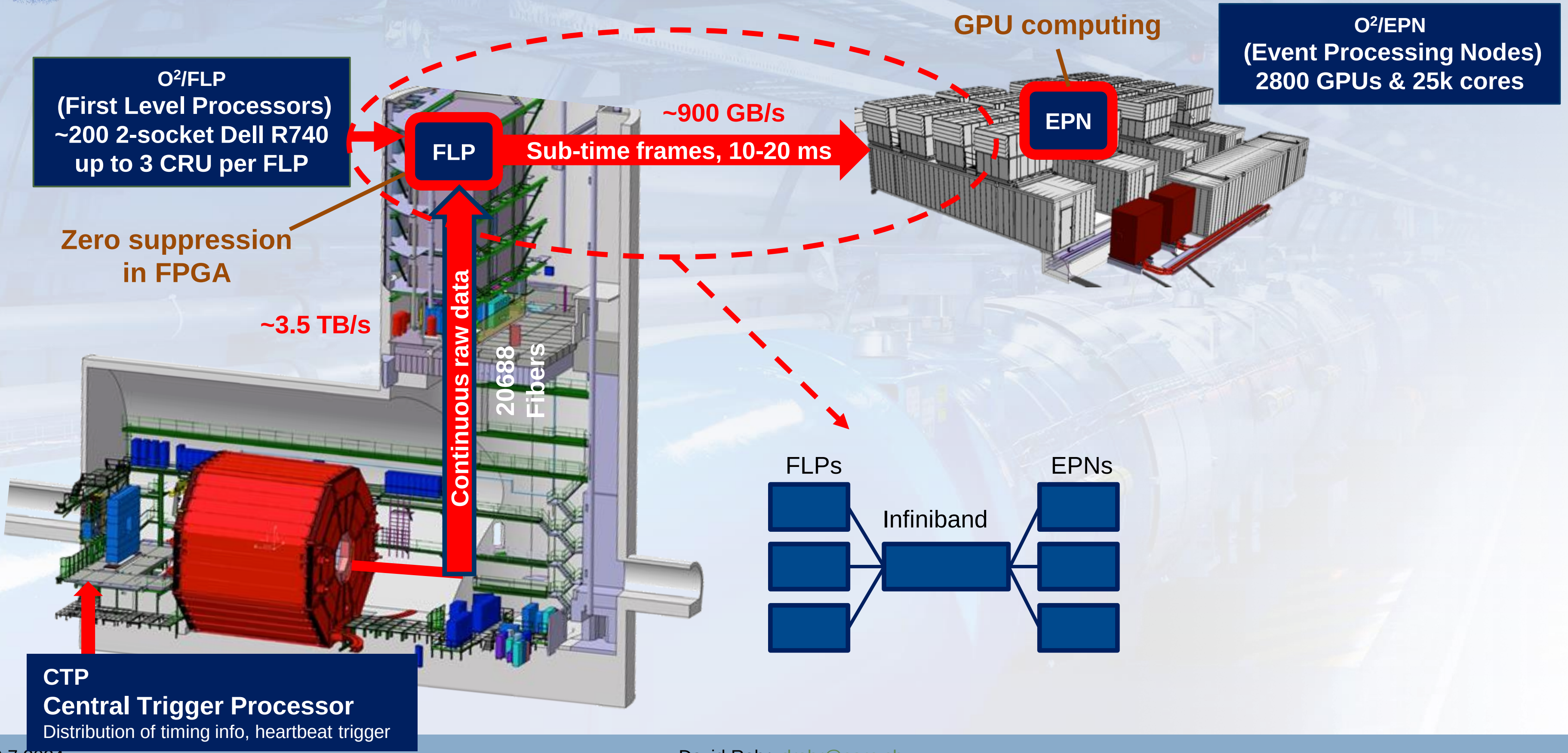
O²/FLP
(First Level Processors)
~200 2-socket Dell R740
up to 3 CRU per FLP

Zero suppression in FPGA

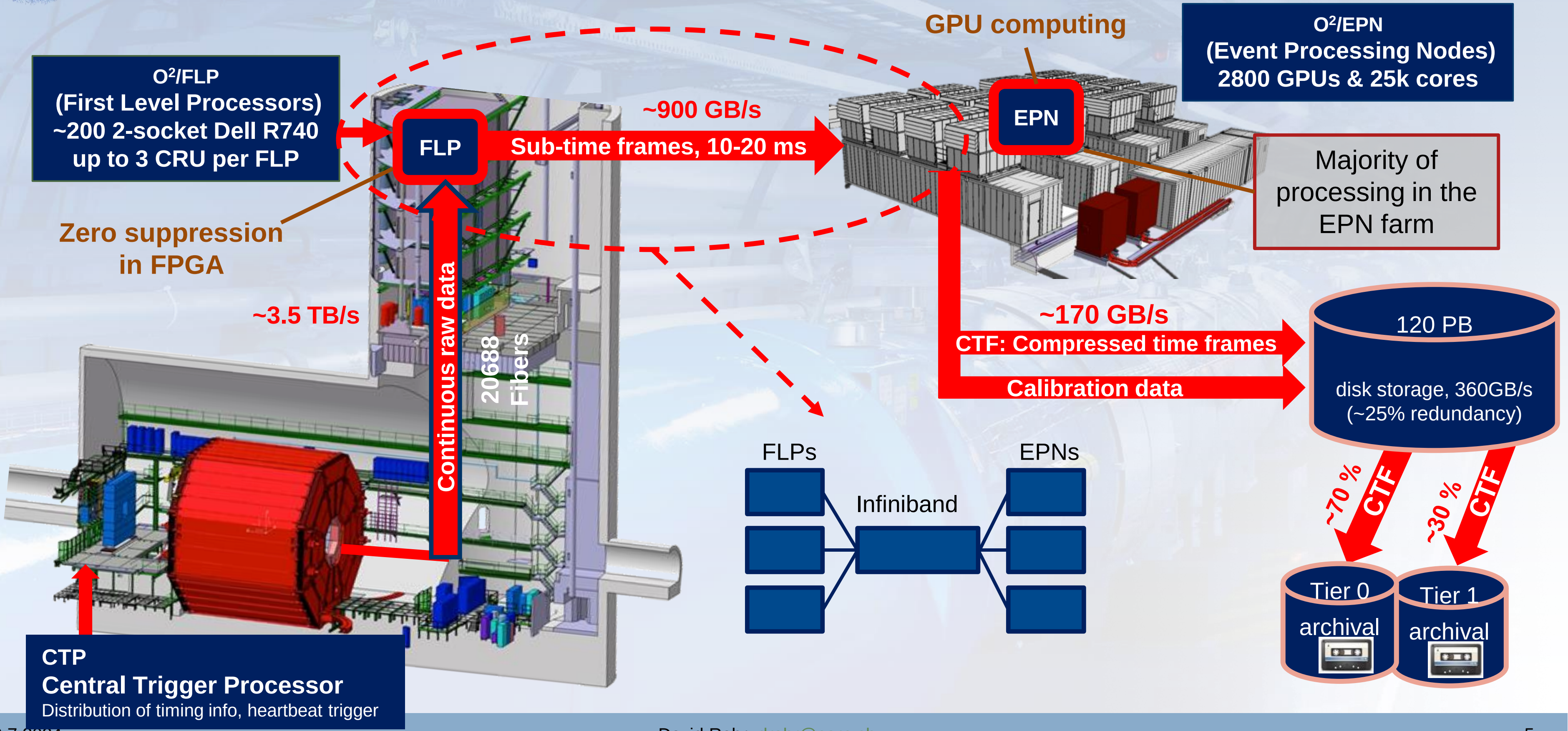


CTP
Central Trigger Processor
Distribution of timing info, heartbeat trigger

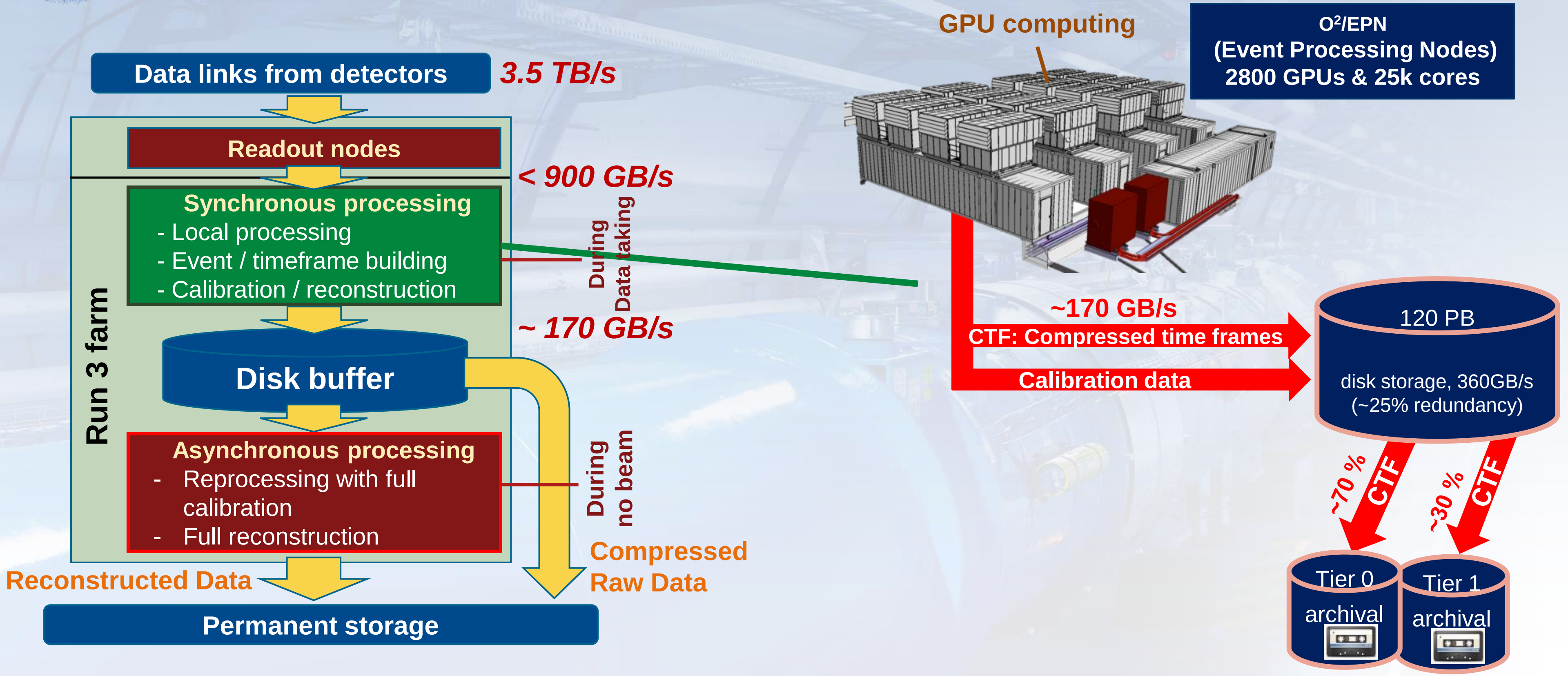
ALICE Raw Data Flow in Run 3



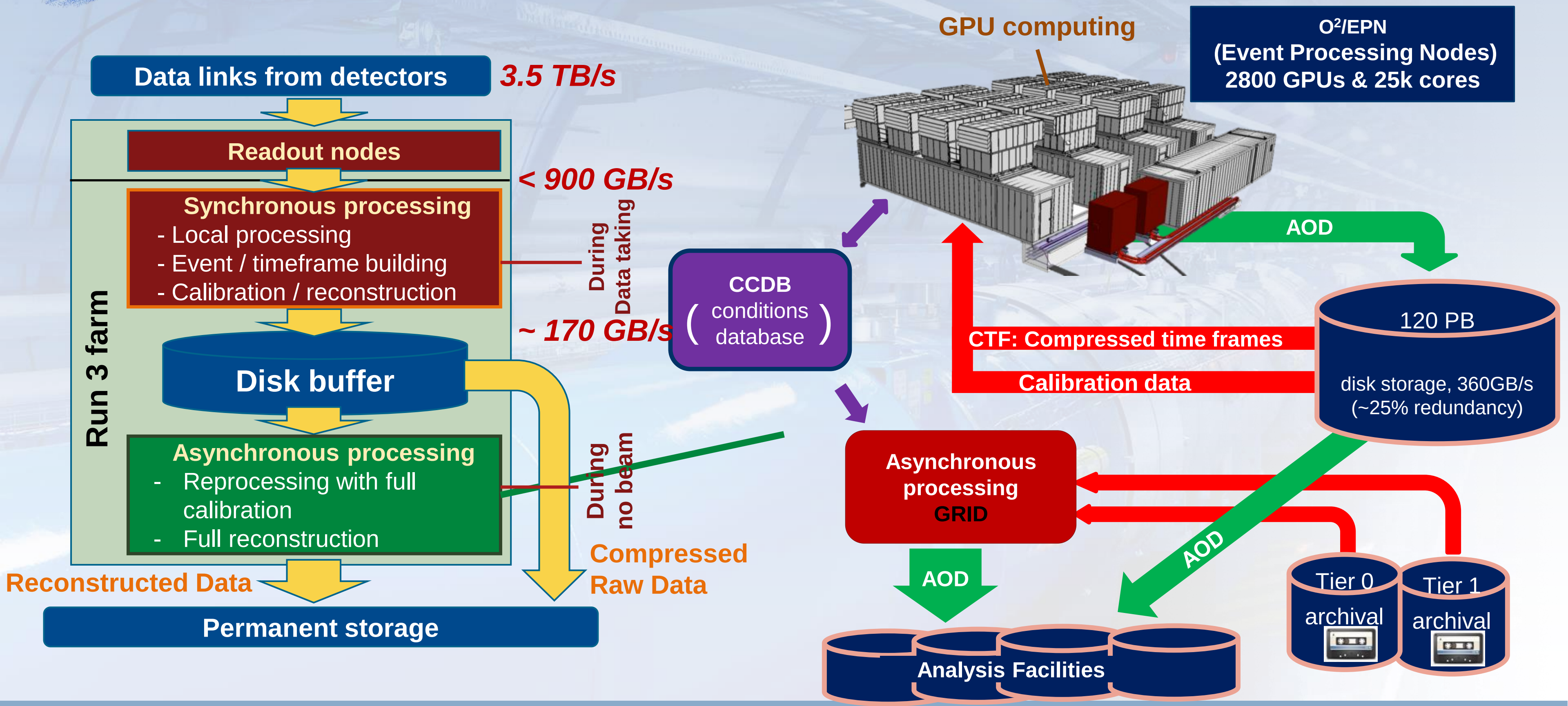
ALICE Raw Data Flow in Run 3



Synchronous (online) and Asynchronous (offline) Processing



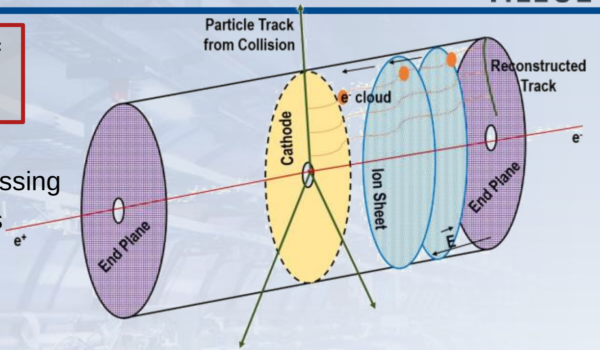
Synchronous (online) and Asynchronous (offline) Processing



O² Processing steps

- **Online processing:**
  - Extract information for **detector calibration**:
    - Previously performed in 2 offline passes over the data after the data taking
    - Run 3 avoids / reduces extra passes over the data but extracts all information in the sync. processing
    - An intermediate step between sync. and async. processing produces the final calibration objects
    - The most complicated calibration is the correction for the TPC space charge distortions

Needs tracking of 1% of tracks

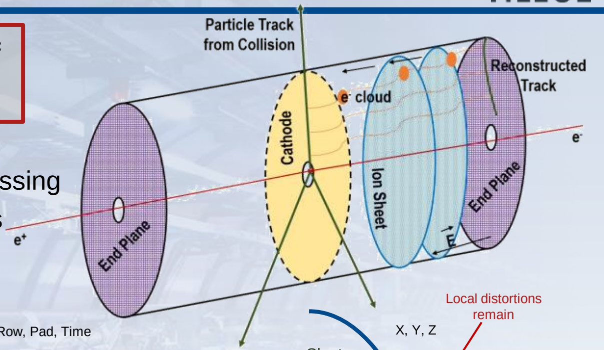


O² Processing steps

- **Online processing:**
  - Extract information for **detector calibration**:
    - Previously performed in 2 offline passes over the data after the data taking
    - Run 3 avoids / reduces extra passes over the data but extracts all information in the sync. processing
    - An intermediate step between sync. and async. processing produces the final calibration objects
    - The most complicated calibration is the correction for the TPC space charge distortions
  - **Data compression**:
    - TPC is the largest contributor of raw data, and we employ sophisticated algorithms like storing space point coordinates as residuals to tracks to reduce the entropy and remove hits not attached to physics tracks
    - We use ANS entropy encoding for all detectors

Needs tracking of 1% of tracks

Needs 100% TPC tracking

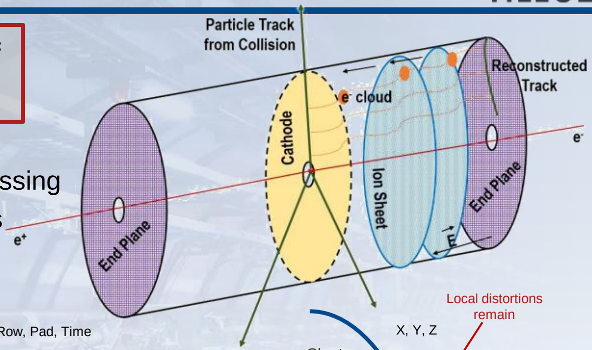


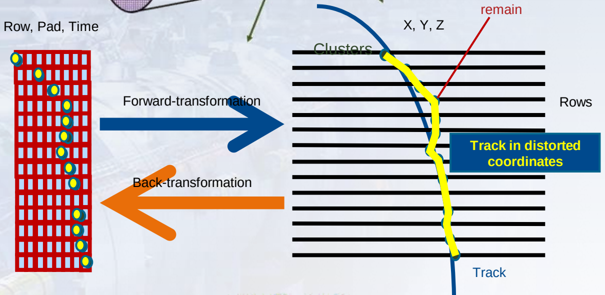
O² Processing steps

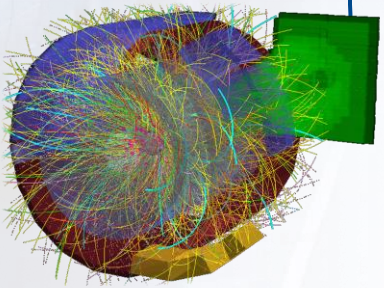
- **Online processing:**
  - Extract information for **detector calibration**:
    - – Previously performed in 2 offline passes over the data after the data taking
    - – Run 3 avoids / reduces extra passes over the data but extracts all information in the sync. processing
    - – An intermediate step between sync. and async. processing produces the final calibration objects
    - – The most complicated calibration is the correction for the TPC space charge distortions
  - **Data compression**:
    - – TPC is the largest contributor of raw data, and we employ sophisticated algorithms like storing space point coordinates as residuals to tracks to reduce the entropy and remove hits not attached to physics tracks
    - – We use ANS entropy encoding for all detectors
  - **Event reconstruction** (tracking, etc.):
    - – Required for calibration, compression, and online quality control
    - – Need **full TPC tracking** for data compression
    - – Need tracking in all detectors for ~1% of the tracks for calibration
    - → **TPC tracking dominant part**, **rest** almost negligible (**< 5%**)

Needs tracking of 1% of tracks

Needs 100% TPC tracking





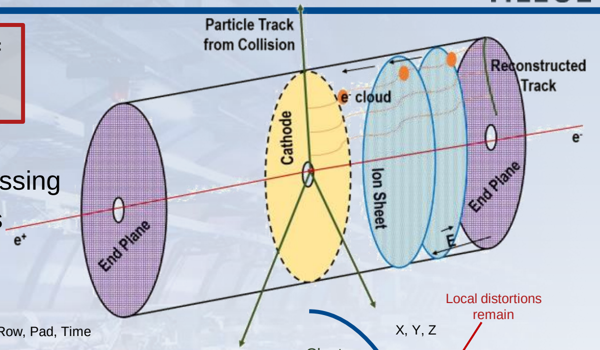


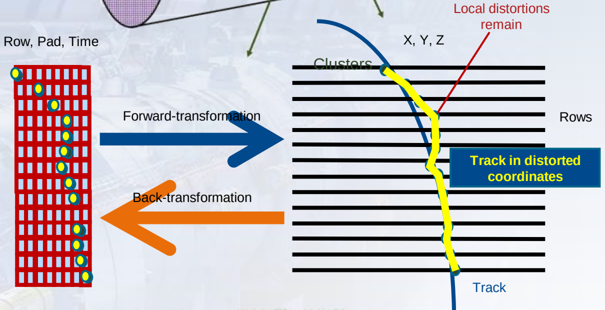
O² Processing steps

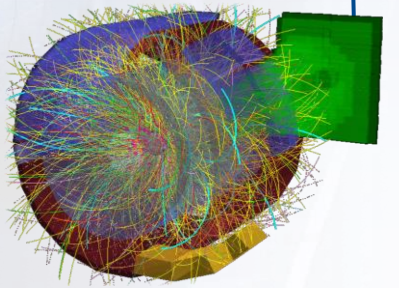
- **Online processing:**
  - Extract information for **detector calibration**:
    - Previously performed in 2 offline passes over the data after the data taking
    - Run 3 avoids / reduces extra passes over the data but extracts all information in the sync. processing
    - An intermediate step between sync. and async. processing produces the final calibration objects
    - The most complicated calibration is the correction for the TPC space charge distortions
  - **Data compression**:
    - TPC is the largest contributor of raw data, and we employ sophisticated algorithms like storing space point coordinates as residuals to tracks to reduce the entropy and remove hits not attached to physics tracks
    - We use ANS entropy encoding for all detectors
  - **Event reconstruction** (tracking, etc.):
    - Required for calibration, compression, and online quality control
    - Need **full TPC tracking** for data compression
    - Need tracking in all detectors for ~1% of the tracks for calibration
    - → **TPC tracking dominant part**, **rest** almost negligible (**< 5%**)

Needs tracking of 1% of tracks

Needs 100% TPC tracking







- **Offline processing (what we called offline before):**
  - **Full reconstruction**, **full calibration**, **all detectors**
  - TPC part faster than in online processing (less hits, no clustering, no compression)
  - → **Different relative importance of GPU** / **CPU** algorithms compared to online processing

GPU usage in ALICE in the past

- **ALICE has a long history of GPU usage in the online systems, and since 2023 also for offline:**

2010
64 * NVIDIA GTX 480 **in Run 1**
Online TPC tracking

2015
180 * AMD S9000 in **Run 2**
Online TPC tracking

Today
~3000 * AMD MI50/MI100 in **Run 3**
Online and Offline barrel tracking

Where to use GPUs?

- Could use GPUs in online reconstruction, offline reconstruction, simulation, analysis, ...
- **Online computing** constrained to **on-site farm**: fully under **our control**, **GPUs** can provide the **required compute power**.
- **Main purpose** of **GPU** in **online farm**: **Keep step with online processing rate**.
- Everything else is nice but secondary – and also general / GRID computing more heterogeneous.

Online reconstruction
(50 kHz Pb-Pb, MC data, no QA / calib)

Processing step	% of time
TPC Processing (Tracking, Clustering, Compression)	99.37 %
EMCAL Processing	0.20 %
ITS Processing (Clustering + Tracking)	0.10 %
TPC Entropy Encoder	0.10 %
ITS-TPC Matching	0.09 %
MFT Processing	0.02 %
TOF Processing	0.01 %
TOF Global Matching	0.01 %
PHOS / CPV Entropy Coder	0.01 %
ITS Entropy Coder	0.01 %
Rest	0.08 %

Offline processing
(650 kHz pp, 2022, no Calorimeters)

Processing step	% of time
TPC Processing (Tracking)	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 %
MCH Tracking	2.02 %
AOD Production	0.88 %
Quality Control	4.00 %
Rest	2.32 %

Offline processing
(47 kHz Pb-Pb, 2024)

Processing step	% of time
TPC Processing (Tracking)	52.39 %
ITS Tracking	12.65 %
Secondary Vertexing	8.97 %
MCH	5.28 %
TRD Tracking	4.39 %
TOF Matching	2.85 %
ITS TPC Matching	2.64 %
Entropy Decoding	2.63 %
AOD Production	1.72 %
Quality Control	1.64 %
Rest	4.84 %

Relative CPU time (linux cputime) with full processing on CPU

GPU usage for online reconstruction

- Could use GPUs in online reconstruction, offline reconstruction, simulation, analysis, ...
- **Online computing** constrained to **on-site farm**: fully under **our control**, **GPUs** can provide the **required compute power**.
- **Main purpose** of **GPU** in **online farm**: **Keep step with online processing rate**
- Everything else is nice but secondary – and also general ... ous.

Online processing fully dominated by TPC

Online reconstruction
(50 kHz Pb-Pb, MC data, no QA / calib)

Processing step	% of time
TPC Processing (Tracking, Clustering, Compression)	99.37 %
EMCAL Processing	0.20 %
ITS Processing (Clustering + Tracking)	0.10 %
TPC Entropy Encoder	0.10 %
ITS-TPC Matching	0.09 %
MFT Processing	0.02 %
TOF Processing	0.01 %
TOF Global Matching	0.01 %
PHOS / CPV Entropy Coder	0.01 %
ITS Entropy Coder	0.01 %
Rest	0.08 %

Running on GPU in baseline scenario
1st GPU offload phase – mandatory for online

(650 kHz pp, 2022, no Calorimeters)

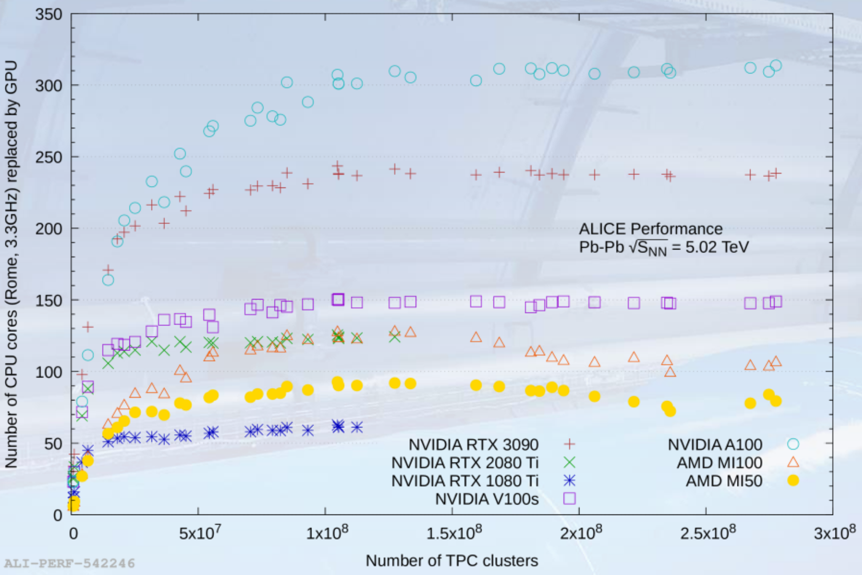
Offline processing
(47 kHz Pb-Pb, 2024)

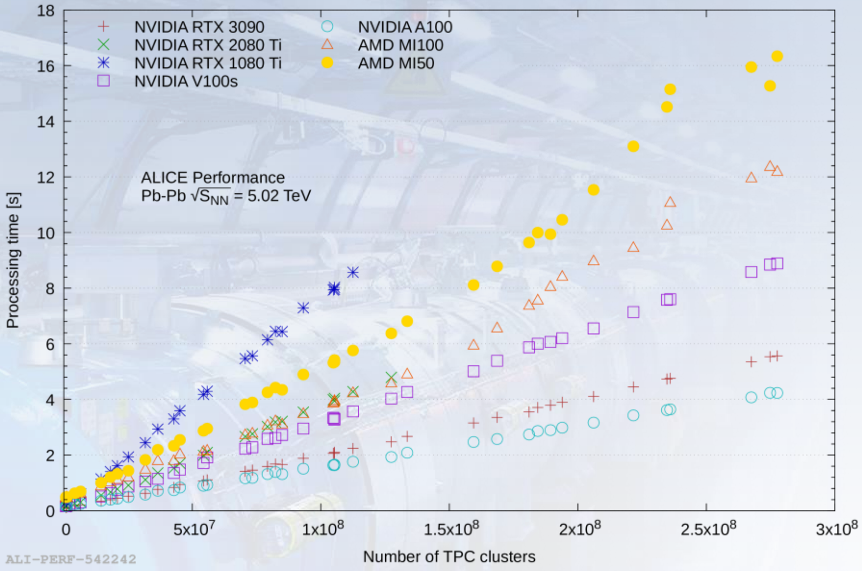
EPN farm designed for online processing

- Optimized for 50 kHz Pb-Pb (peak rate).
- Known from Run 1 & 2 GPU experience that TPC tracking is well suited to run on GPUs.
- Need only enough CPU power and memory buffers to keep GPUs busy.
- Exact total % of TPC depends on how much CPU is used for network IO, event building, quality control, etc., but in any case 95 – 99%.
  - CPUs represent at least 10% of the node's compute capacity, thus absolutely no reason to offload >90%.
  - Makes code more complicated for no benefit.
- GPUs should be at high but not full load, aimed for 30% GPU compute margin.
  - Nodes will be broken, unforeseen things happen, ...

Online processing performance

- Performance of Alice O2 software on different GPU models and compared to CPU.





- **GPU speedup fully linear, no superlinear complexity.**
- **ALICE uses 2240 MI50 and 560 MI100 GPUs in the online farm.**
- **MI50 GPU replaces ~80 AMD Rome CPU cores in online reconstruction.**
  - **~55 CPU cores** in **offline** reconstruction (different algorithm mix).
- **MI100 GPUs ~40% faster.**

Without GPUs, more than 3000 64-core servers would be needed for online processing!
GPUs mandatory for ALICE in Run 3.

Implementation details

- **Multiple GPUs in a server minimize the cost.**
  - Less servers, less network.
  - **Synergies** of using the **same CPU components** for multiple GPUs, same for memory.
- **Splitting the node into 2 NUMA domains minimizes inter-socket communication**
  → **2 virtual EPNs**.
  - Still only **1 HCA** for the input → writing to shared memory segment in **interleaved memory**.
- **GPUs are processing individual time frames → no inter-GPU communication.**
  - Host processes can drive 1 GPU each, or run CPU only tasks.
- **GPUs can be shared between algorithms.**
  - With **memory reuse** if within the same process.
  - With separate memory in case of multiple processes (Not done at the moment).
- **Benchmarked with MC data: For 100% utilization of 8 GPUs** (AMD MI50)**, we need:**
  - ~**50 CPU cores**, ~**400 GB** of memory, **30 GB/s** network input speed, GPU PCIe negligible.

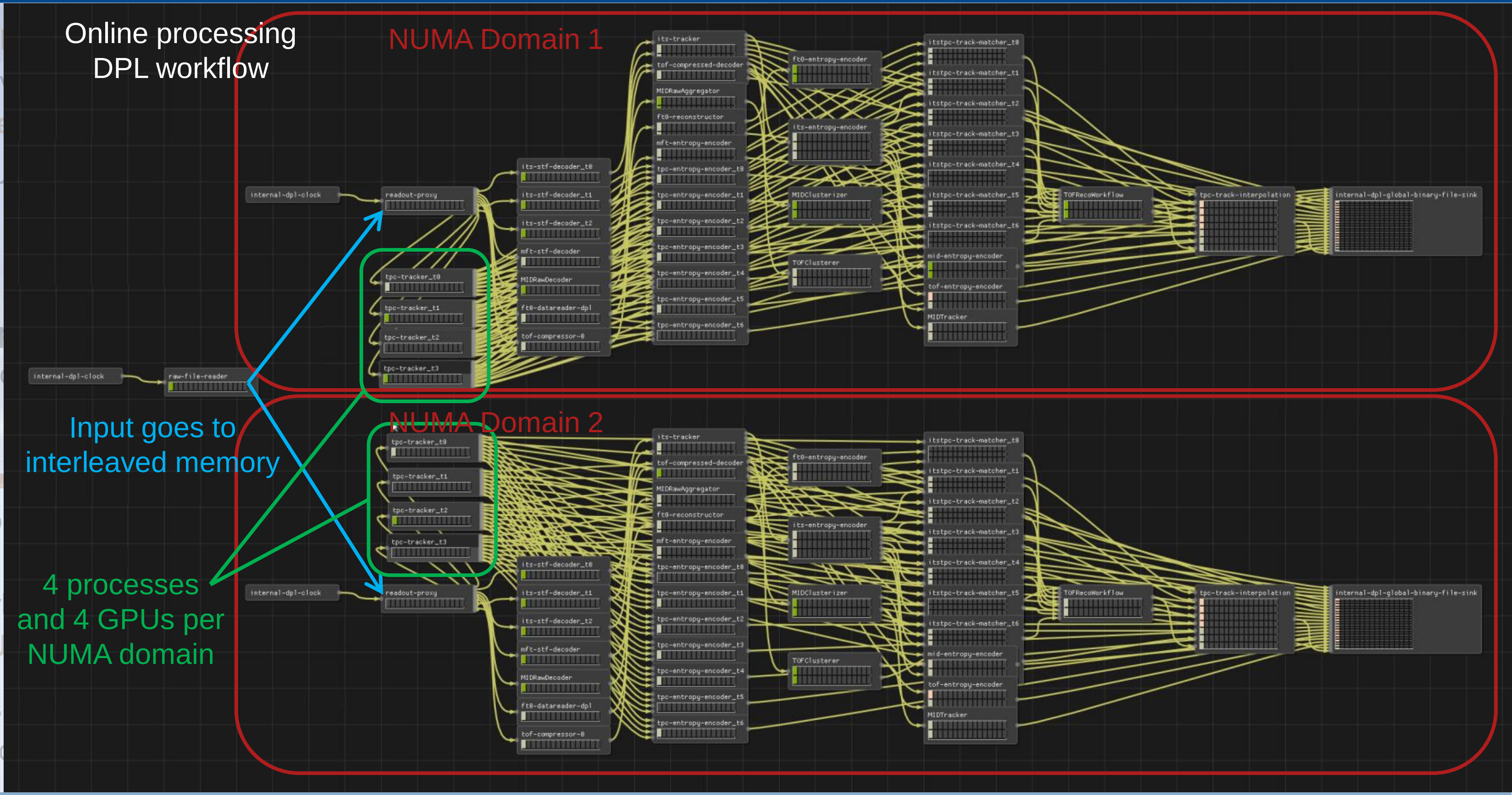
Implementation details

- **Multiple GPUs in a server minimize the cost.**
  - Less servers, less network.
  - **Synergies** of using the **same CPU components** for multiple GPUs, same for memory.
- **Splitting the node into 2 NUMA domains minimizes inter-socket communication**
  → **2 virtual EPNs**.
  - Still only **1 HCA** for the input → writing to shared memory segment in **interleaved memory**.
- **GPUs are processing individual time frames → no inter-GPU communication.**
  - Host processes can drive 1 GPU each, or run CPU only tasks.
- **GPUs can be shared between algorithms.**
  - With **memory reuse** if within the same process.
  - With separate memory in case of multiple processes (Not done at the moment).
- **Benchmarked with MC data: For 100% utilization of 8 GPUs** (AMD MI50)**, we need:**
  - ~**50 CPU cores**, ~**400 GB** of memory, **30 GB/s** network input speed, GPU PCIe negligible.
- **Selected server:**
  - Supermicro AS-4124GS-TNR, **8 * MI50** GPU, **2 * 32 core** AMD Rome 7452 CPU (2.35 GHz), **512 GB RAM** (16 * 32GB)
  - Infiniband HDR / HDR100 network.

Implementation details

For details on DPL workflows, see talk about SW status.

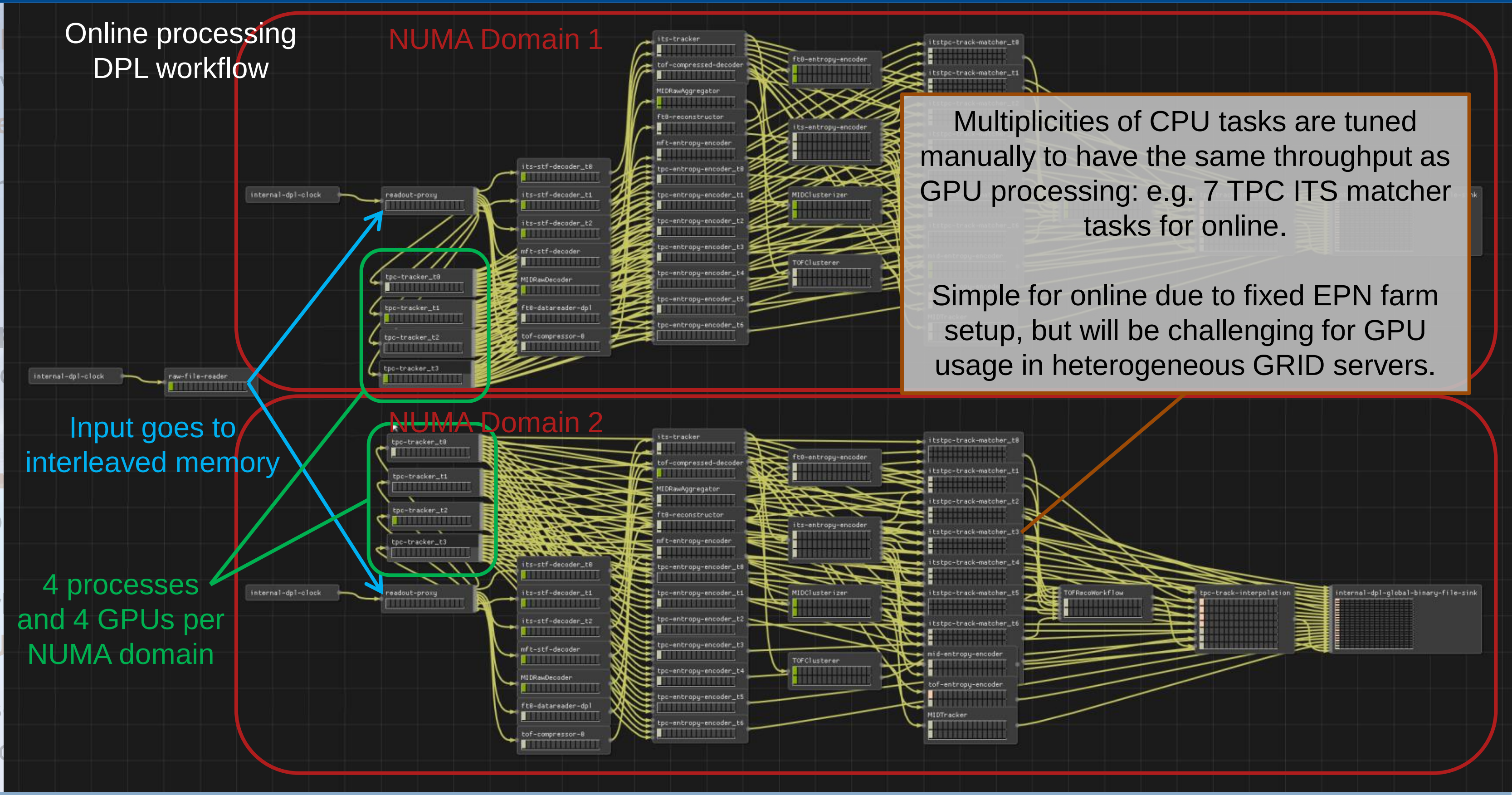
- Multiple GI
  - Less ser
  - Synergie
- Splitting th
  → 2 virtual
  - Still only
- GPUs are
  - Host pro
- GPUs can
  - With me
  - With sep
- Benchmar
  - ~50 CPU
- Selected s
  - Supermic



Implementation details

For details on DPL workflows, see talk about SW status.

- Multiple GI
  - Less ser
  - Synergie
- Splitting th
  → 2 virtual
  - Still only
- GPUs are
  - Host pro
- GPUs can
  - With mer
  - With sep
- Benchmar
  - ~50 CPU
- Selected s
  - Supermic



Experience from online processing

- **The EPN farm easily handled the online processing.**
- **Peak Pb-Pb rate in 2023 was 47 kHz (slightly less than nominal 50 kHz).**
- **CPU peak load was 32 of 64 cores used (design foresaw 44 cores used, but software was optimized since).**
  - Gives headroom to run additional QC, etc.
- **Minimum free memory: 30%.**
- **Average GPU peak load at peak rate over the farm was 82.5% → 17.5% margin left (v.s. 30% design margin).**
  - TPC data size ~6% higher than expected from simulations.
  - 7 servers not in data taking (in maintenance or excluded for parallel standalone tests).
  - Decided to run some additional algorithms on GPUs, e.g. online TPC dEdx, reducing the margin slightly.
- **Some software improvements are ongoing (some already deployed), and we aim to get back to 30% margin despite the additional processing on GPUs.**

Experience with GPUs from admin / hardware side

- **More GPU failures than other components, still below 3% since purchase in LS2, as expected:**
  - 8 GPUs per server
  - Each GPU has its own memory, voltage regulator, complicated board, etc. in addition to the GPU chip.
- **Second highest are RAM modules.**
- **Majority (>80%) of failures in burn-in phase (first few months)**

- **Vendors are prioritizing first ML, second HPC centers that need FP64, HEP is a special and small customer.**
  - HEP code is more complex than most ML / HPC code, can be challenging for the compilers.
  - Good support with fast turnaround is critical.
  - Once everything is running, one could say “never touch a running system”, but our software is constantly evolving...
- **Running heterogeneous nodes (MI50 with 64 physical CPU cores, MI100 with 96 physical cores) quite smooth, no experience with more different nodes, e.g. different vendors.**

GPU usage for offline reconstruction

ALICE

Online processing fully dominated by TPC

Online reconstruction
(50 kHz Pb-Pb, MC data, no QA / calib)

Processing step	% of time
TPC Processing (Tracking, Clustering, Compression)	99.37 %
EMCAL Processing	0.20 %
ITS Processing (Clustering + Tracking)	0.10 %
TPC Entropy Encoder	0.10 %
ITS-TPC Matching	0.09 %
MFT Processing	0.02 %
TOF Processing	0.01 %
TOF Global Matching	0.01 %
PHOS / CPV Entropy Coder	0.01 %
ITS Entropy Coder	0.01 %
Rest	0.08 %

(650 kHz pp, 2022, no Calorimeters)

Processing step	% of time
TPC Processing (Tracking)	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 %
MCH Tracking	2.02 %
AOD Production	0.88 %
Quality Control	4.00 %
Rest	2.32 %

Offline processing
(47 kHz Pb-Pb, 2024)

Processing step	% of time
TPC Processing (Tracking)	52.39 %
ITS Tracking	12.65 %
Secondary Vertexing	8.97 %
MCH	5.28 %
TRD Tracking	4.39 %
TOF Matching	2.85 %
ITS TPC Matching	2.64 %
Entropy Decoding	2.63 %
AOD Production	1.72 %
Quality Control	1.64 %
Rest	4.84 %

Baseline scenario covers online, which is 99% TPC.
Optimistic scenario shall improve GPU usage in offline.

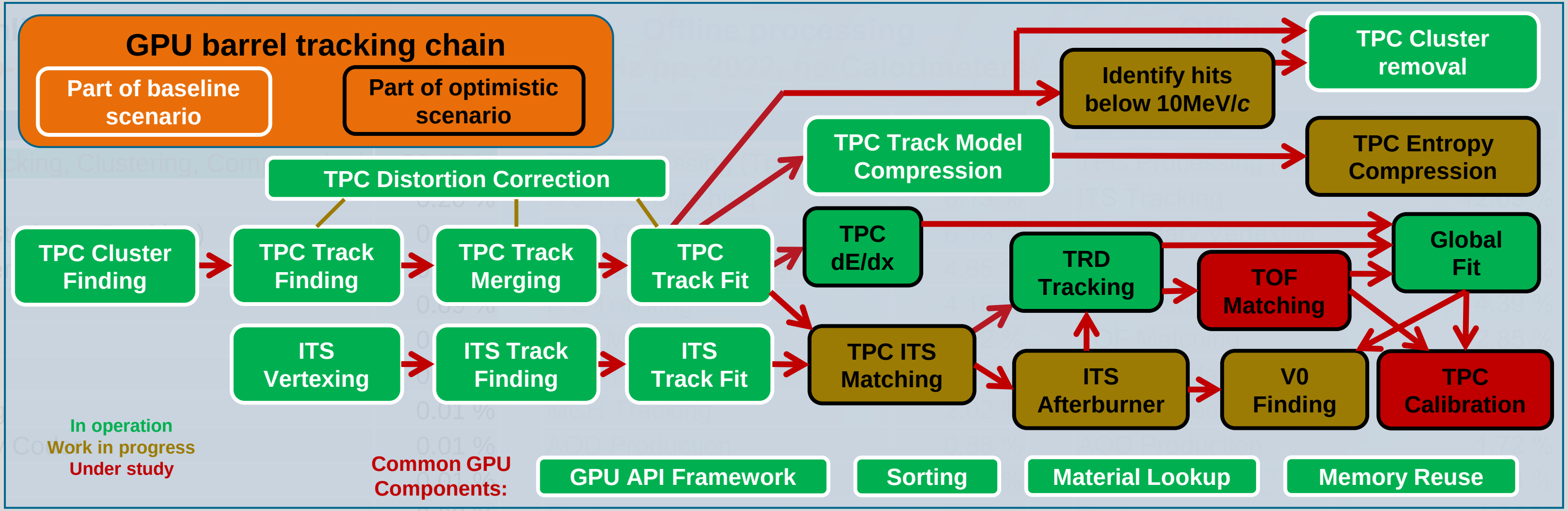
Running on GPU in baseline scenario
1st GPU offload phase – mandatory for online

Running on GPU in optimistic scenario
2nd GPU offload phase – improve offline

GPU usage for offline reconstruction

Candidate for GPU offload in optimistic scenario: Central Barrel Global Tracking Chain

- Consecutive processing steps, thus no need to transfer forth and back between host and GPU.
- Most task tracking related, and can operate on many tracks in parallel.



Running on GPU in baseline scenario
1st GPU offload phase – mandatory for online

Running on GPU in optimistic scenario
2nd GPU offload phase – improve offline

GPU usage for offline reconstruction

Baseline scenario:
~60% on GPU
→ 2.5x speedup

Optimistic scenario:
~80% on GPU
→ 5x speedup

Online reconstruction
(50 kHz Pb-Pb, MC data, no QA / calib)

Processing step	% of time
TPC Processing (Tracking, Clustering, Compression)	99.37 %
EMCAL Processing	0.20 %
ITS Processing (Clustering + Tracking)	0.10 %
TPC Entropy Encoder	0.10 %
ITS-TPC Matching	0.09 %
MFT Processing	0.02 %
TOF Processing	0.01 %
TOF Global Matching	0.01 %
PHOS / CPV Entropy Coder	0.01 %
ITS Entropy Coder	0.01 %
Rest	0.08 %

Offline processing
(650 kHz pp, 2022, no Calorimeters)

Processing step	% of time
TPC Processing (Tracking)	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 %
MCH Tracking	2.02 %
AOD Production	0.88 %
Quality Control	4.00 %
Rest	2.32 %

Offline processing
(47 kHz Pb-Pb, 2024)

Processing step	% of time
TPC Processing (Tracking)	52.39 %
ITS Tracking	12.65 %
Secondary Vertexing	8.97 %
MCH	5.28 %
TRD Tracking	4.39 %
TOF Matching	2.85 %
ITS TPC Matching	2.64 %
Entropy Decoding	2.63 %
AOD Production	1.72 %
Quality Control	1.64 %
Rest	4.84 %

Running on GPU in baseline scenario
1st GPU offload phase – mandatory for online

Running on GPU in optimistic scenario
2nd GPU offload phase – improve offline

Real speedup in offline reconstruction (2023, baseline)

- **For offline reconstruction, EPN nodes are used as GRID nodes.**
  - **Identical workflow** as on other **GRID** sites, only different configuration using GPU, more memory, more CPU cores.
  - EPN farm split in **2 scheduling pools**: online and offline.
    - Unused nodes in the online pool are moved to the offline pool.
    - As needed for data-taking, nodes are moved to the online pool with lead time to let the current jobs finished.
      - If needed immediately, GRID jobs are killed and nodes moved immediately.

Real speedup in offline reconstruction (2023, baseline)

- **For offline reconstruction, EPN nodes are used as GRID nodes.**
  - **Identical workflow** as on other **GRID** sites, only different configuration using GPU, more memory, more CPU cores.
  - EPN farm split in **2 scheduling pools**: online and offline.
    - Unused nodes in the online pool are moved to the offline pool.
    - As needed for data-taking, nodes are moved to the online pool with lead time to let the current jobs finished.
      - If needed immediately, GRID jobs are killed and nodes moved immediately.
- **Performance benchmarks cover multiple cases:**
  - EPN split into 16 * **8 cores**, or into 8 * **16 cores**, ignoring the GPU : to compare CPUs and GPUs.
  - EPN split into 8 or 2 identical fractions: **1 NUMA** domain (4 GPUs) or **1 GPU**.
- **Processing time per time-frame while the GRID job is running (neglecting overhead at begin / end).**
  - In all cases server **fully loaded** with **identical jobs**, to avoid effects from HyperThreading, memory, etc.

For a fair comparison, needed to determine the fastest CPU-only and fastest GPU configuration of offline reconstruction.
For all settings, obtained the optimal process multiplicity tuning settings.

Real speedup in offline reconstruction (2023, baseline)

- **For offline reconstruction, EPN nodes are used as GRID nodes.**
  - **Identical workflow** as on other **GRID** sites, only different configuration using GPU, more memory, more CPU cores.
  - EPN farm split in **2 scheduling pools**: online and offline.
    - Unused nodes in the online pool are moved to the offline pool.
    - As needed for data-taking, nodes are moved to the online pool with lead time to let the current jobs finished.
      - If needed immediately, GRID jobs are killed and nodes moved immediately.
- **Performance benchmarks cover multiple cases:**
  - EPN split into 16 * **8 cores**, or into 8 * **16 cores**, ignoring the GPU : to compare CPUs and GPUs.
  - EPN split into 8 or 2 identical fractions: **1 NUMA** domain (4 GPUs) or **1 GPU**.
- **Processing time per time-frame while the GRID job is running (neglecting overhead at begin / end).**
  - In all cases server **fully loaded** with **identical jobs**, to avoid effects from HyperThreading, memory, etc.

Configuration (2022 pp, 650 kHz)	Time per TF (11ms, 1 instance)	Time per TF (11ms, full server)	
CPU 8 core	76.91s	4.81s	
CPU 16 core	34.18s	**4.27s**	Factor 2.51 / Matches expected factor 2.5
1 GPU + 16 CPU cores	14.60s	1.83s	
1 NUMA domain (4 GPUs + 64 cores)	3.5s	**1.70s**	

Configuration used for async processing on EPNs. (Also resembles most the online processing configuration)

Offline reconstruction on GPU : plans

- **Gradually shifting to running more steps on GPU (optimistic scenario).**
  - Several components seem ready, but integration is pending...
    - **ITS** and **TPC** standalone tracking can run on **GPU**, but **not yet within the same process**.
    - **TRD** tracking on **GPU** is **ready**, but **needs TPC-ITS** matched tracks as input, which are **not yet available** on GPUs.
  - But GPU usage is slowly increasing.
    - **TPC CTF track model decoding** was **ported to GPU** recently, yielding **1.5% to 5% speedup in 2024** (depending on which data).
    - Done by a **master student in 6 months**, showing that the framework can be used by newcomers to move code to GPU.

Offline reconstruction on GPU : plans

- **Gradually shifting to running more steps on GPU (optimistic scenario).**
  - Several components seem ready, but integration is pending...
    - **ITS** and **TPC** standalone tracking can run on **GPU**, but **not yet within the same process**.
    - **TRD** tracking on **GPU** is **ready**, but **needs TPC-ITS** matched tracks as input, which are **not yet available** on GPUs.
  - But GPU usage is slowly increasing.
    - **TPC CTF track model decoding** was **ported to GPU** recently, yielding **1.5% to 5% speedup in 2024** (depending on which data).
    - Done by a **master student in 6 months**, showing that the framework can be used by newcomers to move code to GPU.

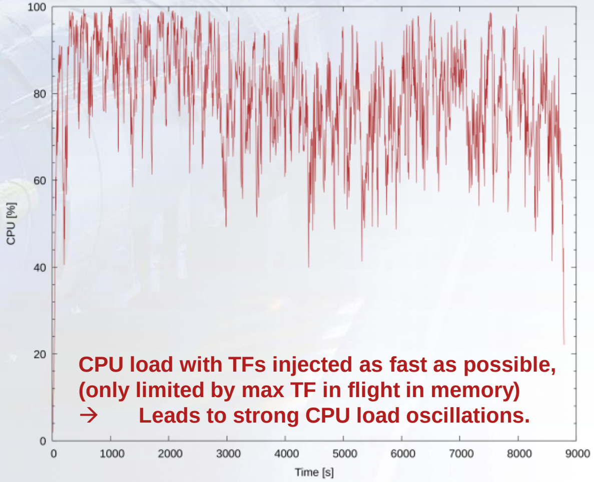
- **Facing 2 challenges running on other GPU models in the GRID:**
  - Need to provide **software compiled** for the **on-site GPU model** on **CVMFS**.
    - So far have a list of AMD and NVIDIA GPU types for which we compile.
      - **Compile time** increases by **~3 minutes per GPU type**, cannot simply compile for all models.
      - Using **run time compilation** for optimizations, could **compile** for **additional GPU types on the fly**.
  - **Process multiplicity tuning** depending on **number of CPU cores** / **GPU model performance**.
    - Currently setting up for test on **NERSC** site.
      - Can get interactive sessions for testing.
      - **Similar to EPN**, 64 cores, 1 NVIDIA GPU, but more powerful than MI50, so fits for 64 cores.

Online v.s. Offline on GPU

- **Online**:
  - Time frames come in at **fixed rate**, and **processing needs to keep up**.
  - Aiming for “**GPU-bound**” processing at **~70% GPU load** (**30% margin**) – load during **2023 Pb-Pb** was **82.5%** load.
  - **CPUs** should stay **below 70%** load – load during **2023 Pb-Pb** was **~50%**.

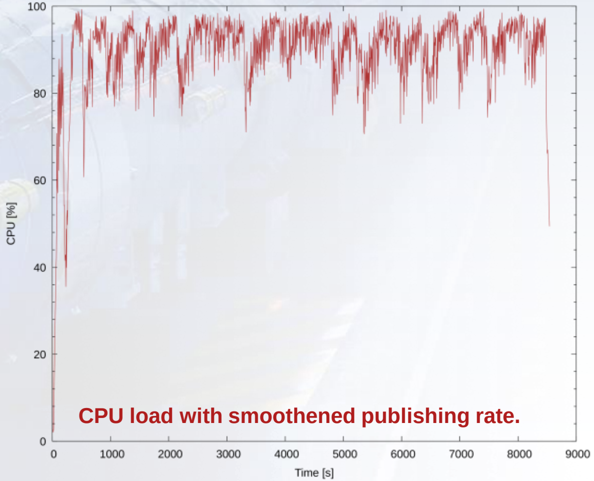
Online v.s. Offline on GPU

- **Online**:
  - Time frames come in at **fixed rate**, and **processing needs to keep up**.
  - Aiming for "**GPU-bound**" processing at **~70% GPU load** (**30% margin**) – load during **2023 Pb-Pb** was **82.5%** load.
  - **CPUs** should stay **below 70%** load – load during **2023 Pb-Pb** was **~50%**.
- **Offline**:
  - We can **define** the **time frame publishing rate** at the **source**.
    - **Naive approach**: publish **as fast as possible** with limiting the maximum number of time frames in flight.
    - **Yields oscillations** in the processing chain...



Online v.s. Offline on GPU

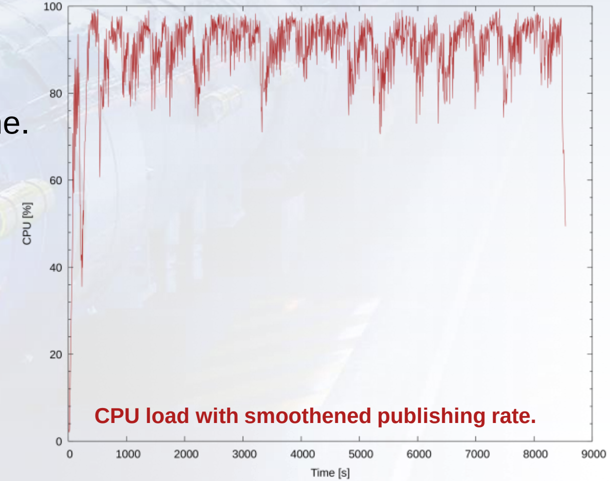
- **Online**:
  - Time frames come in at **fixed rate**, and **processing needs to keep up**.
  - Aiming for "**GPU-bound**" processing at **~70% GPU load** (**30% margin**) – load during **2023 Pb-Pb** was **82.5%** load.
  - **CPUs** should stay **below 70%** load – load during **2023 Pb-Pb** was **~50%**.
- **Offline**:
  - We can **define** the **time frame publishing rate** at the **source**.
    - **Naive approach**: publish **as fast as possible** with limiting the maximum number of time frames in flight.
    - **Yields oscillations** in the processing chain, better to **smoothen** the publishing rate.



CPU load with smoothened publishing rate.

Online v.s. Offline on GPU

- **Online**:
  - Time frames come in at **fixed rate**, and **processing needs to keep up**.
  - Aiming for "**GPU-bound**" processing at **~70% GPU load** (**30% margin**) – load during **2023 Pb-Pb** was **82.5%** load.
  - **CPUs** should stay **below 70%** load – load during **2023 Pb-Pb** was **~50%**.
- **Offline**:
  - We can **define** the **time frame publishing rate** at the **source**.
    - **Naive approach**: publish **as fast as possible** with limiting the maximum number of time frames in flight.
    - **Yields oscillations** in the processing chain, better to **smoothen** the publishing rate.
  - **Aiming for 100% CPU load**, and offloading as much as possible to GPU.
    - Processing **CPU-bound**, even inefficient GPU offload will decrease the wall time.
  - **Baseline scenario** on EPNs: **60%** of **workload** on **GPUs**, but **GPUs** have **90%** of the **compute power**
    → **GPU load < 50%**.
    - Running with 2 instead of 4 GPUs on the EPN gives the same performance
    - Thus NVIDIA system with 1 fast GPU can keep up.

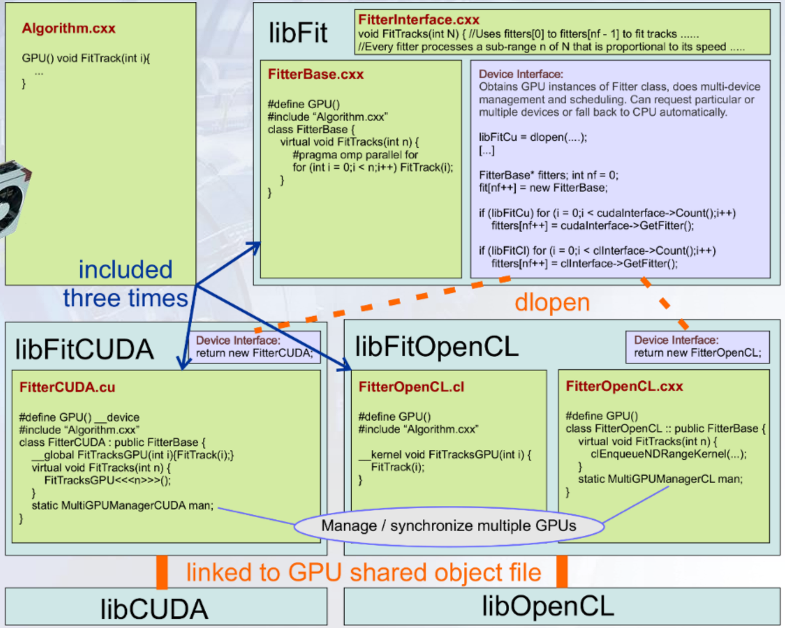


CPU load with smoothened publishing rate.

Plugin system for multiple APIs with common source code

- **Generic common C++ Code compatible to CUDA, OpenCL, HIP, and CPU (with pure C++, OpenMP, or OpenCL).**
  - OpenCL needs clang compiler (ARM or AMD ROCm) or AMD extensions (TPC track finding only on Run 2 GPUs and CPU for testing)
  - Certain worthwhile algorithms have a vectorized code branch for CPU using the Vc library
  - All GPU code swapped out in dedicated libraries, same software binaries run on GPU-enabled and CPU servers

- **Screening different platforms for best price / performance.**
  (including some non-competitive platforms for cross-checks and validation.)
  - **CPUs** (**AMD Zen**, **Intel Skylake**)
    C++ backend with **OpenMP**, AMD **OCL**
  - **AMD GPUs**
    (**S9000** with **OpenCL 1.2**, **MI50** / **Radeon 7** / **Navi** with **HIP** / **OCL 2.x**)
  - **NVIDIA GPUs**
    (**RTX 2080** / **RTX 2080 Ti** / **Tesla T4** with **CUDA**)
  - **ARM Mali GPU** with **OCL 2.x**
    (Tested on dev-board with Mali G52)



Conclusions

- **ALICE employs GPUs heavily to speed up online and offline processing.**
  - **99%** of **online reconstruction** on the **GPU** (no reason at all to port the rest).
  - Since 2023 ~**60%** of full **offline processing** (for 650 kHz pp) on **GPU** (if offline jobs on the EPN farm).
    - Aim to increase to **80%** with full barrel tracking on GPU (**optimistic scenario**).
    - Proof of concept workflow running also on server with NVIDIA GPUs, next step is to test at NERSC.
- **Online processing successful in 2021 - 2024.**
  - **pp** data taking and **Pb-Pb** went **smooth** up to the highest Pb-Pb rate (47 kHz) in 2023.
  - **GPU Compute margin was 17.5%.**
  - **Future improvements should restore the 30% design margin.**
- **Online farm would need >3000 64-core servers if built with CPUs only – prohibitively expensive.**
- **Offline reconstruction runs TPC reconstruction on the GPUs in the EPN farm, and in CPU-only style on the CERN GRID site.**
  - **EPN** nodes are **2.5x** faster when using **GPUs**.
  - **Optimistic scenario** should increase this to **5x**.
  - **Working on first test** to run with **GPUs** (of different vendor) on **GRID** sites (NERSC).

Lessons learned

- **GPUs can speed up the processing significantly.**
  - Not necessarily all workloads needs to run on GPU, but the hot spot.
- **Inexperienced users can contribute improvements to algorithms, for implementing full new reconstruction steps on GPU more expert knowledge is needed.**
- **Scheduling for online and offline processing is different.**
- **Should also optimize for memory perhaps sacrificing a bit of performance.**
  - ALICE reduced TF length in 2023 from 11ms to 2.8ms to reduce the memory footprint.
  - **Memory** is more **limited** on GRID sites than on your online farm.
- **A common software framework for multiple GPU types allows for changing the vendor and simplifies debugging.**
- **Default build should contain all GPU backends, to be enabled transparently and optionally (e.g. via plugins).**
- **Having the full reconstruction in a single monolithic process is failure-prone and difficult to debug (Run 3), too many individual processes can have huge memory demand → good compromise needed.**
- **No fallback for too slow online processing, and there are always unforeseen effects. 30% compute margin turned out reasonable.**
- **Our code might have "average complexity" as CPU application, but our GPU code is more complicated than ML / most HPC code and compilers might not be ready for it.**

Lessons learned

- **GPUs can speed up the processing significantly.**
  - Not necessarily all workloads needs to run on GPU, but the hot spot.
- **Inexperienced users can contribute improvements to algorithms, for implementing full new reconstruction steps on GPU more expert knowledge is needed.**
- **Scheduling for online and offline processing is different.**
- **Should also optimize for memory perhaps sacrificing a bit of performance.**
  - ALICE reduced TF length in 2023 from 11ms to 2.8ms to reduce the memory footprint.
  - Memory is more limited on GRID sites than on your online farm.
- **A common software framework for multiple GPU types allows for changing the vendor and simplifies debugging.**
- **Default build should contain all GPU backends, to be enabled transparently and optionally (e.g. via plugins).**
- **Having the full reconstruction in a single monolithic process is failure-prone and difficult to debug (Run 3), too many individual processes can have huge memory demand → good compromise needed.**
- **No fallback for too slow online processing, and there are always unforeseen effects. 30% compute margin turned out reasonable.**
- **Our code might have "average complexity" as CPU application, but our GPU code is more complicated than ML / most HPC code and compilers might not be ready for it.**
  - Meanwhile filed > 150 bug reports to AMD, ARM, Clang, NVIDIA, actually stopped counting at 100...

Bug repors:
https://github.com/RadeonOpenCompute/ROCm/issues/866
https://github.com/ROCmSoftwarePlatform/hipCUB/issues/50
https://github.com/RadeonOpenCompute/hcc/issues/1257
https://github.com/RadeonOpenCompute/hcc/issues/1274
https://github.com/davidrohr/AliceO2/issues/4
https://github.com/ROCm-Developer-Tools/HIP/pull/894
https://github.com/ROCm-Developer-Tools/HIP/pull/1302
https://github.com/ROCm-Developer-Tools/HIP/issues/892
https://github.com/ROCm-Developer-Tools/HIP/issues/893
https://github.com/ROCm-Developer-Tools/HIP/issues/1107
https://github.com/ROCm-Developer-Tools/HIP/issues/1126
https://github.com/ROCm-Developer-Tools/HIP/issues/1131
https://github.com/ROCm-Developer-Tools/HIP/issues/1141
https://github.com/ROCm-Developer-Tools/HIP/issues/1185
https://github.com/ROCm-Developer-Tools/HIP/issues/1314
https://github.com/ROCm-Developer-Tools/HIP/issues/1335
https://github.com/ROCm-Developer-Tools/HIP/issues/1401
https://github.com/ROCm-Developer-Tools/HIP/issues/1493
https://github.com/ROCm-Developer-Tools/HIP/issues/1532
https://github.com/ROCm-Developer-Tools/HIP/issues/1538
https://github.com/ROCm-Developer-Tools/HIP/issues/1556
https://github.com/KhronosGroup/SPIRV-LLVM-Translator/issues/302
https://github.com/KhronosGroup/SPIRV-LLVM-Translator/issues/303
https://github.com/KhronosGroup/SPIRV-LLVM-Translator/issues/304
https://github.com/KhronosGroup/SPIRV-LLVM-Translator/issues/357
https://github.com/KhronosGroup/SPIRV-LLVM-Translator/issues/385
https://github.com/ROCmSoftwarePlatform/rocThrust/issues/61
https://github.com/KhronosGroup/OpenCL-Docs/issues/50
https://github.com/KhronosGroup/OpenCL-Docs/issues/66
https://bugs.llvm.org/show_bug.cgi?id=40603
https://bugs.llvm.org/show_bug.cgi?id=40707
https://bugs.llvm.org/show_bug.cgi?id=40778
https://bugs.llvm.org/show_bug.cgi?id=41070
https://bugs.llvm.org/show_bug.cgi?id=41567
https://bugs.llvm.org/show_bug.cgi?id=41609
https://bugs.llvm.org/show_bug.cgi?id=41963
https://bugs.llvm.org/show_bug.cgi?id=42031
https://bugs.llvm.org/show_bug.cgi?id=42033
https://bugs.llvm.org/show_bug.cgi?id=42060
https://bugs.llvm.org/show_bug.cgi?id=42385
https://bugs.llvm.org/show_bug.cgi?id=42387
https://bugs.llvm.org/show_bug.cgi?id=42390
https://bugs.llvm.org/show_bug.cgi?id=43057
https://bugs.llvm.org/show_bug.cgi?id=43145
https://bugs.llvm.org/show_bug.cgi?id=44176
https://bugs.llvm.org/show_bug.cgi?id=44177
https://reviews.llvm.org/D59603
https://reviews.llvm.org/D58708
https://reviews.llvm.org/D58719
https://reviews.llvm.org/D59646