The O2 software framework and GPU usage in ALICE online and offline reconstruction in Run 3

David Rohr, Giulio Eulisse for the ALICE Collaboration

CHEP 2023

10.5.2023

drohr@cern.ch

ALICE DATA TAKING / PROCESSING CONCEPT

David Rohr, drohr@cern.ch

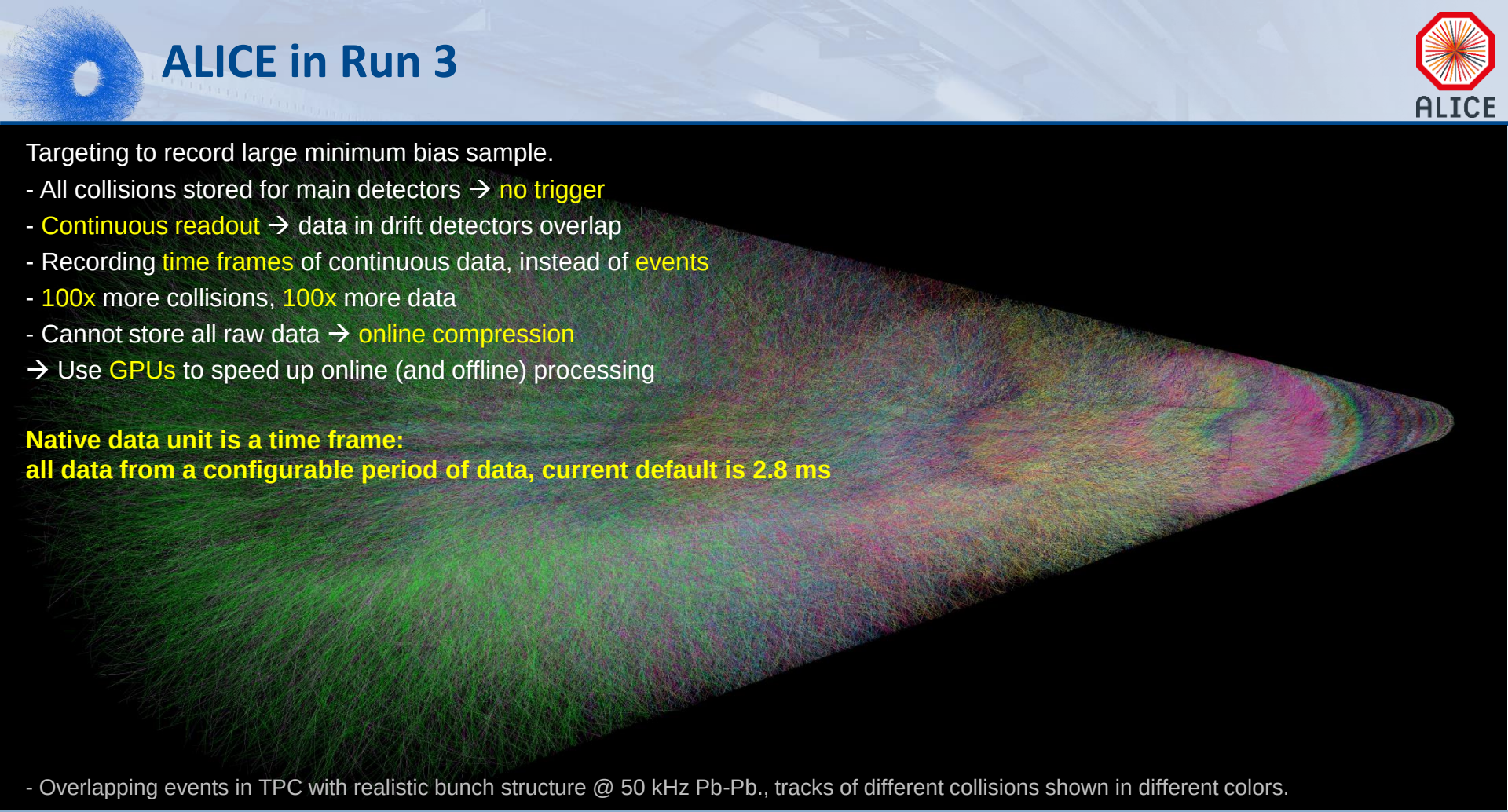
ALICE in Run 3

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, 100x more data
- Cannot store all raw data → online compression

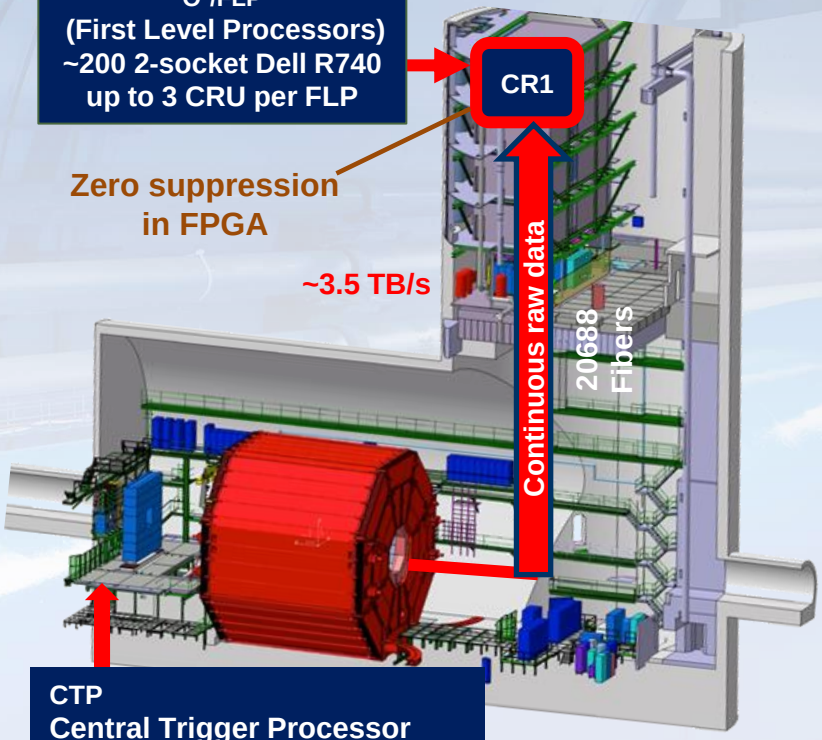
→ Use GPUs to speed up online (and offline) processing

Native data unit is a time frame:
all data from a configurable period of data, current default is 2.8 ms

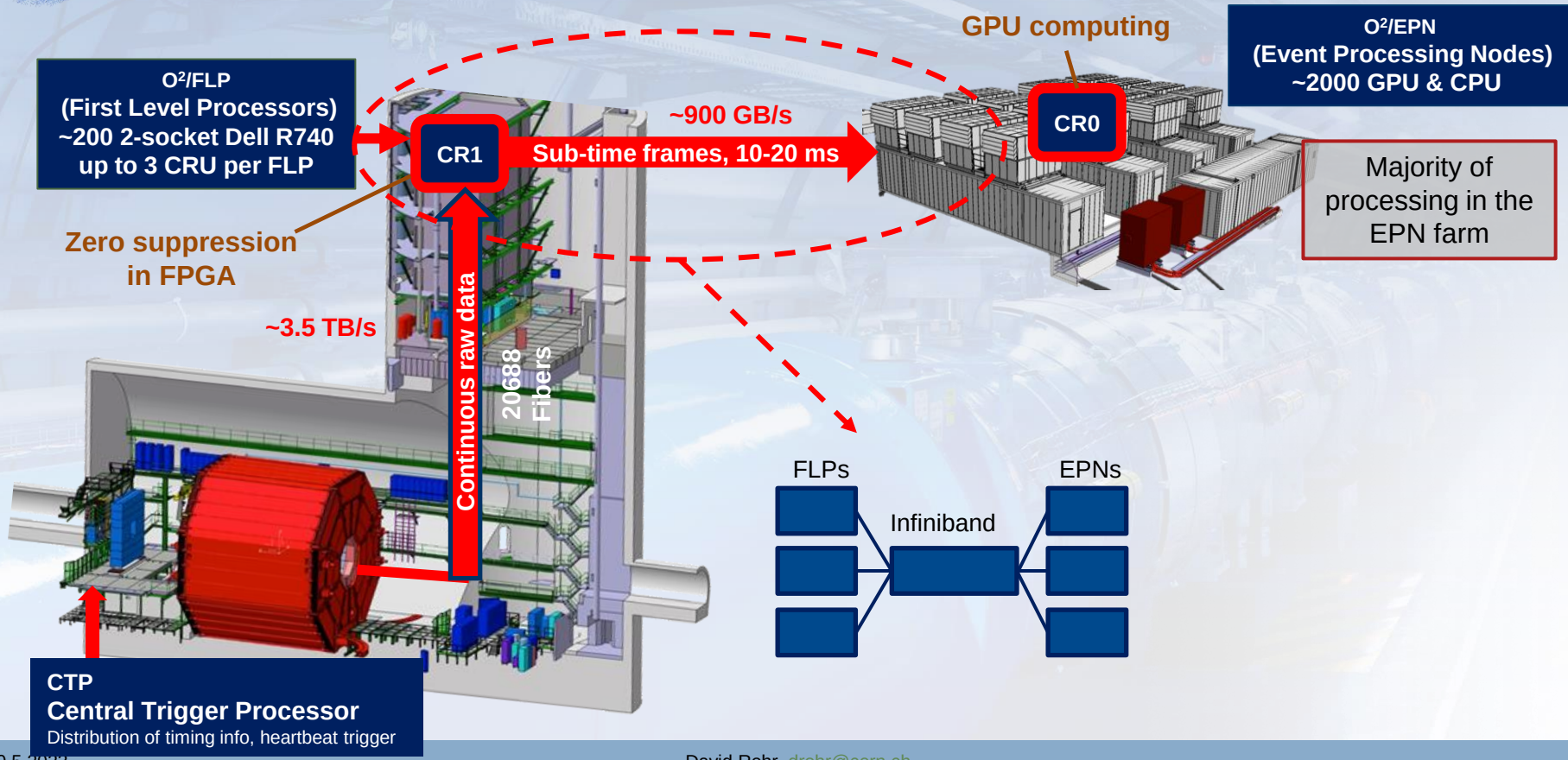


- Overlapping events in TPC with realistic bunch structure @ 50 kHz Pb-Pb., tracks of different collisions shown in different colors.

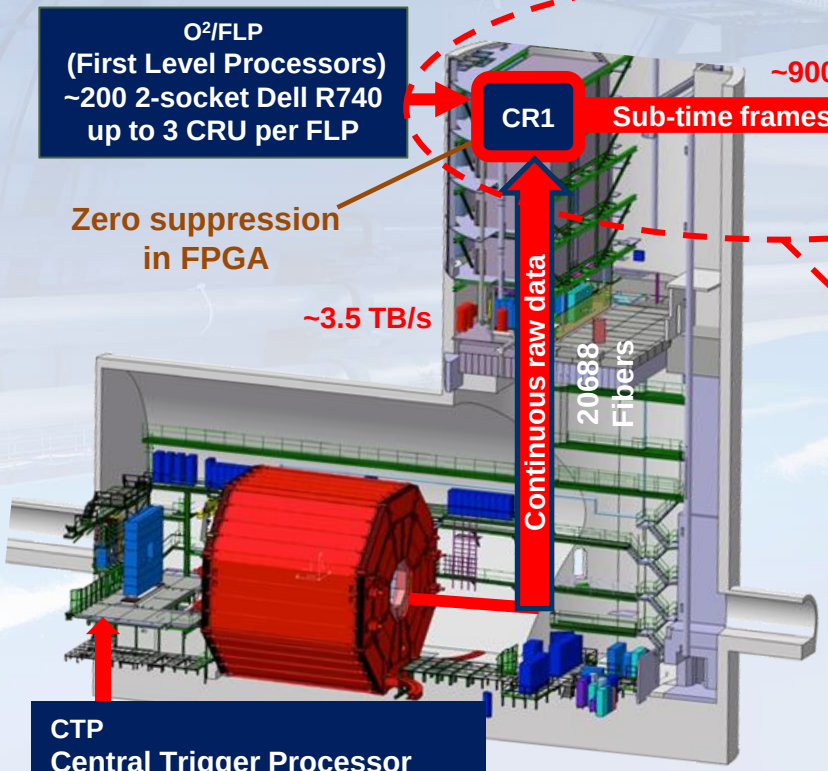
ALICE Raw Data Flow in Run 3



ALICE Raw Data Flow in Run 3



ALICE Raw Data Flow in Run 3



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

CR1

~900 GB/s
Sub-time frames, 10-20 ms

GPU computing

CR0

O²/EPN
(Event Processing Nodes)
~2000 GPU & CPU

Majority of processing in the EPN farm

Zero suppression in FPGA

~3.5 TB/s

Continuous raw data

20688 Fibers

~130 GB/s
CTF: Compressed time frames
Calibration data

80 PB
disk storage, 360GB/s
(~25% redundancy)

FLPs
Infiniband
EPNs

~70 % CTF
~30 % CTF

Tier 0
archival

Tier 1
archival

CTP
Central Trigger Processor
Distribution of timing info, heartbeat trigger

Synchronous and Asynchronous Processing

Data links from detectors > 3.5 TB/s

Run 3 farm

Readout nodes > 900 GB/s

Synchronous processing
- Local processing
- Event / timeframe building
- Calibration / reconstruction

During Data taking

Disk buffer

Asynchronous processing
- Reprocessing with full calibration
- Full reconstruction

During no beam

Reconstructed Data

Compressed Raw Data

Permanent storage ~ 130 GB/s

GPU computing

O²/EPN (Event Processing Nodes) ~2000 GPU & CPU

Synchronous processing

~130 GB/s

CTF: Compressed time frames

Calibration data

80 PB

disk storage, 360GB/s (~25% redundancy)

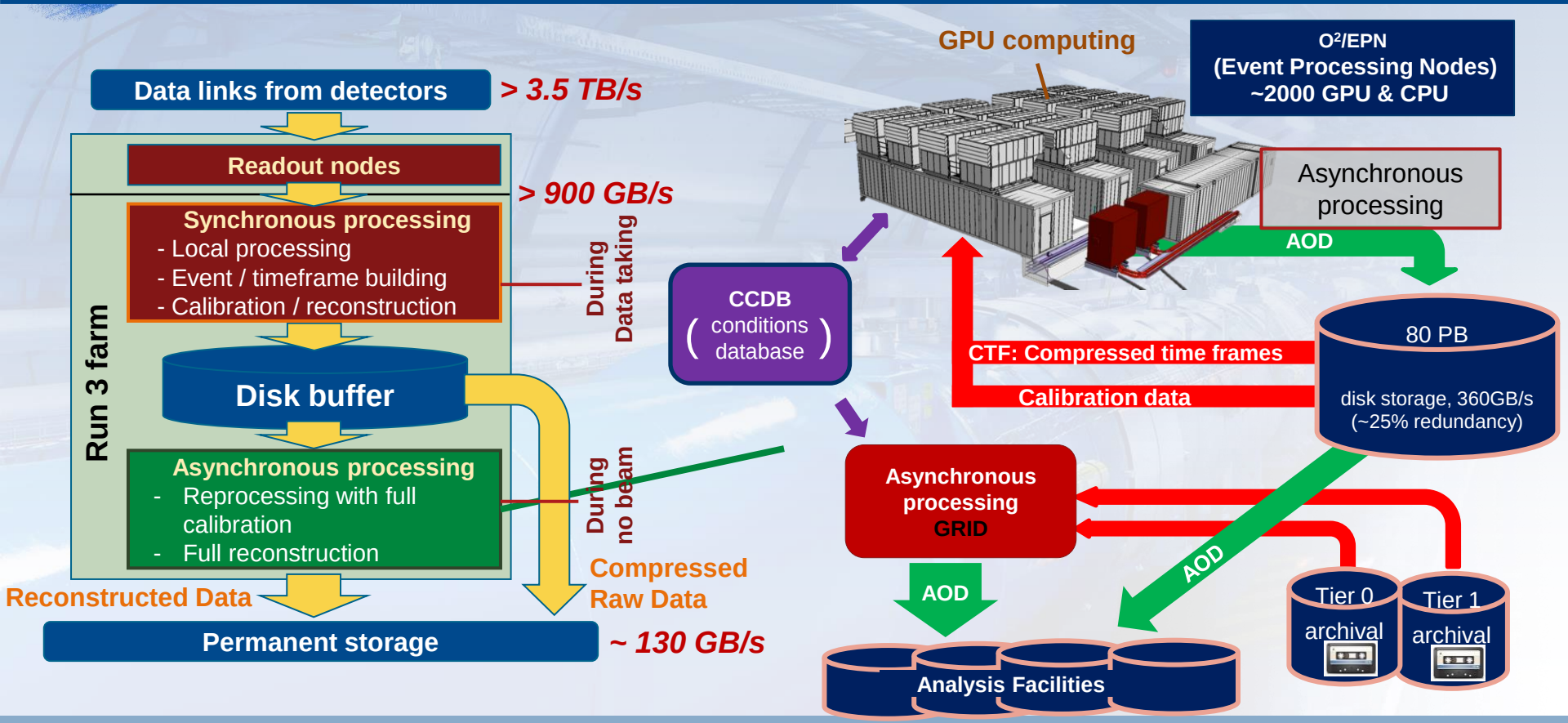
~70 % CTF

~30 % CTF

Tier 0 archival

Tier 1 archival

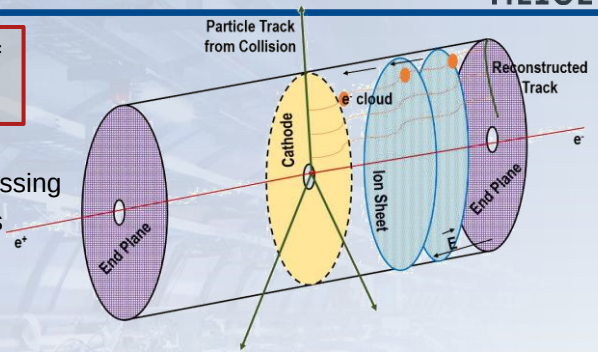
Synchronous and Asynchronous Processing



O^2 Processing steps

- **Synchronous processing (what we called online before):**
  - 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

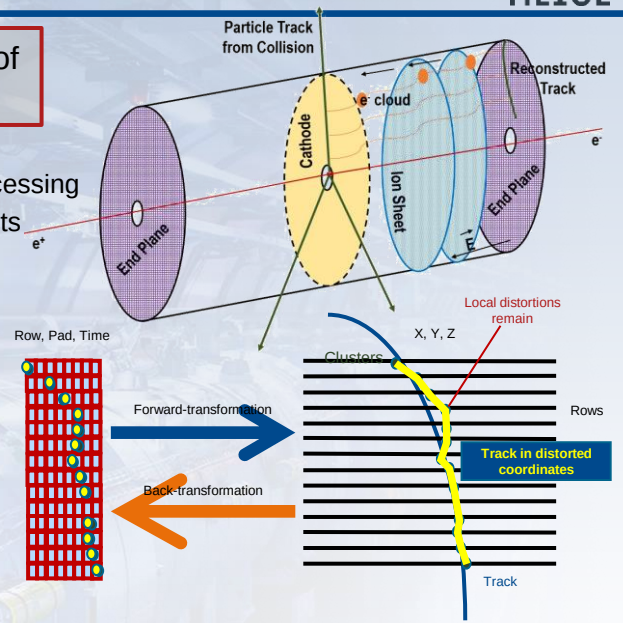


O² Processing steps

- **Synchronous processing (what we called online before):**
  - 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

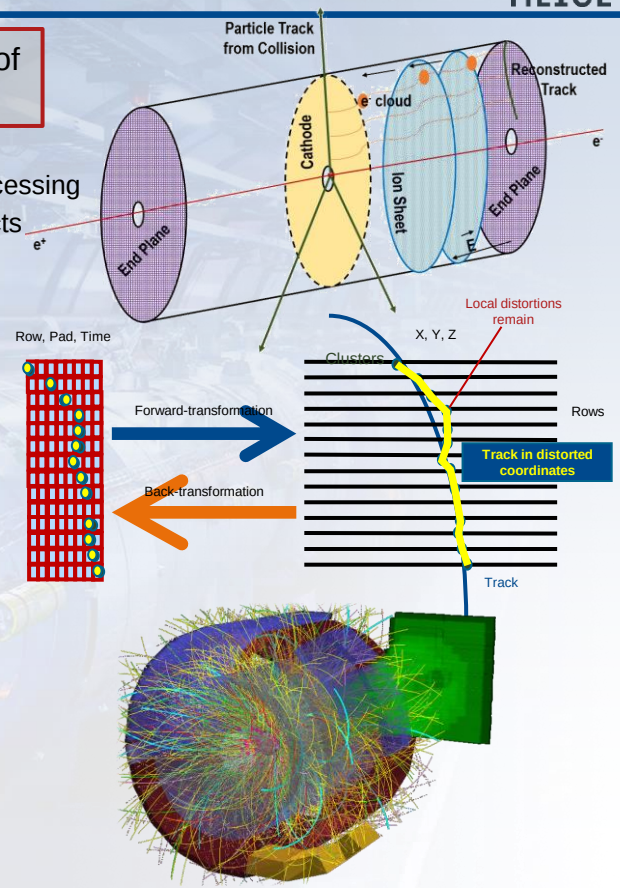


O^2 Processing steps

- **Synchronous processing (what we called online before):**
  - 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

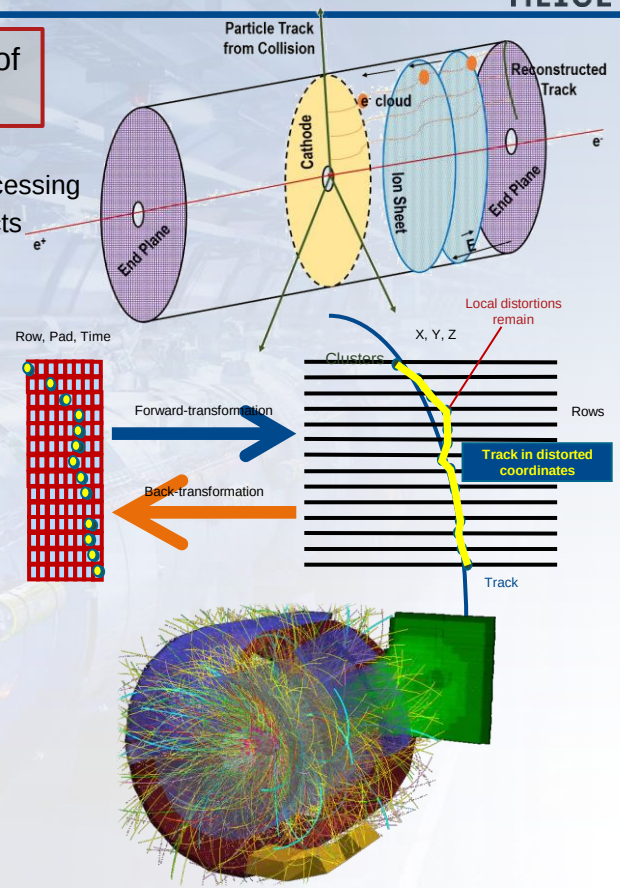


O^2 Processing steps

- **Synchronous processing (what we called online before):**
  - 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



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

ALICE GPU USAGE STRATEGY

David Rohr, drohr@cern.ch

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
>2000 * AMD MI50 in **Run 3**
Online and Offline barrel tracking

Overview of compute time of reconstruction steps

- **The table below shows the relative compute time (linux cpu time) of the processing steps running on the processor.**

Totally dominated by TPC: >99%

Synchronous processing
(50 kHz Pb-Pb, MC data)

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 %

Asynchronous processing
(650 kHz pp, real data, calorimeters not in run)

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 %

Only data processing steps
Quality control, calibration, event building excluded!

Overview of compute time of reconstruction steps

Synchronous processing
(50 kHz Pb-Pb, MC data)

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 %

Only data processing steps
Quality control, calibration, event building excluded!

- **Synchronous processing** :
  - **99%** of compute time spent for **TPC**.
  - **EPN farm build for synchronous processing!**
- **Asynchronous reprocessing :**
  - More detectors with significant computing contribution.
  - To be kept in mind, as EPNS also run async. Reco.

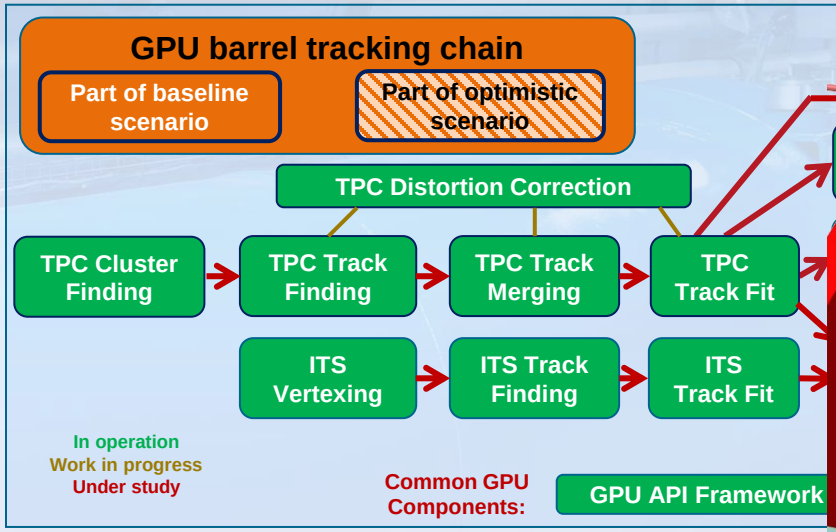
- **GPUs** well suited for **TPC** reco (from Run 1 and 2 experience).
- **GPUs** provide the **required compute power**.
- Time frame concepts yields large enough GPU data chunks.
- Following up **2 scenarios** for EPN GPU processing:

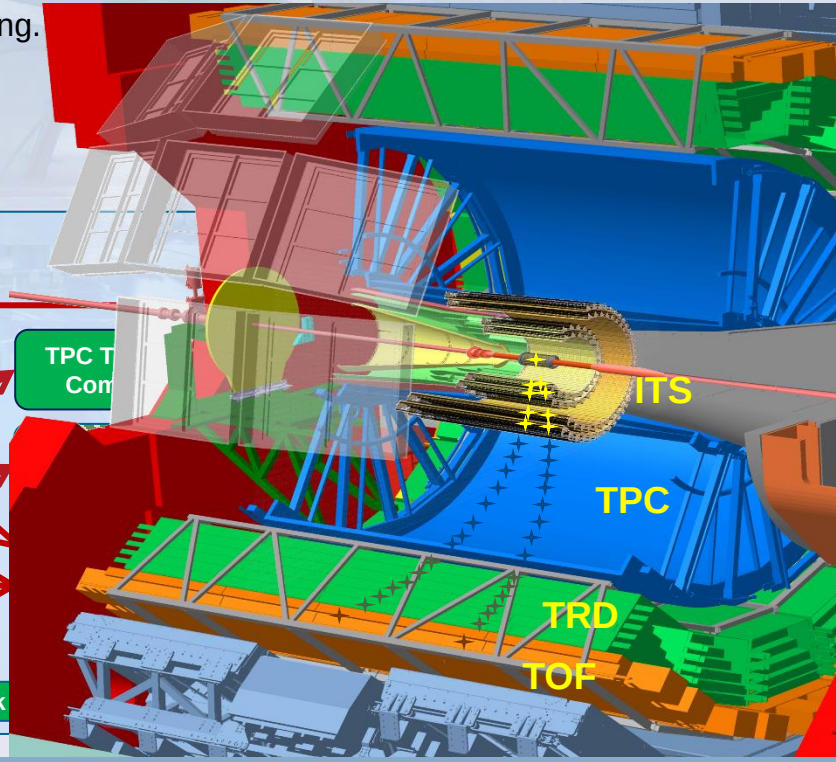
Baseline solution (available today):
- Mandatory for synchronous processing
TPC sync. reco on GPU

Optimistic solution (under development):
- Achieve best GPU usage in async phase
- Run most of tracking + X on GPU

Central barrel global tracking chain

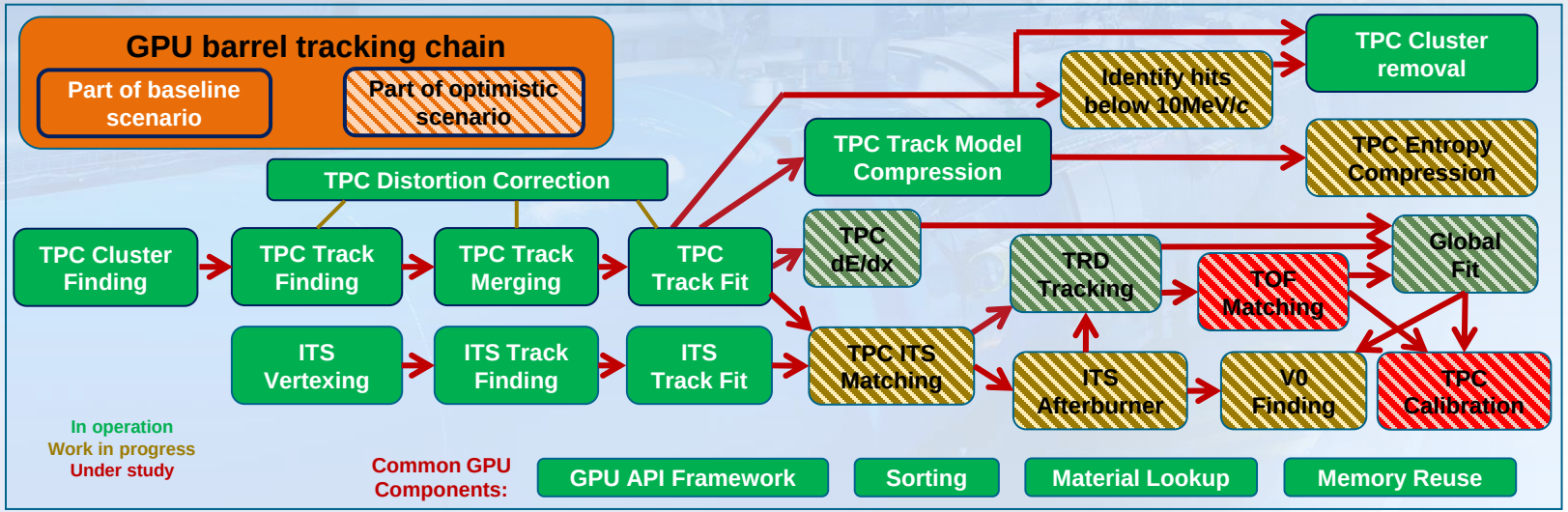
- **Central barrel tracking chosen as best candidate for optimistic scenario for asynchronous reco:**
  - Mandatory **baseline scenario** includes everything that must run on the GPU during synchronous reconstruction.
  - **Optimistic scenario** includes everything related to the barrel tracking.





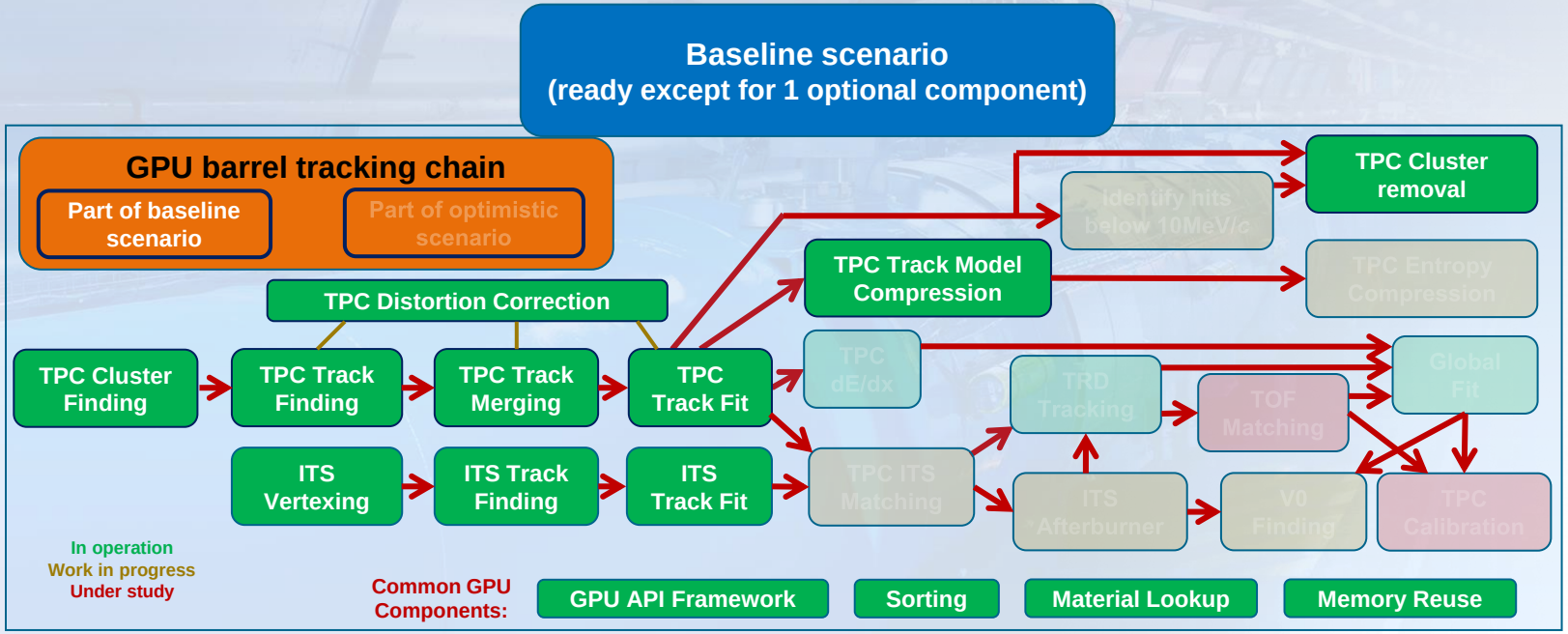
Central barrel global tracking chain

- **Central barrel tracking chosen as best candidate for optimistic scenario for asynchronous reco:**
  - Mandatory **baseline scenario** includes everything that must run on the GPU during synchronous reconstruction.
  - **Optimistic scenario** includes everything related to the barrel tracking.



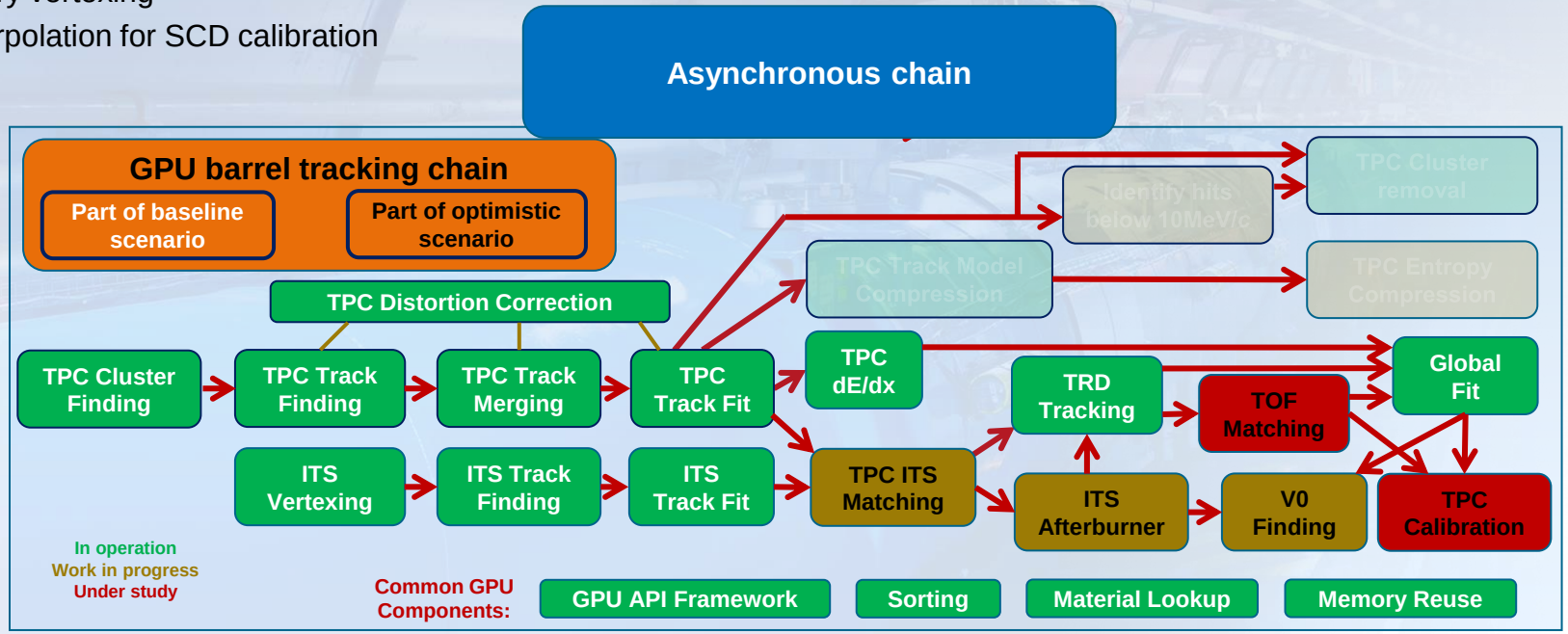
Central barrel global tracking chain

- **Baseline scenario fully implemented.**
  - Not mandatory to speed up the synchronous GPU code further.



Central barrel global tracking chain

- **Several steps missing in asynchronous reconstruction:**
  - Matching to ITS
  - Matching to TOF
  - Secondary vertexing
  - TPC interpolation for SCD calibration



IMPLEMENTATION

David Rohr, drohr@cern.ch

Implementation principles

1. **GPU code should be modular, such that individual parts can run independently.**
   - Multiple consecutive components on the GPU should operate with as little host interaction as possible.
2. **GPU code should not depend on one particular vendor or API.**
   - No usage of special features that are not portable.
3. **GPU usage should be optional: running the O2 software should not require any vendor libraries installed.**
   - All GPU code is contained in plugins, with a common interface.
   - Using CPU or GPU version is transparent to the user.
   - In principle, even multiple plugins (GPU backends) can run on the same node.
4. **Not much time should be spend for memory management.**
   - We allocate one large memory segment, and then distribute memory chunks internally.
5. **Processing on GPU and data transfer should overlap, such that the GPU does not idle while waiting for data.**
   - This is implemented via a pipelined processing within time frames, and we also overlap consecutive time frames.
6. **Data chunks (time frames) processed by the GPU must be large enough to exploit the full parallelism.**
7. **GPU and CPU output should be as close as possible.**
   - But small differences due to concurrency or non-associative floating point arithmetic cannot be avoided.

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).

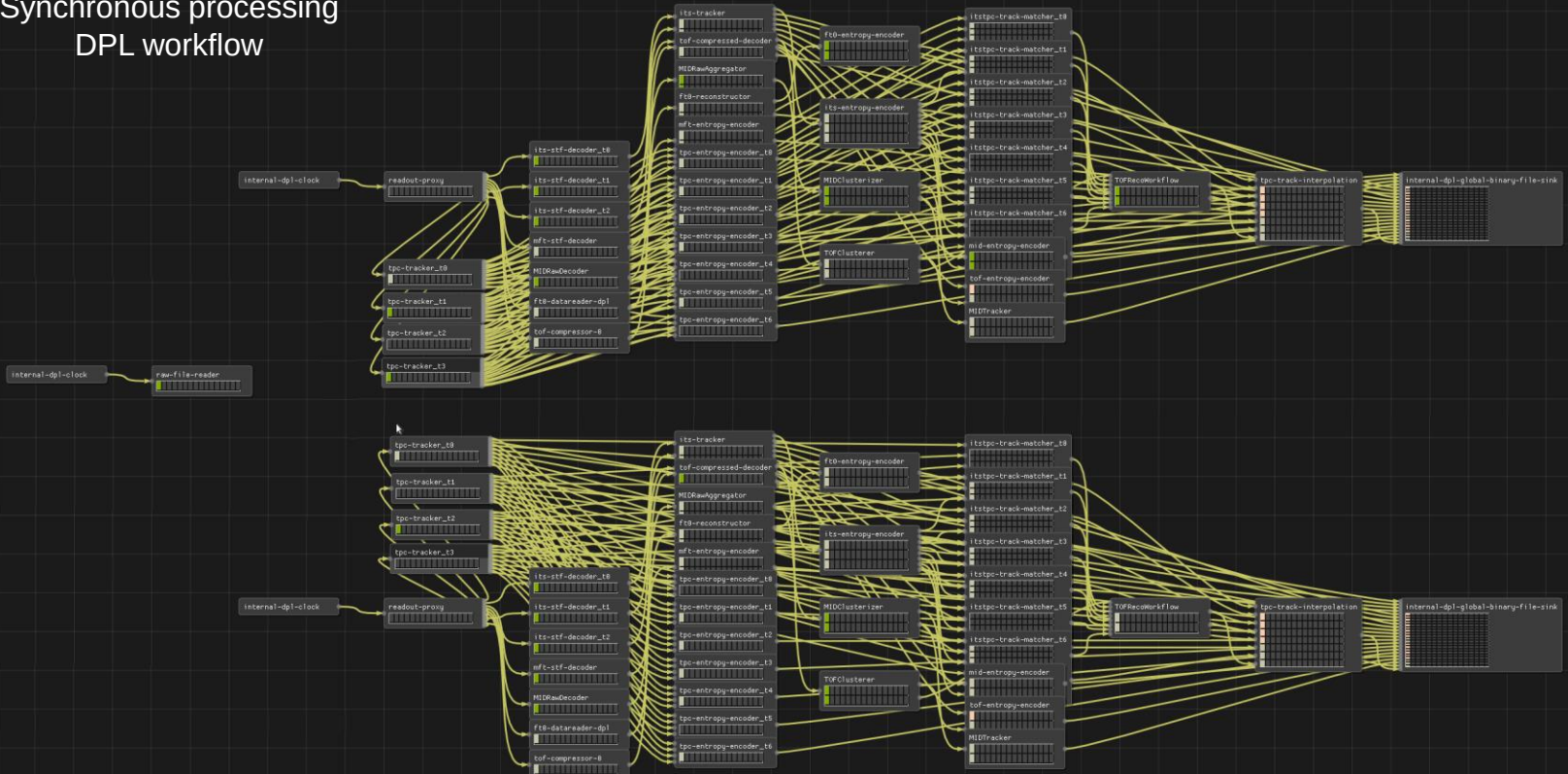
Implementation details

- **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, 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

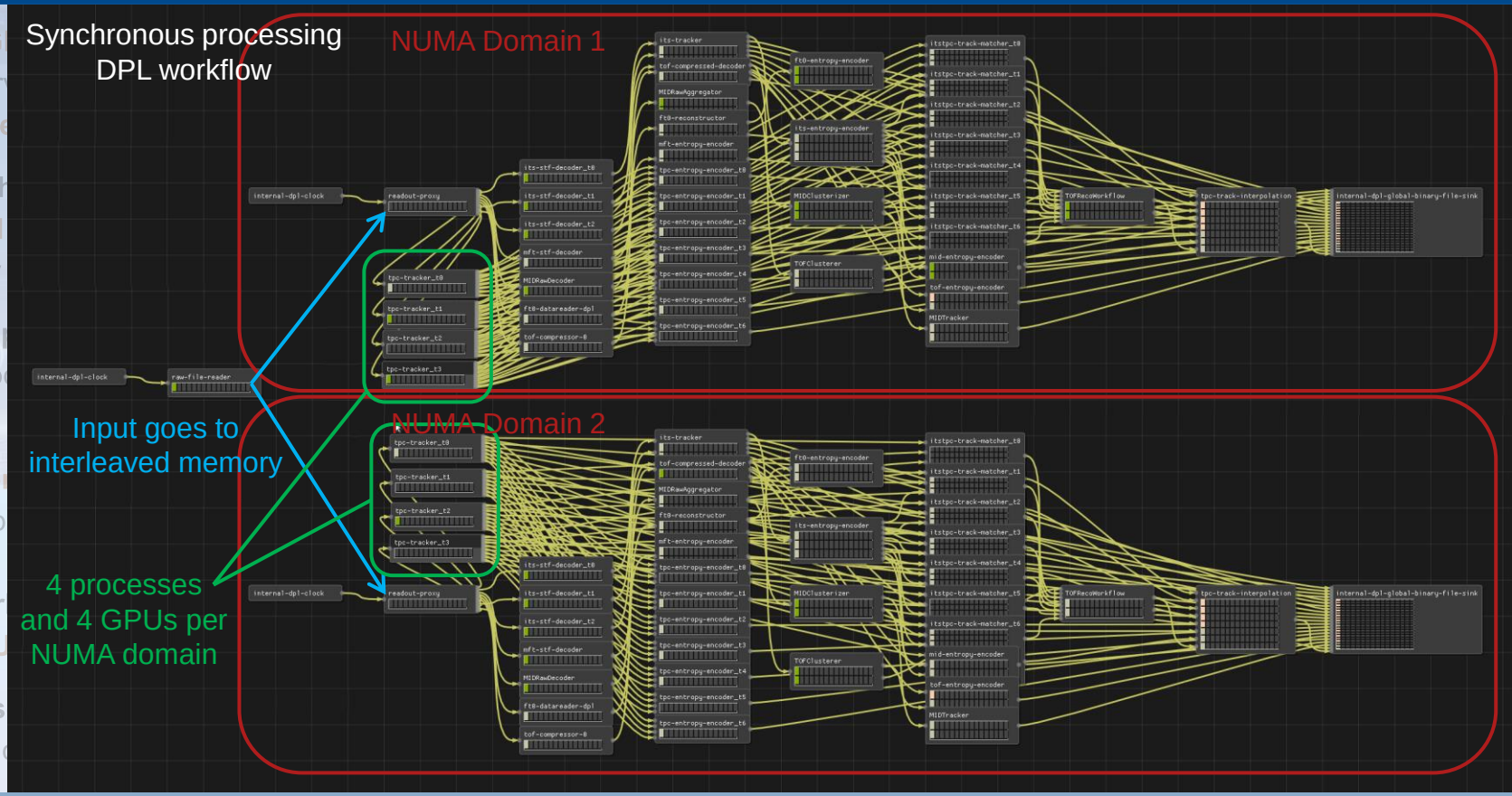
- Multiple G
  - Less ser
  - Synergi
- Splitting th
  - → 2 virtual
  - Still only
- GPUs are
  - Host pro
- GPUs can
  - With me
  - With sep
- Benchmar
  - ~50 CPU
- Selected s
  - Supermi

Synchronous processing
DPL workflow



Implementation details

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



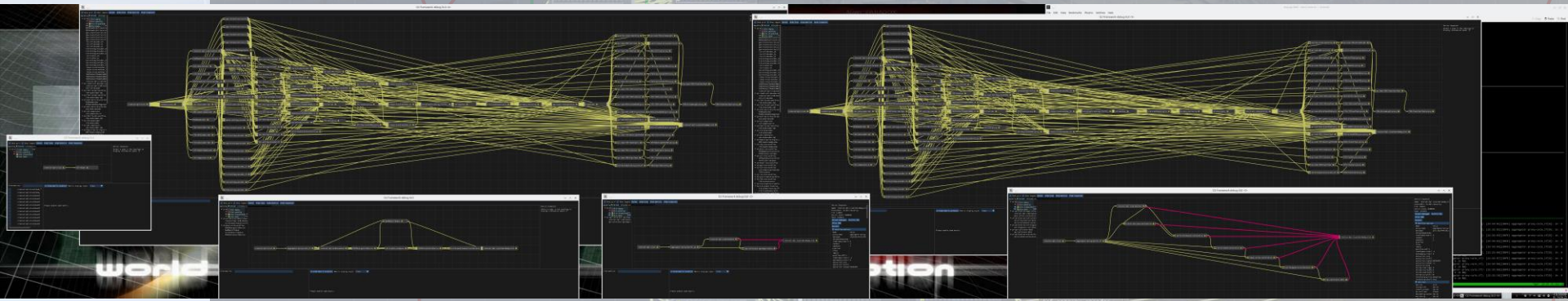
Implementation details

- Multiple GPUs in one server
- Less servers, less network
- Synergies of using the same CPU components for multi
- Splitting the node into 2 NUMA domains, running
  → 2 virtual EPNs.
- Still only 1 HCA for the input → writ
- GPUs are processing individual time
- Selected server:
  - Supermicro AS-4124GS-TNR, 8 * MI50 GPU, 2 * AMD Rome 7452 CPU (2.35 GHz), 512 GB RAM (16 * 32GB)

Synchronous processing
DPL workflow

NUMA Domain 1

To illustrate the complexity:
Full synchronous workflow including
Quality Control and Calibration

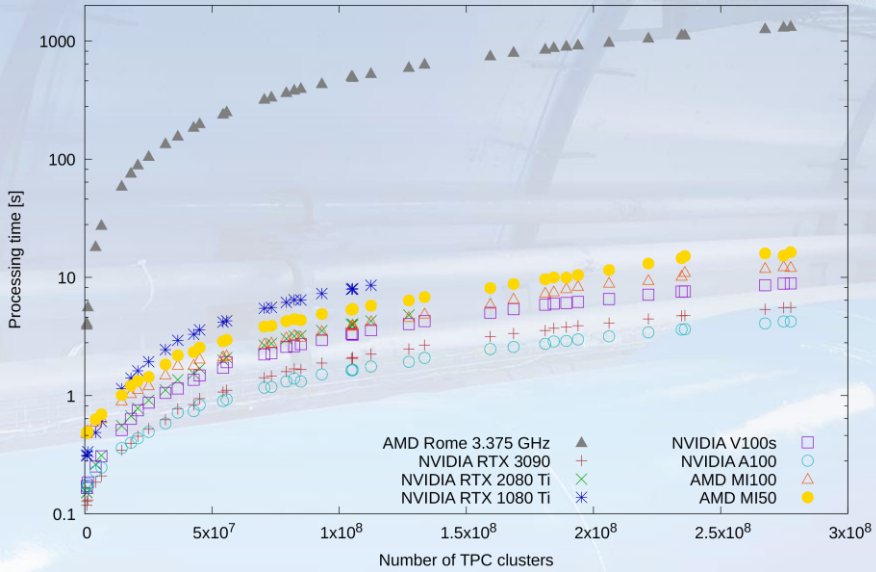


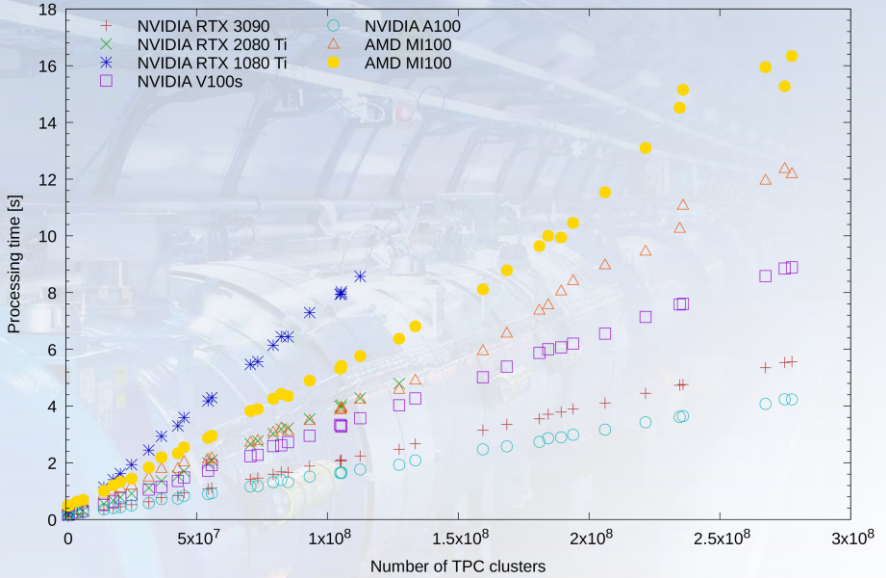
PERFORMANCE

David Rohr, drohr@cern.ch

Synchronous processing performance

- **GPU Tests with 8 MI100 performed in EPN prototype server at P2.**



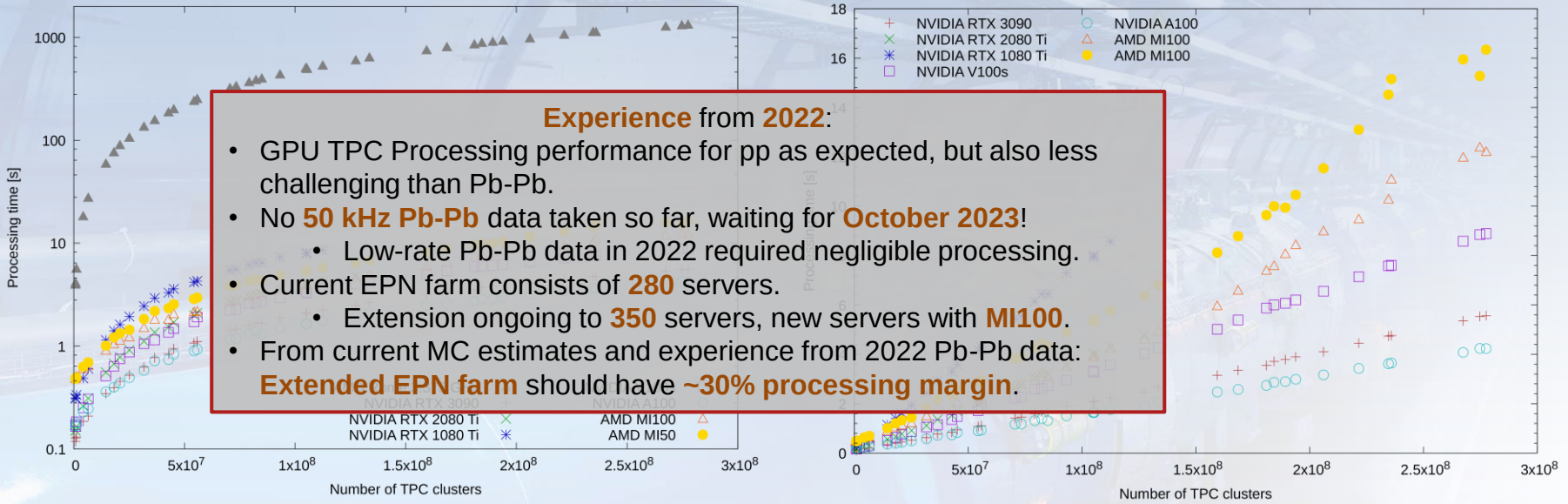


- **MI50 GPU replaces ~80 AMD Rome CPU cores in synchronous reconstruction.**
  - Includes **TPC clusterization**, which is **not optimized** for the CPU!
  - **~55 CPU cores** in **asynchronous** reconstruction (more realistic comparison).
- **Validated software with MI100 GPU, ca 35% faster.**

Without GPUs, more than 2000 64-core servers would be needed for online processing!

Synchronous processing performance

- **GPU Tests with 8 MI100 performed in EPN prototype server at P2.**



Experience from 2022:

- GPU TPC Processing performance for pp as expected, but also less challenging than Pb-Pb.
- No **50 kHz Pb-Pb** data taken so far, waiting for **October 2023**!
  - Low-rate Pb-Pb data in 2022 required negligible processing.
- Current EPN farm consists of **280** servers.
  - Extension ongoing to **350** servers, new servers with **MI100**.
- From current MC estimates and experience from 2022 Pb-Pb data: **Extended EPN farm** should have **~30% processing margin**.

- **MI50 GPU replaces ~80 AMD Rome CPU cores in synchronous reconstruction.**
  - Includes **TPC clusterization**, which is **not optimized** for the CPU!
  - **~55 CPU cores** in **asynchronous** reconstruction (more realistic comparison).
- **Validated software with MI100 GPU, ca 35% faster.**

Without GPUs, more than 2000 64-core servers would be needed for online processing!

Overview of compute time of reconstruction steps

- **The table below shows the relative compute time (linux cpu time) of the processing steps running on the processor.**

Synchronous processing
(50 kHz Pb-Pb, MC data, processing only)

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 %

Asynchronous processing
(650 kHz pp, real data, calorimeters not in run)

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 %

Overview of compute time of reconstruction steps

- **The table below shows the relative compute time (linux cpu time) of the processing steps running on the processor.**
  - Synchronous reconstruction fully dominated by the TPC (99%), no reason to offload anything else to the GPU.
  - In async reco, currently the 61.4% TPC are on the GPU, with the full optimistic scenario (full barrel tracking) it will be 79.77%.
    - 85% of EPN compute power (measured in async TPC tracking code) is in the GPUs, i.e. need to offload >= 85% to obtain 100% GPU usage.

Synchronous processing
(50 kHz Pb-Pb, MC data, processing only)

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 %

Asynchronous processing
(650 kHz pp, real data, calorimeters not in run)

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 %

Running on GPU in baseline scenario

Running on GPU in optimistic scenario

Overview of compute time of reconstruction steps

- **Async reco GPU speedup on the EPN:**
  - The **speed of light** is **~6.5x** speedup, since **85%** of the **compute power** is in the **GPU** (reduce the CPU time by 85%, more becomes GPU-bound).
    - Only in case everything scales as well as TPC processing.
    - Even then cannot be reached since GPU processing needs CPU resources.
  - **Today**, offloading the ~**60%** of the async to the GPU should yield a **speedup** around **2.5x**.
    - We remove 60% of the CPU time, while we are still CPU-bound, but we have some overhead CPU resources for driving the 8 GPUs.
  - In the **optimistic scenario**, by offloading **80%** we might get close to **5x**.
    - Still a bit away from the speed of light.

Asynchronous processing
(650 kHz pp, real data, calorimeters not in run)

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 %

Running on GPU in baseline scenario

Running on GPU in optimistic scenario

Real speedup in asynchronous reconstruction

- **For asynchronous 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**: synchronous and asynchronous.
    - Unused nodes in the synchronous pool are moved to the asynchronous pool.
    - As needed for data-taking, nodes are moved to the synchronous pool with lead time to let the current jobs finished.
      - If needed immediately, GRID jobs are killed and nodes moved immediately.

Real speedup in asynchronous reconstruction

- **For asynchronous 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**: synchronous and asynchronous.
    - Unused nodes in the synchronous pool are moved to the asynchronous pool.
    - As needed for data-taking, nodes are moved to the synchronous 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 (1 instance)	Time per TF (full server)
CPU 8 core	76.91s	4.81s
CPU 16 core	34.18s	**4.27s**
1 GPU + 16 CPU cores	14.60s	1.83s
1 NUMA domain (4 GPUs + 64 cores)	3.5s	**1.70s**

Factor 2.51
Matches expected factor 2.5

Real speedup in asynchronous reconstruction

- **For asynchronous 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**: synchronous and asynchronous.
    - Unused nodes in the synchronous pool are moved to the asynchronous pool.
    - As needed for data-taking, nodes are moved to the synchronous 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 (1 instance)	Time per TF (full server)
CPU 8 core	76.91s	4.81s
CPU 16 core	34.18s	**4.27s**
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 (Also resembles most the synchronous processing configuration)

Factor 2.51
Matches expected factor 2.5

Conclusions

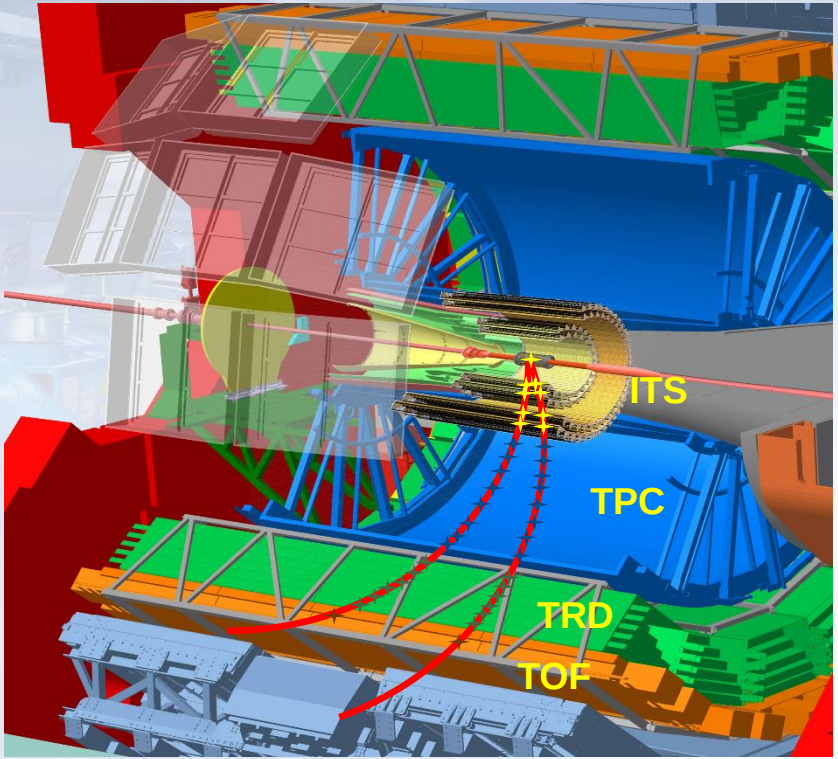
- **ALICE employs GPUs heavily to speed up online and offline processing.**
  - **99%** of **synchronous reconstruction** on the **GPU** (no reason at all to port the rest).
  - Today ~**60%** of full **asynchronous processing** (for 650 kHz pp) on **GPU** (if offline jobs on the EPN farm).
    - Will increase to **80%** with full barrel tracking (**optimistic scenario**).
    - GPU represent **85%** of EPN compute power, no reason to offload more than 85% (**speed of light**).
- **Synchronous processing successful in 2021 - 2023.**
  - Pb-Pb 2022 not really stressful for processing, since no 50 kHz Pb-Pb.
    - 50 kHz Pb-Pb processing validated with data replay of MC data (~ 30 % margin).
  - Performed online pp processing with additional processing steps up to 2.6 MHz inelastic interaction rate.
- **Asynchronous reconstruction has started, processing the TPC reconstruction on the GPUs in the EPN farm, and in CPU-only style on the CERN GRID site.**
  - EPN nodes are 2.51x faster when using GPUs.

BACKUP

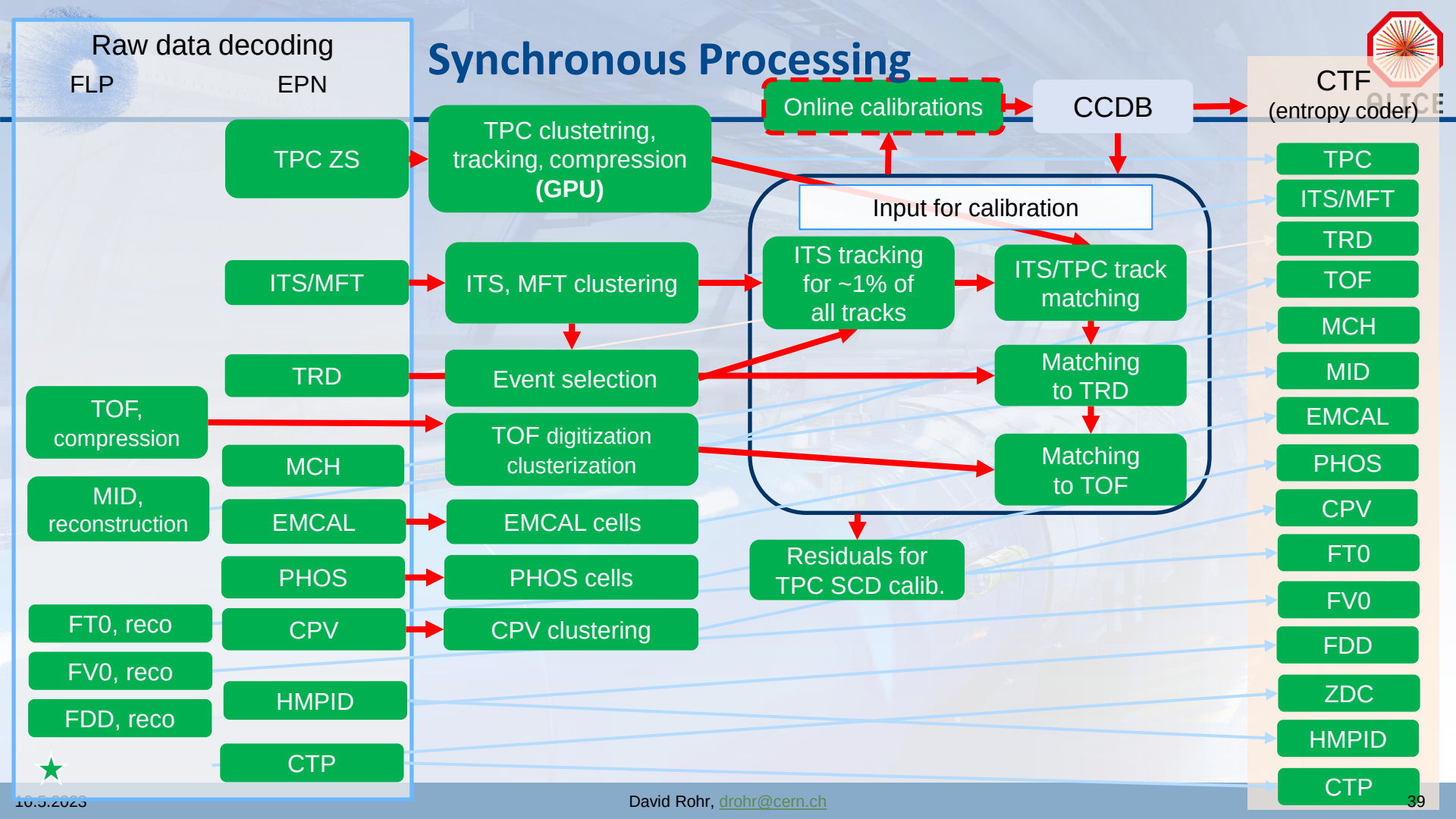
The ALICE detector in Run 3

- **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 is **~2.5 ms** (32 LHC orbits)

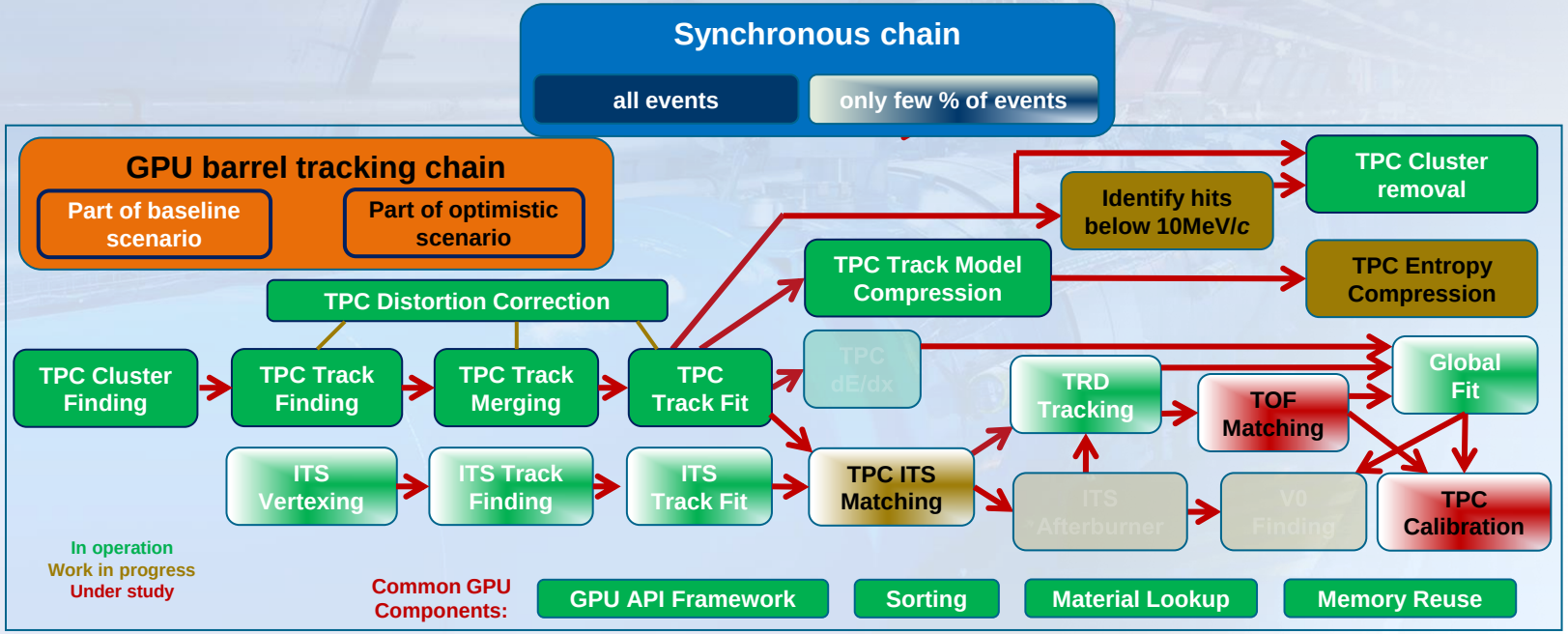


Synchronous Processing

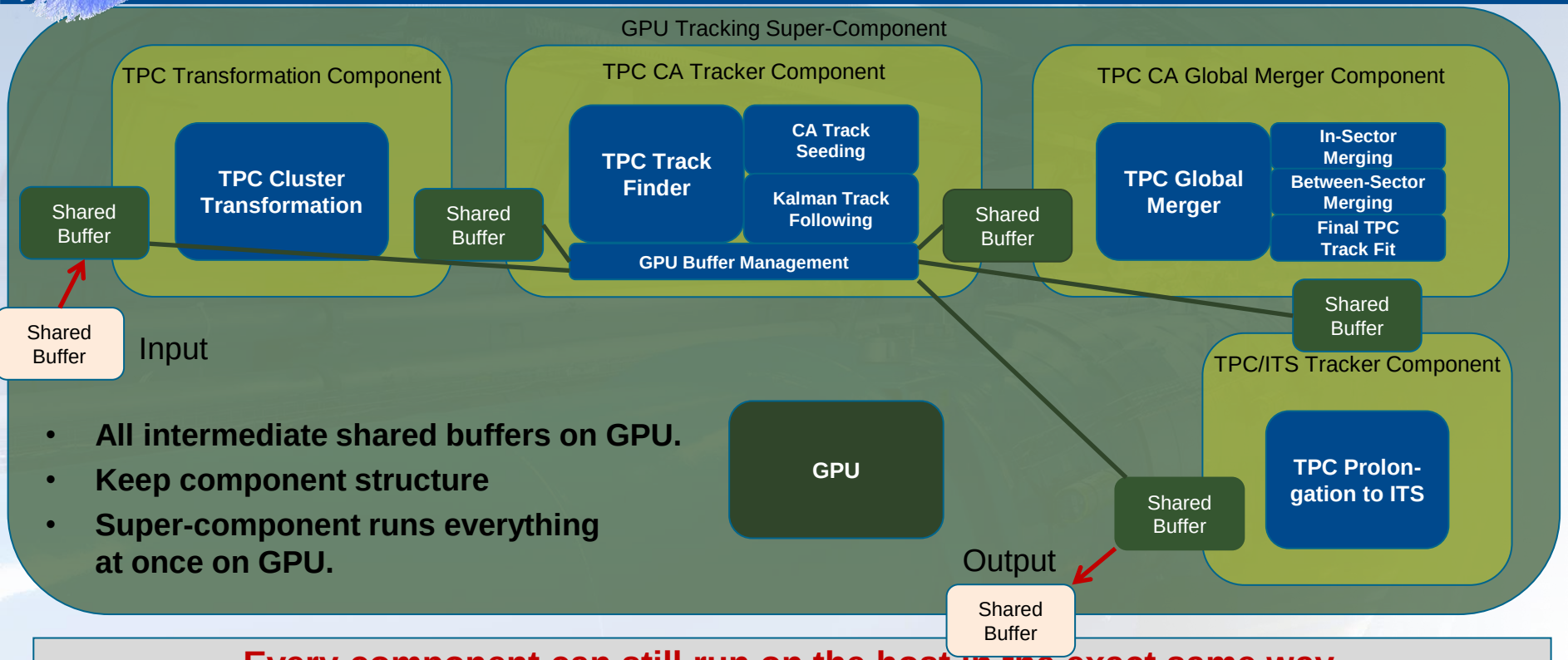


Central barrel global tracking chain

- **TPC synchronous processing almost fully on the GPU.**
  - 2 optional parts still being investigated for sync. reco on GPU: TPC entropy encoding / Looper identification < 10 MeV.



Modular GPU code



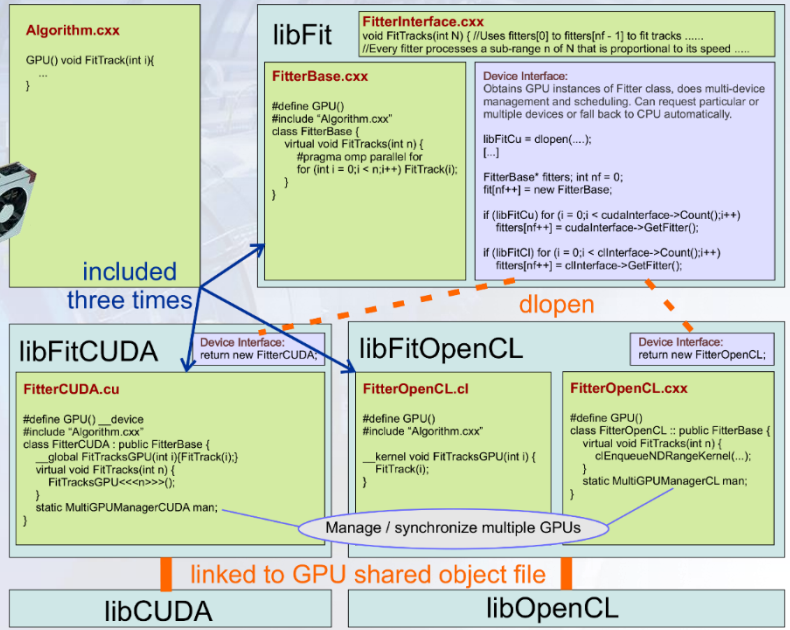
- **All intermediate shared buffers on GPU.**
- **Keep component structure**
- **Super-component runs everything at once on GPU.**

Every component can still run on the host in the exact same way.
Shared buffers either in host memory or in GPU memory.

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

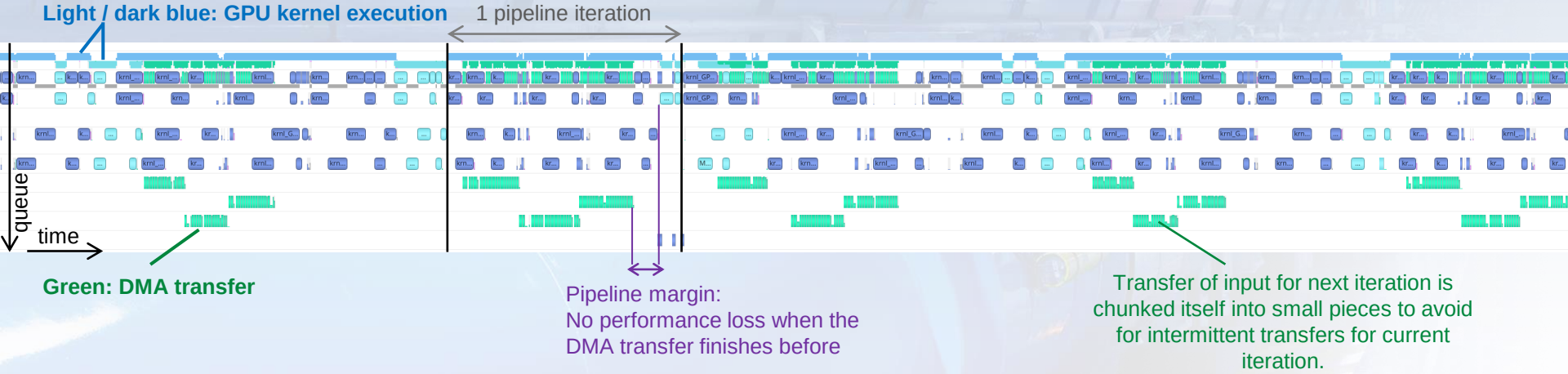
- **Screening different platforms for best price / performance.**
  (including some non-competitive platforms for cross-checks and validation.)
  - **CPUs** (**AMD Zen**, **Intel Skylake**)
    C++ backend with **OpenMP**, AMD **OCL**
  - **AMD GPUs**
    (**S9000** with **OpenCL 1.2**, **MI50** / **Radeon 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)



Memory allocation / Pipelined processing

- **Custom allocator: grabs all GPU memory, gives out chunks manually, memory will be reused when possible.**
  - Classically: reuse memory between events.
    - Single events too small for GPU → Process time frames.
  - ALICE reuses memory between different algorithms in a TF, possibly between chunks of collisions in a TF.

- **Zoomed-in plot of TPC Clusterization stage** (part with **largest DMA transfers** → most difficult to hide in pipeline).



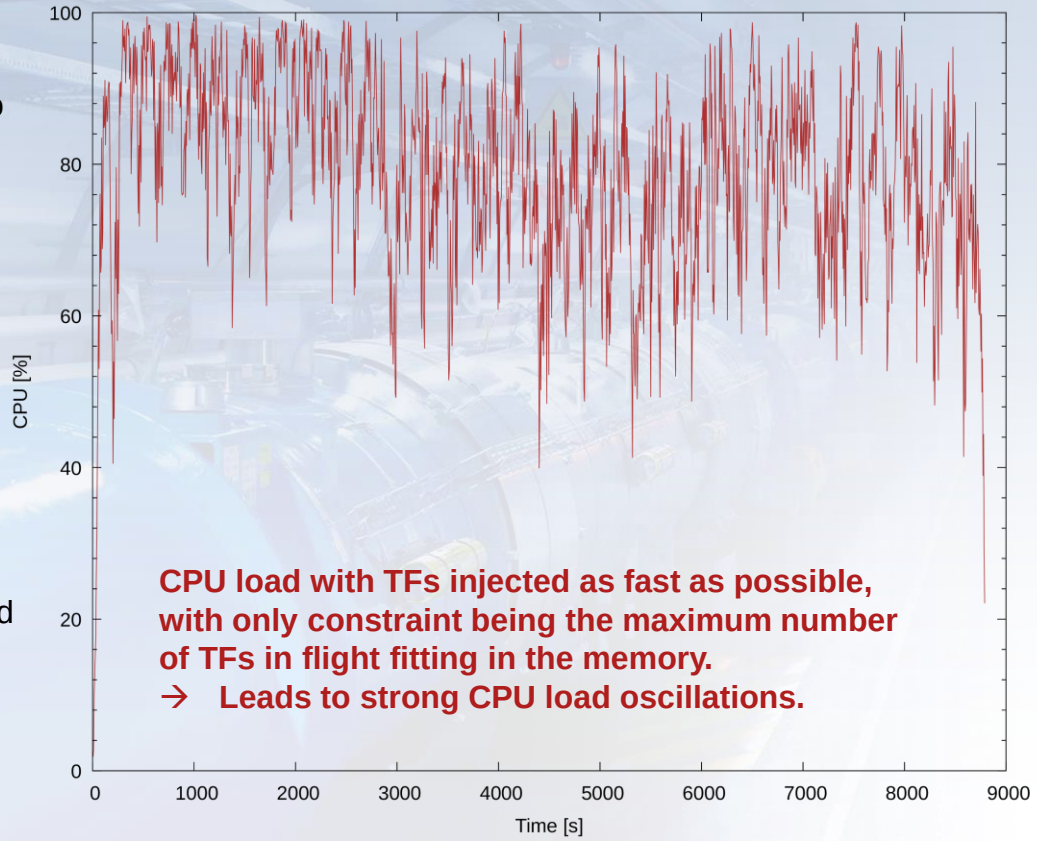
- **Full profile** of 3 time frames: **100% GPU utilization** with kernel execution, **No performance loss from data transfer**!

Time frame scheduling sync vs. async

- **In synchronous processing:** the rate is defined from the data taking: We get **351 TFs per second.**
  - The EPN farm must handle that rate, and we want to have some CPU capacity margin.
- **In asynchronous processing:** we want to process TFs **as fast as possible**, and ideally reach **100% CPU load**.

- We need many TFs in flight, to exploit the full parallelism with DPL and use all CPU cores via DPL pipelines.
- The available memory limits the maximum number of TFs in flight.

- A constant TF publishing rate is ideal to spread the load horizontally and vertically in the processing graph.
- Injecting TFs into the chain with unstable rate leads to oscillations in the processing.



CPU load with TFs injected as fast as possible, with only constraint being the maximum number of TFs in flight fitting in the memory.
→ Leads to strong CPU load oscillations.

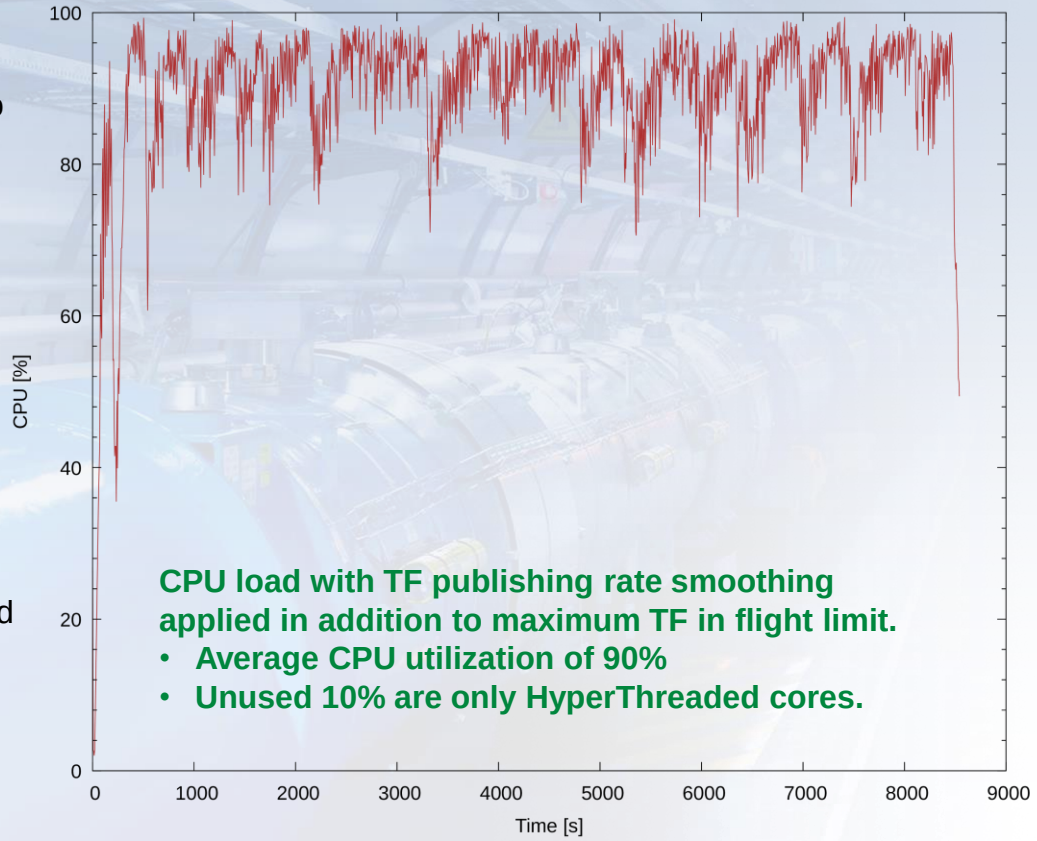
Real speedup in asynchronous reconstruction

- **In synchronous processing:** the rate is defined from the data taking: We get **351 TFs per second.**
  - The EPN farm must handle that rate, and we want to have some CPU capacity margin.
- **In asynchronous processing:** we want to process TFs **as fast as possible**, and ideally reach **100% CPU load**.

- We need many TFs in flight, to exploit the full parallelism with DPL and use all CPU cores via DPL pipelines.
- The available memory limits the maximum number of TFs in flight.

- A constant TF publishing rate is ideal to spread the load horizontally and vertically in the processing graph.
- Injecting TFs into the chain with unstable rate leads to oscillations in the processing.

→ **Heuristic to smoothen TF publishing rate solves the problem.**



CPU load with TF publishing rate smoothing applied in addition to maximum TF in flight limit.
- **Average CPU utilization of 90%**
- **Unused 10% are only HyperThreaded cores.**

Real speedup in asynchronous reconstruction

- **Overhead at begining / end of job:**
  - Constant overhead at start / stop of processing: **149 s (1.8%)**
  - → Negligible compared to job runtime (benchmark job was 8491 **s**, could be extended to >10h)
  - Additional time needed for AOD checking / merging: **238s** (**2.8%**, CPU only Postprocessing to speed up analysis)
  - Time lost at processing dip at the beginning during condition fetching / initialization: **32s** (**0.4%**)
- **Some interesting performance comparisons:**
  - 1 GPU workflow, running **isolated** on a node v.s. running **8 times** in parallel on a node: ??% faster (HyperThreading).
  - 1 NUMA workflow, with **rate smoothing** v.s. **without rate smoothing**: **11.6%** faster.
- **Benefits of 2 * 1 NUMA domain workflow over 8 * 1 GPU workflow:**
  - Not all CPU processes duplicated → fewer processes, and significantly less memory consumption (~ 100 GB difference).
  - Share the CPU processes in DPL workflow → more CPU capacity compensates load fluctuations, less context switches.

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

Factor 2.51
Matches expected factor 2.5