GPU-accelerated online and offline processing at ALICE

David Rohr for the ALICE Collaboration

SONIC: Heterogeneous Computing for Science Workshop

1.3.2024

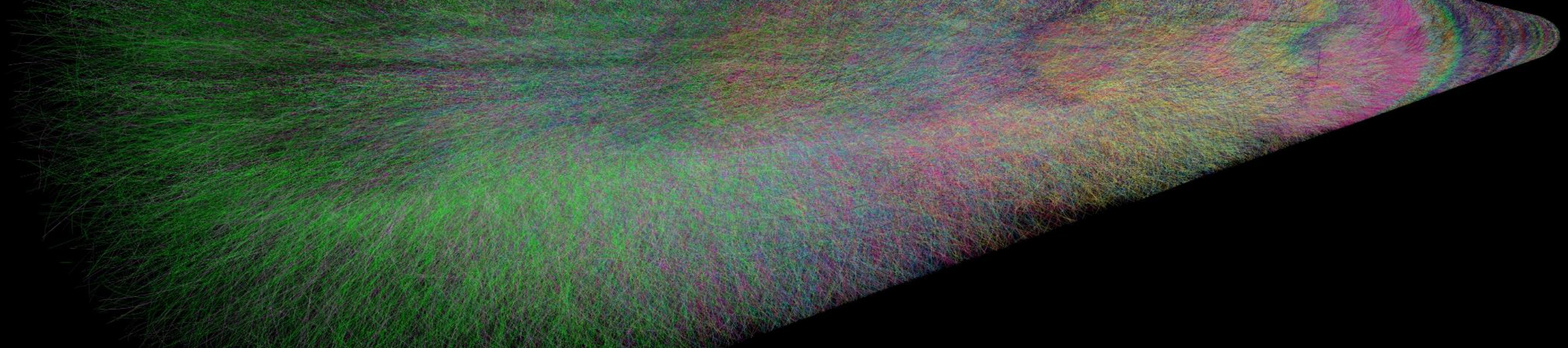
drohr@cern.ch

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

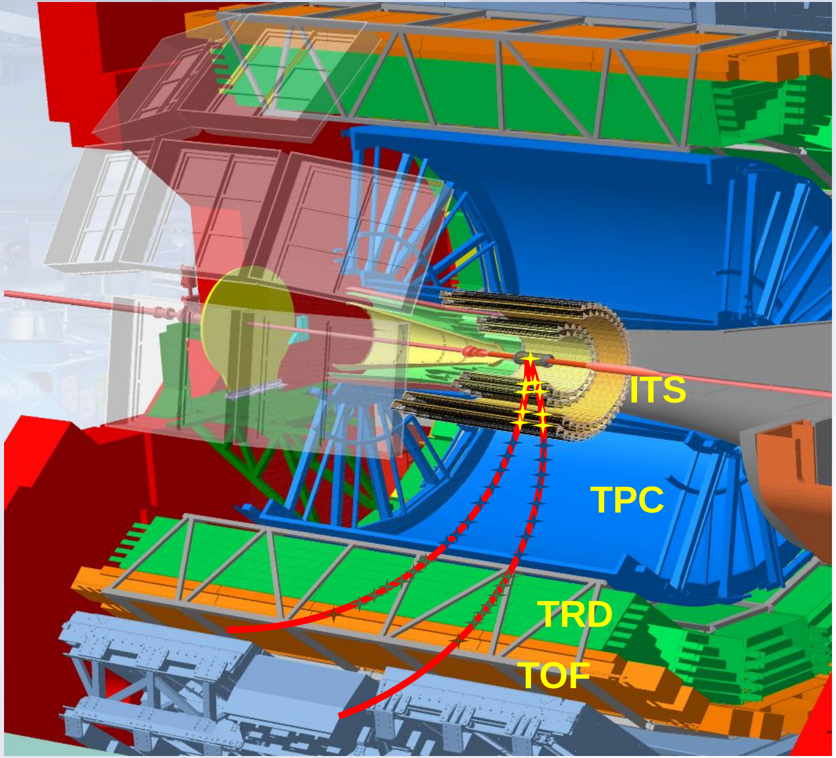
→ Use GPUs to speed up online (and offline) processing



- 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.

The ALICE detector in Run 3

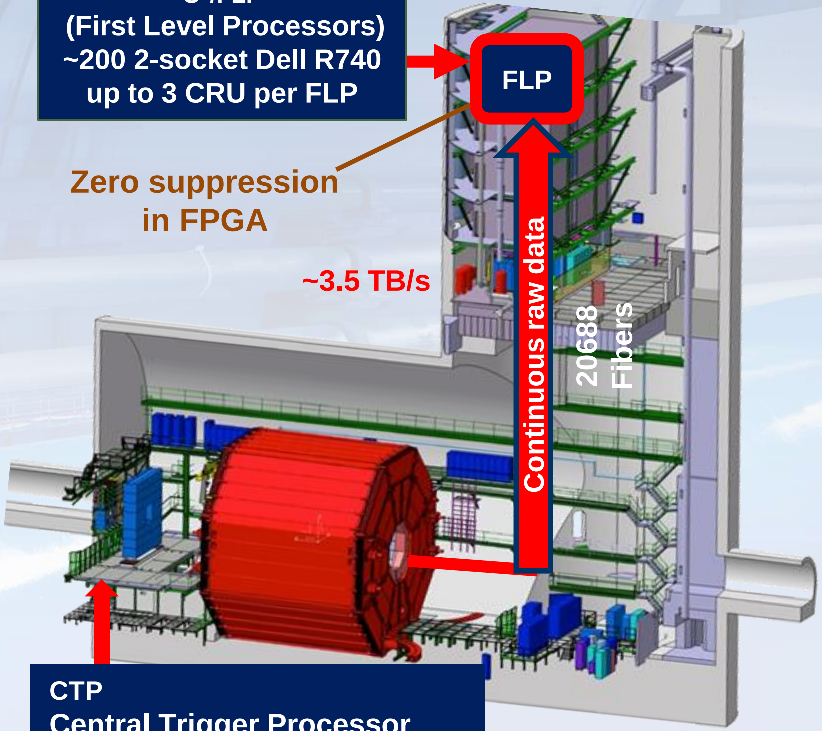
- **ALICE uses mainly 3 detectors for barrel tracking: ITS, TPC, TRD + (TOF)**
  - **7 layers ITS** (Inner Tracking System – silicon tracker)
  - **152 pad rows TPC** (Time Projection Chamber)
  - **6 layers TRD** (Transition Radiation Detector)
  - **1 layer TOF** (Time Of Flight Detector)



- **ALICE performs continuous readout.**
- **Native data unit is a time frame: all data from a configurable period of data up to 256 LHC orbits.**
  - Default was ~11 ms (128 LHC orbits) before 2023.
  - Current default is **~2.8 ms** (32 LHC orbits)

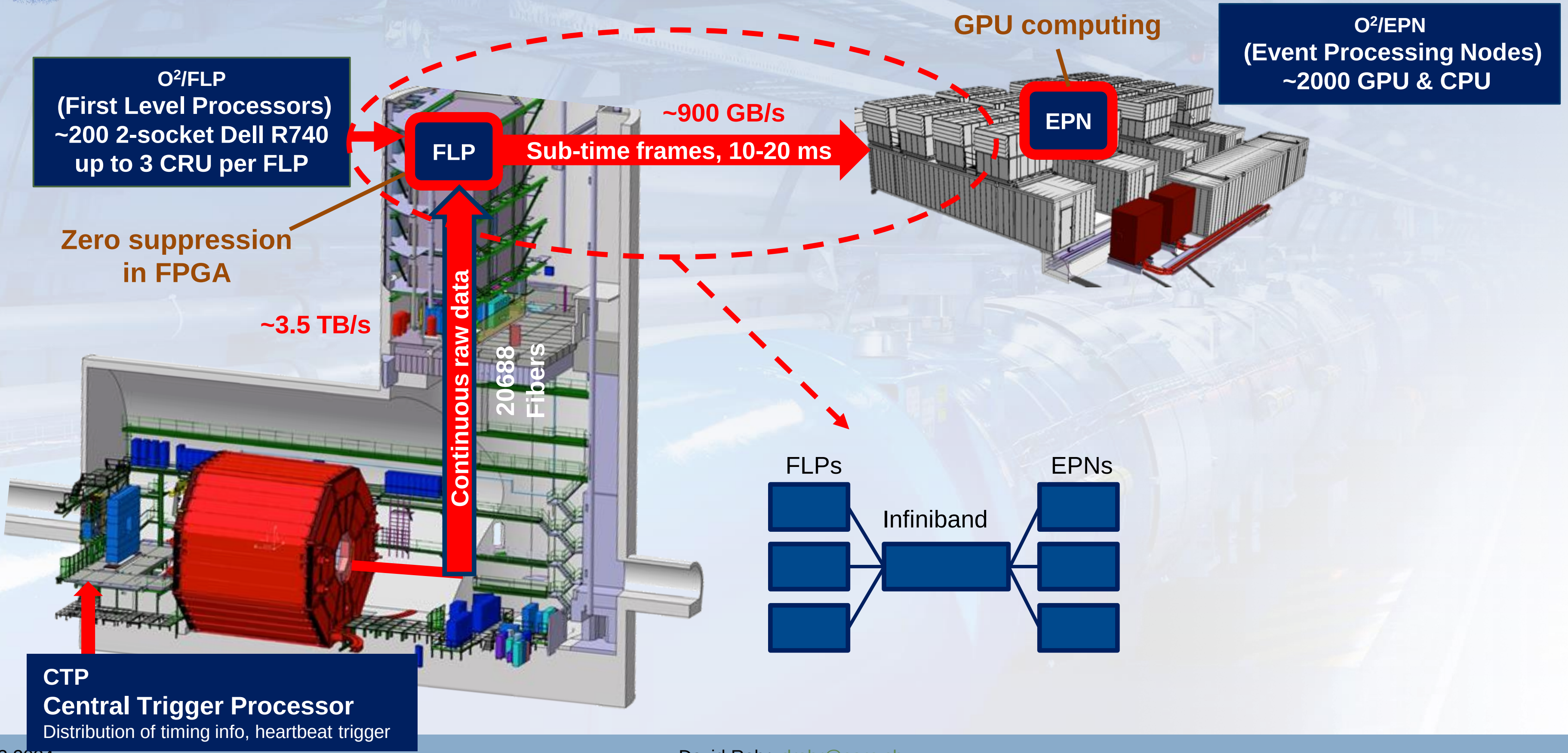
ALICE Raw Data Flow in Run 3

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

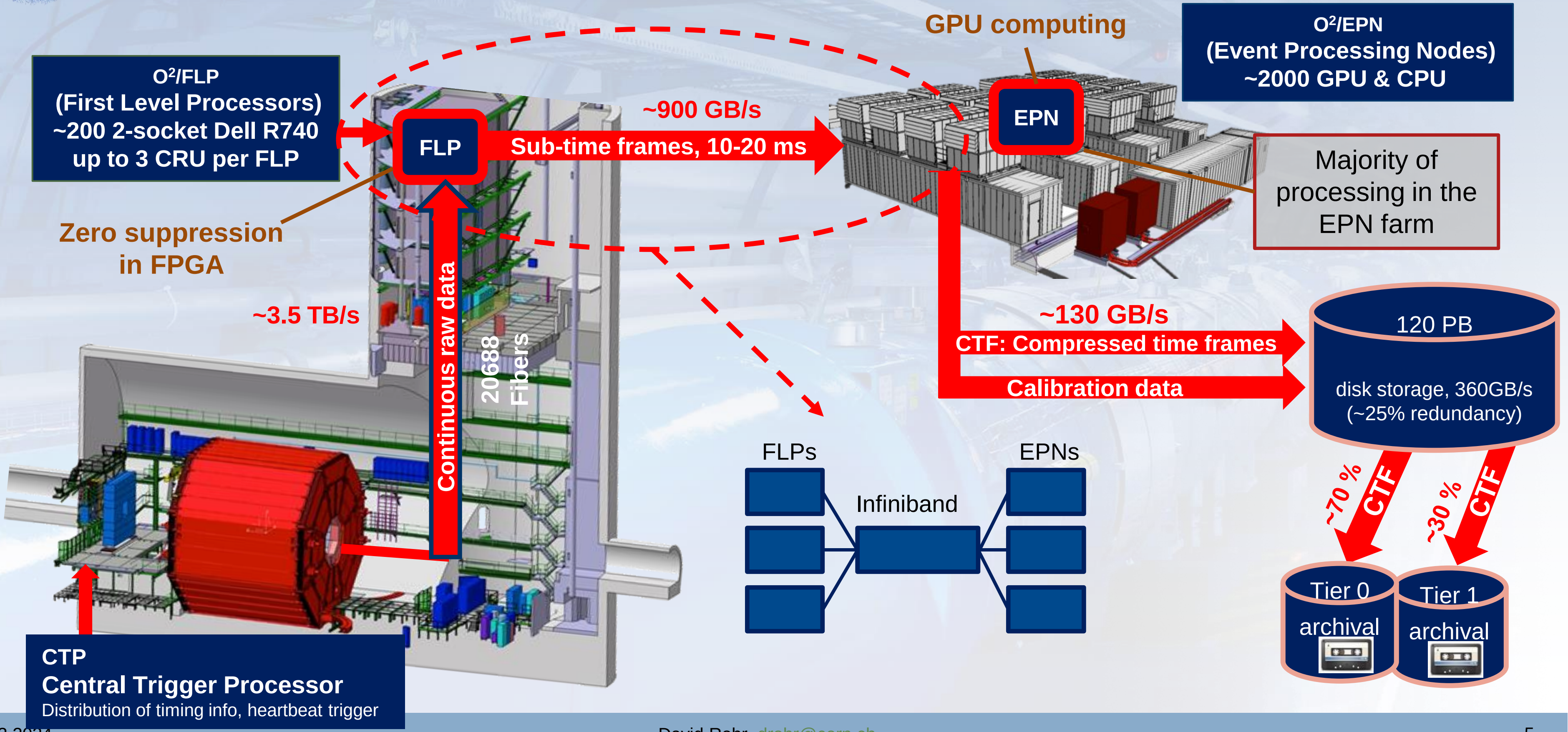


CTP
Central Trigger Processor
Distribution of timing info, heartbeat trigger

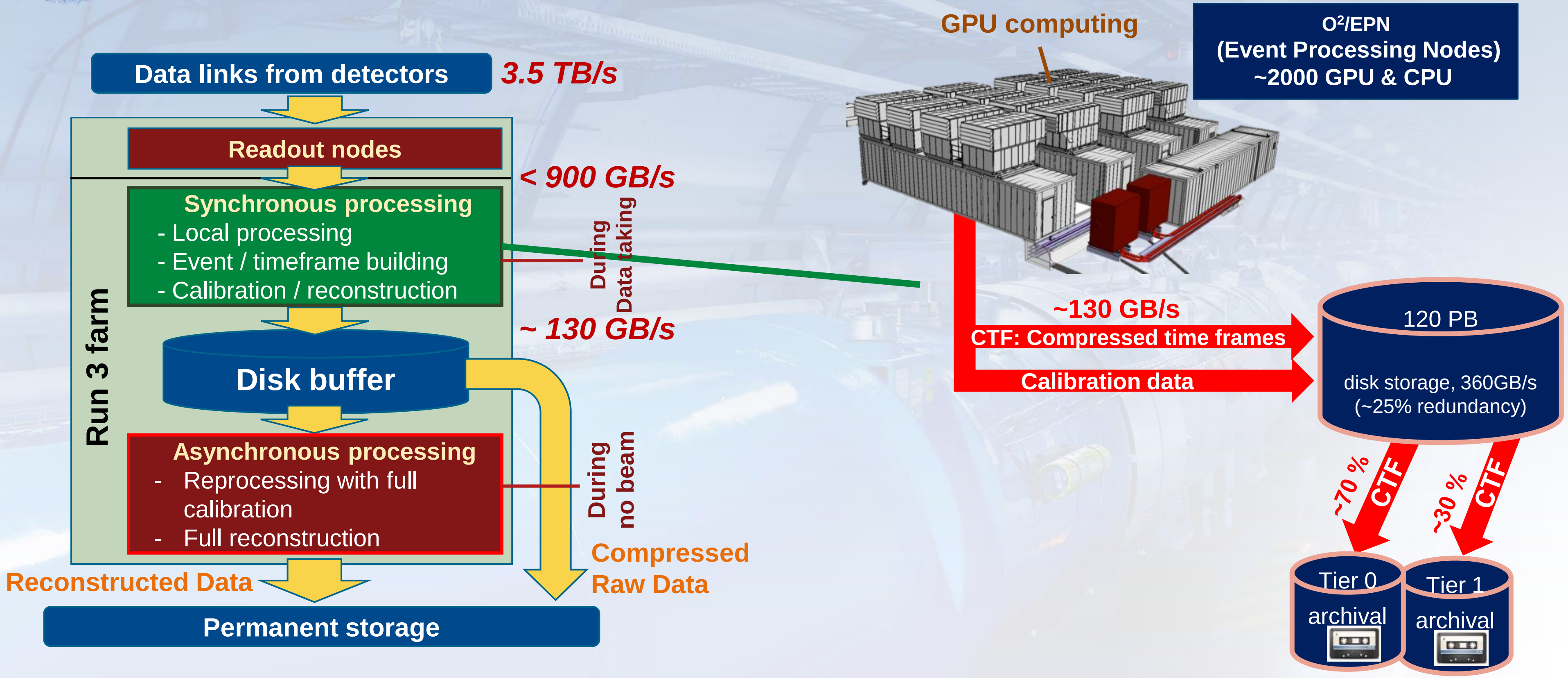
ALICE Raw Data Flow in Run 3



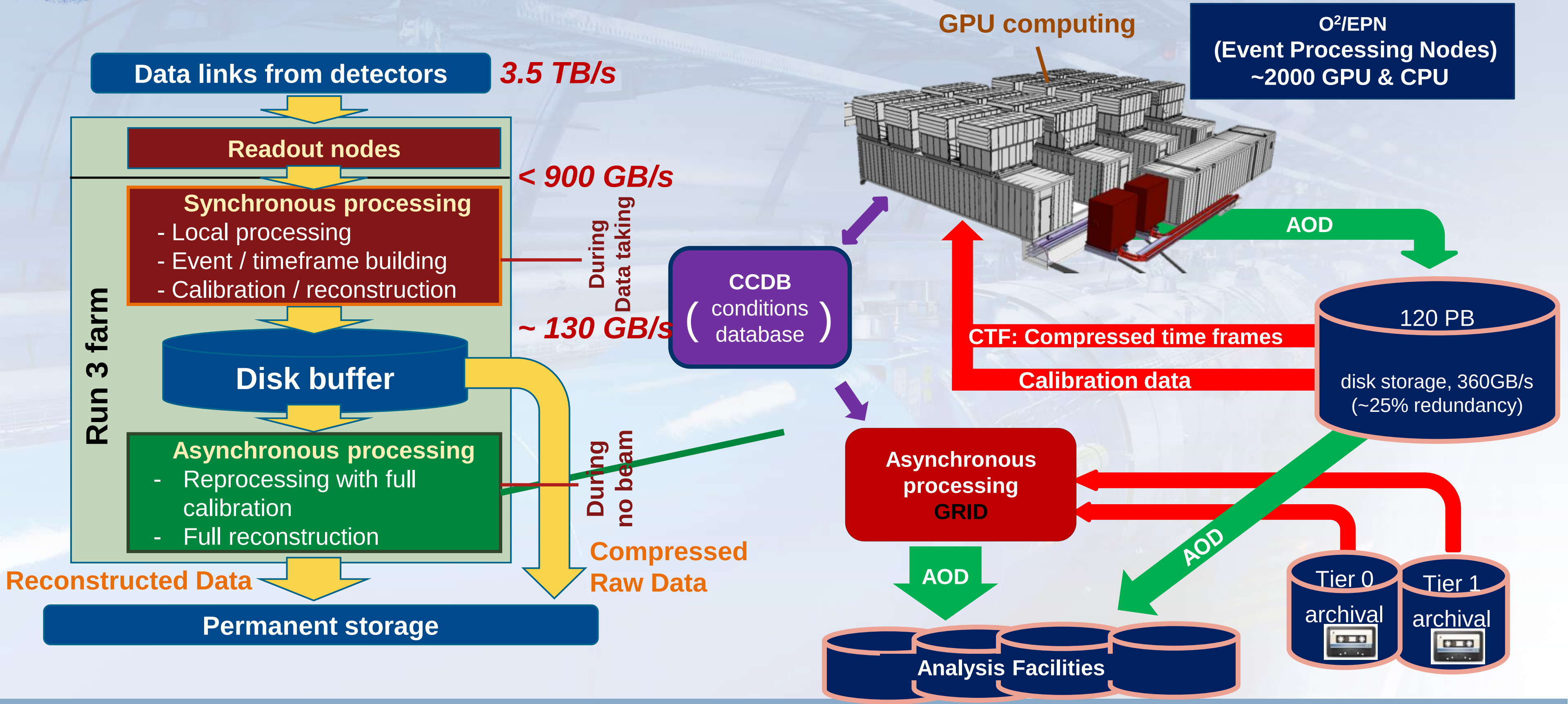
ALICE Raw Data Flow in Run 3



Synchronous and Asynchronous Processing



Synchronous and Asynchronous Processing



David Rohr, drohr@cern.ch

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 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!

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.

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 %

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

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

ALICE

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.

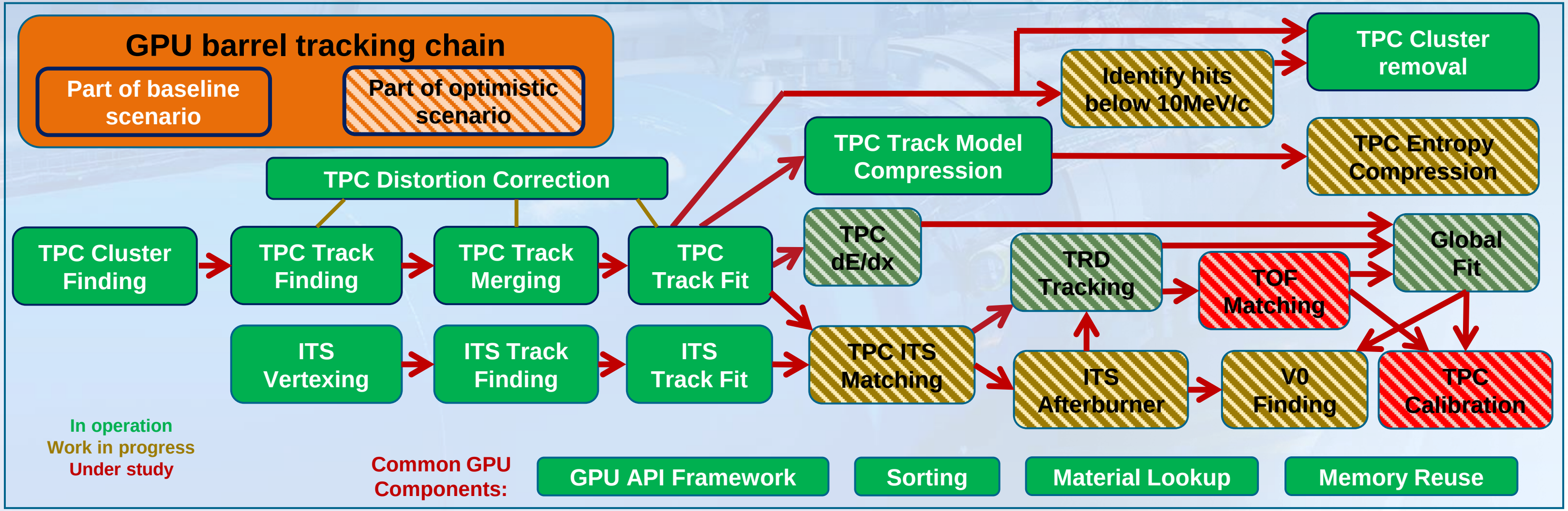
- **GPUs** well suited for **TPC** reco (from Run 1 and 2 experience).
- **GPUs** provide the **required compute power**.
  - Time frame concepts yields large enough GPU data chunks.
- Following up **2 scenarios** for EPN GPU processing:

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

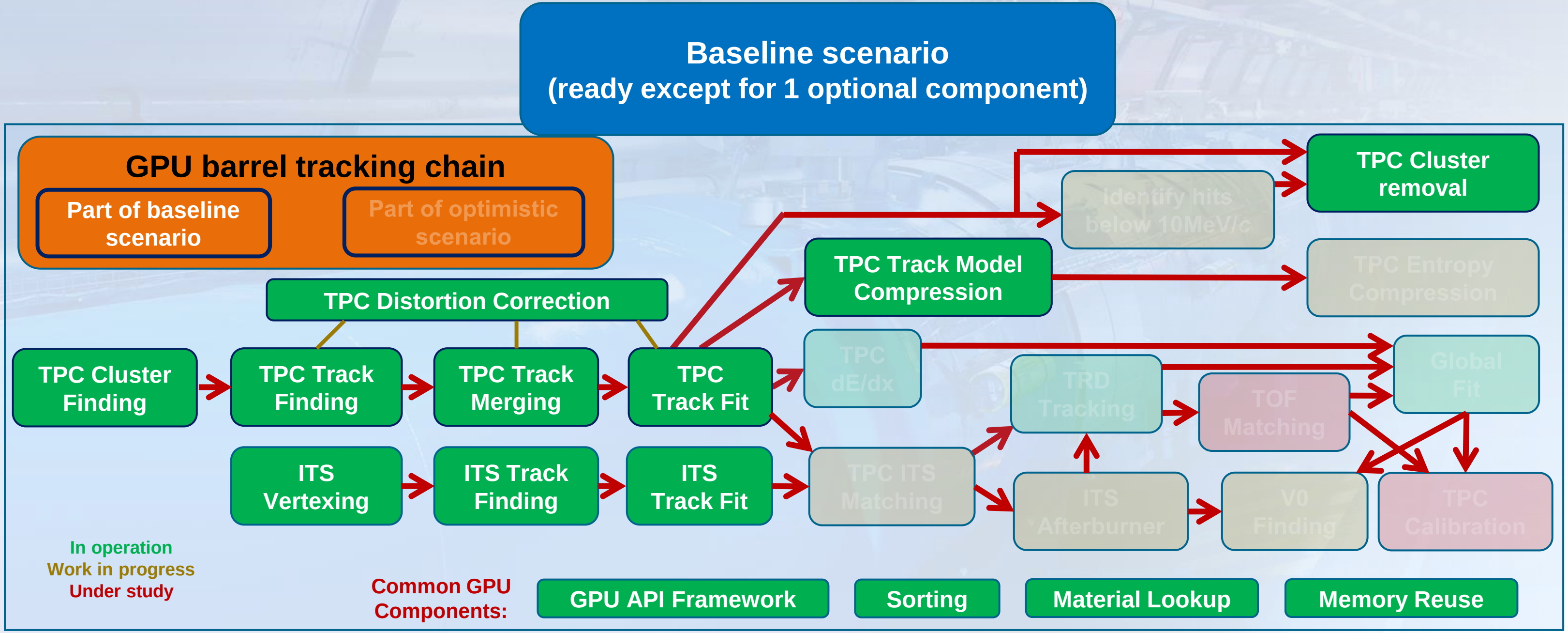
Central barrel global tracking chain

- **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.



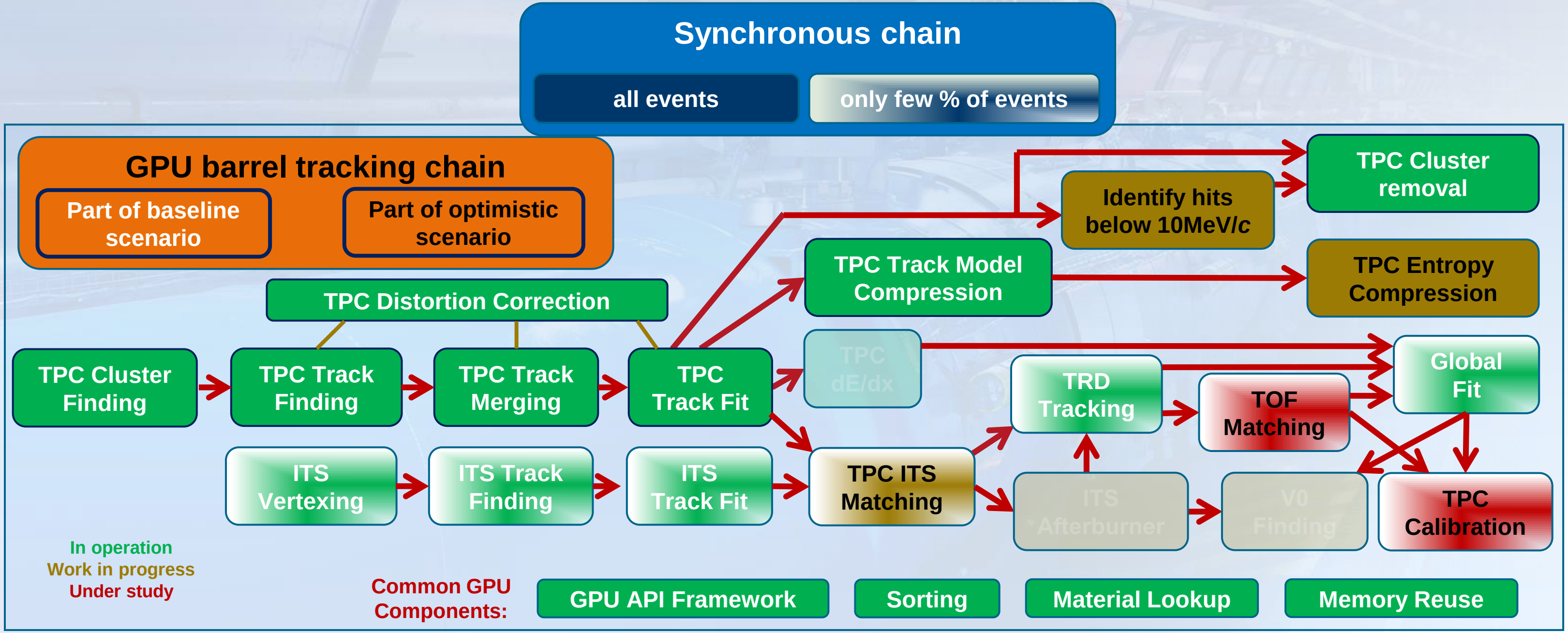
Central barrel global tracking chain

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



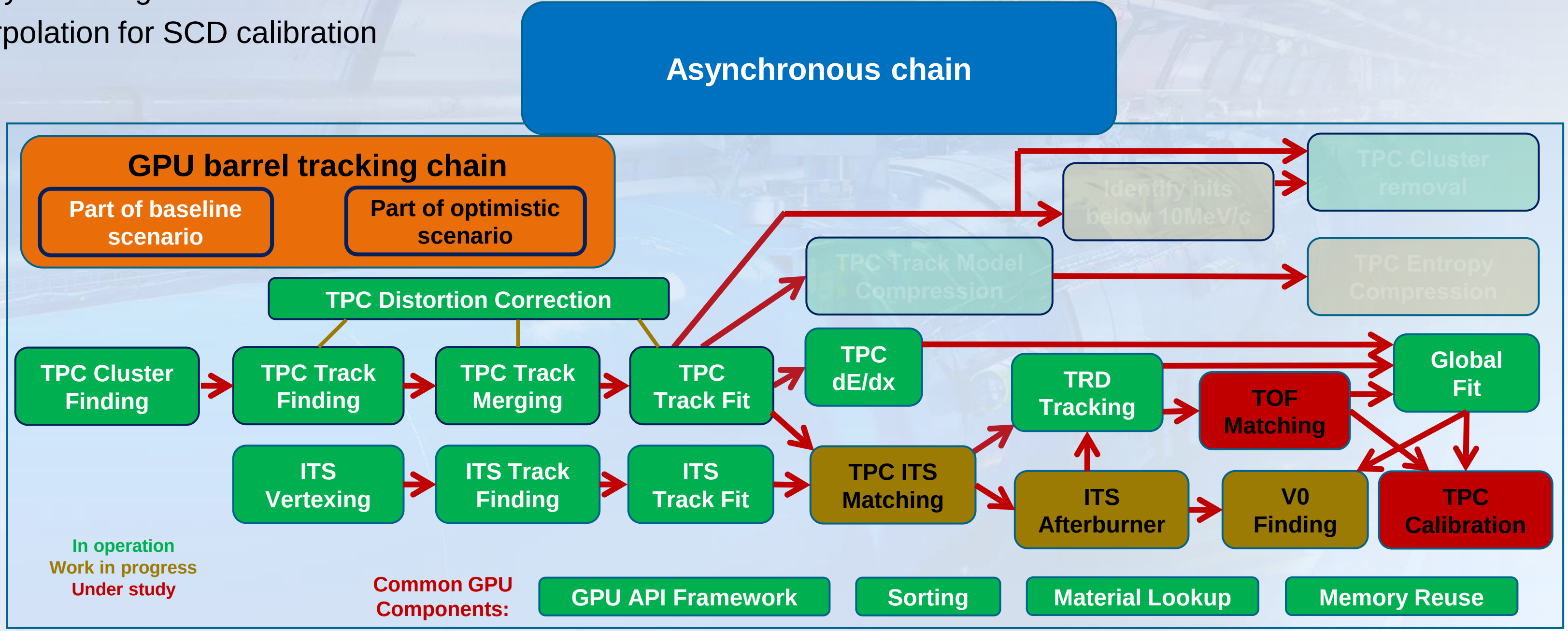
Central barrel global tracking chain

- **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.



Central barrel global tracking chain

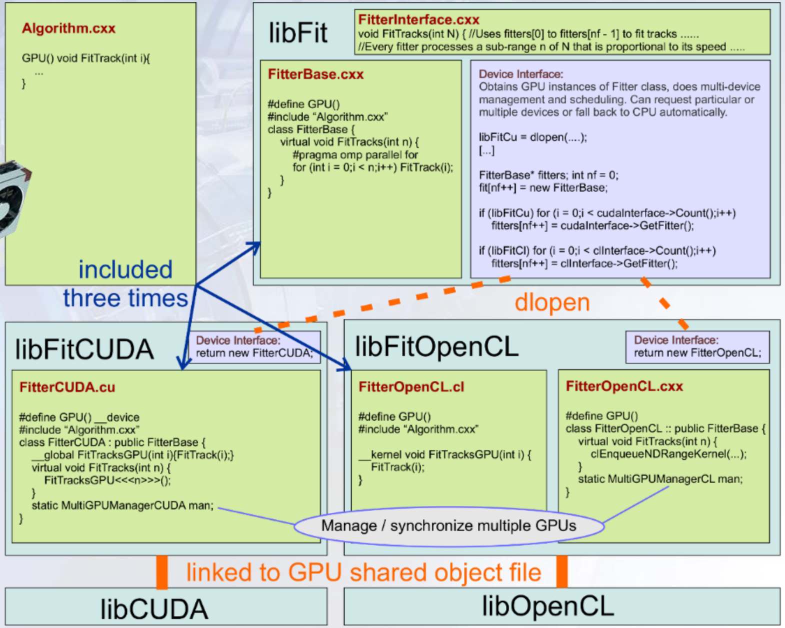
- **Several steps missing in asynchronous reconstruction:**
  - Matching to ITS
  - Matching to TOF
  - Secondary vertexing
  - TPC interpolation for SCD calibration



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)



Pipelined processing

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

Light / dark blue: GPU kernel execution

1 pipeline iteration

Green: DMA transfer

Pipeline margin:
No performance loss when the DMA transfer finishes before

Transfer of input for next iteration is chunked itself into small pieces to avoid for intermittent transfers for current iteration.

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

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 be generic C++ and not depend on one particular vendor or API. (O2 supports CUDA, HIP, OpenCL)**
   - No usage of special features that are not portable.
3. **GPU usage should be optional and transparent: running O2 should not require any vendor libraries installed.**
   - All GPU code is contained in plugins, with a common interface.
   - Even multiple plugins (GPU backends) can run on the same node.
4. **Minimize time spent 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 processed by the GPU must be large enough to exploit the full parallelism.**
   - Fulfilled by design with TFs containing > 100 collisions.
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.

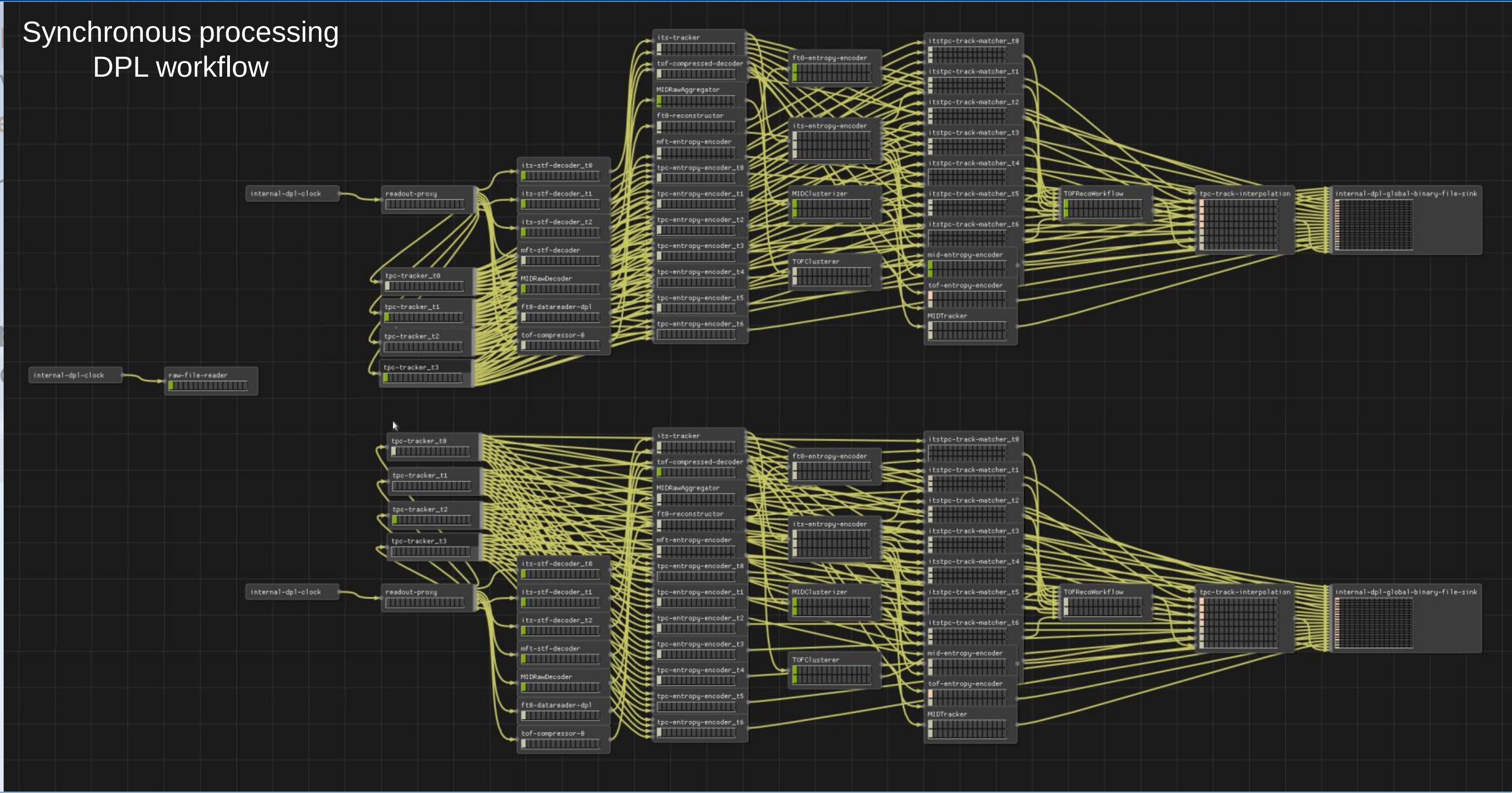
Implementation details

- **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.

Implementation details

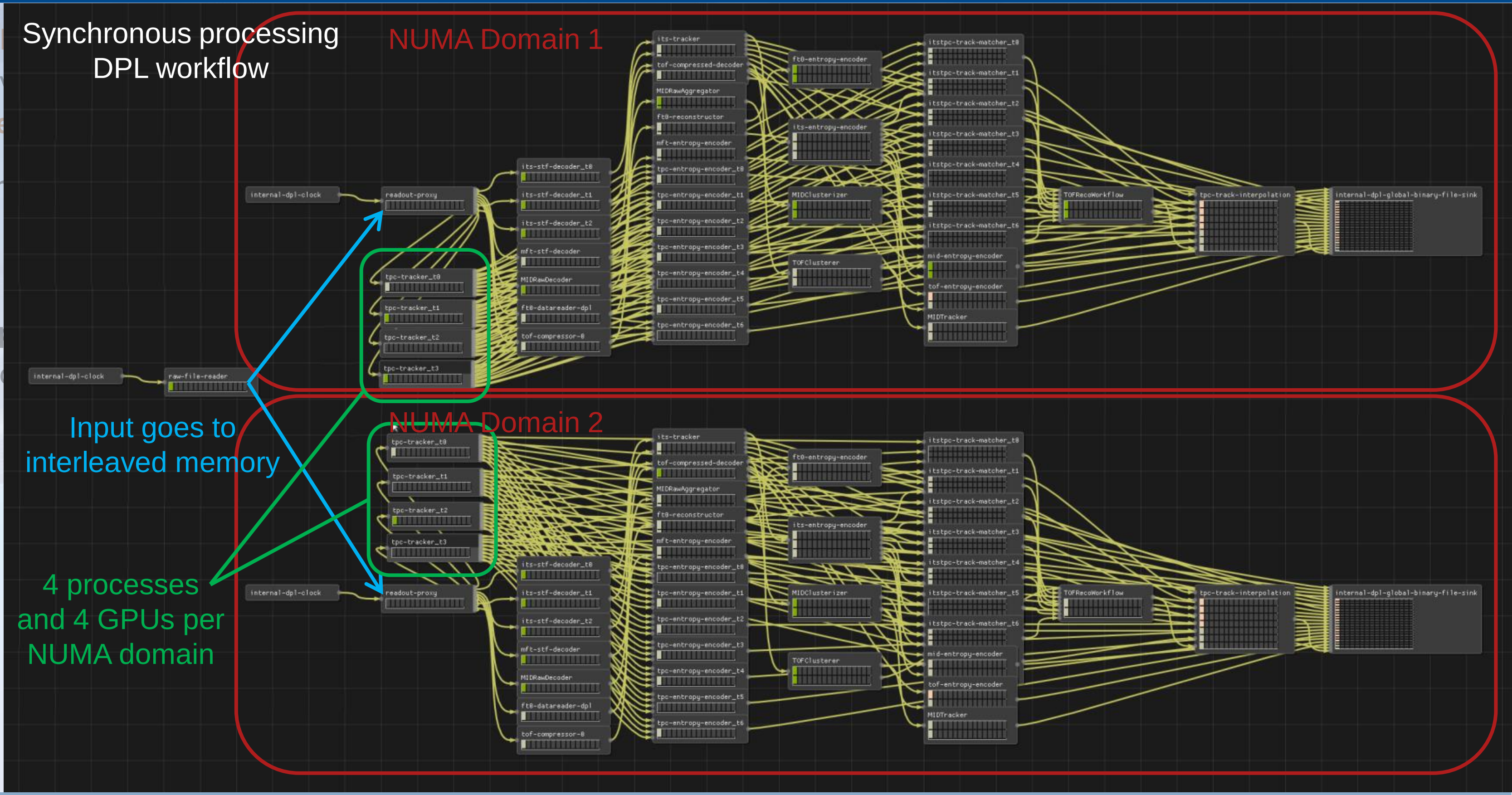
- Multiple GI
  - Less ser
  - Synergie
- Splitting th
  → 2 virtual
  - Still only
- GPUs are
  - Host pro

Synchronous processing
DPL workflow



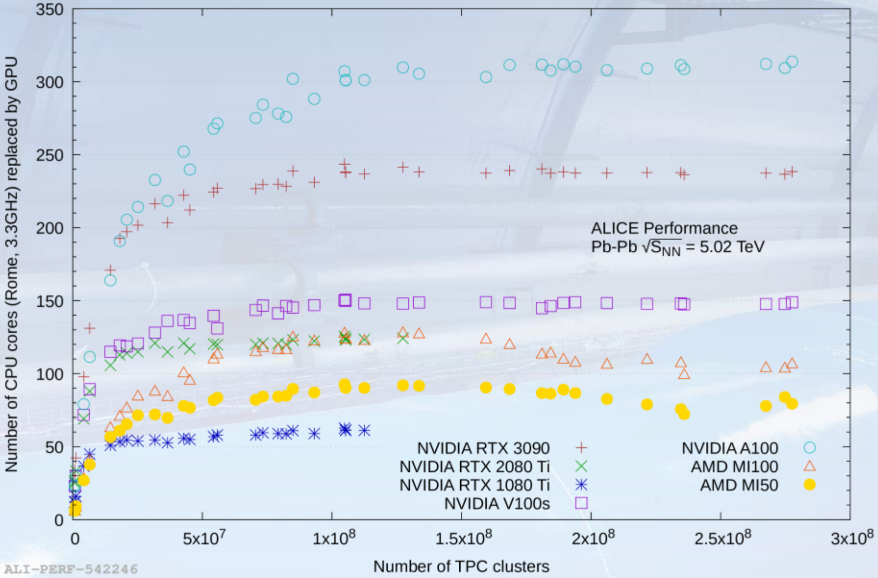
Implementation details

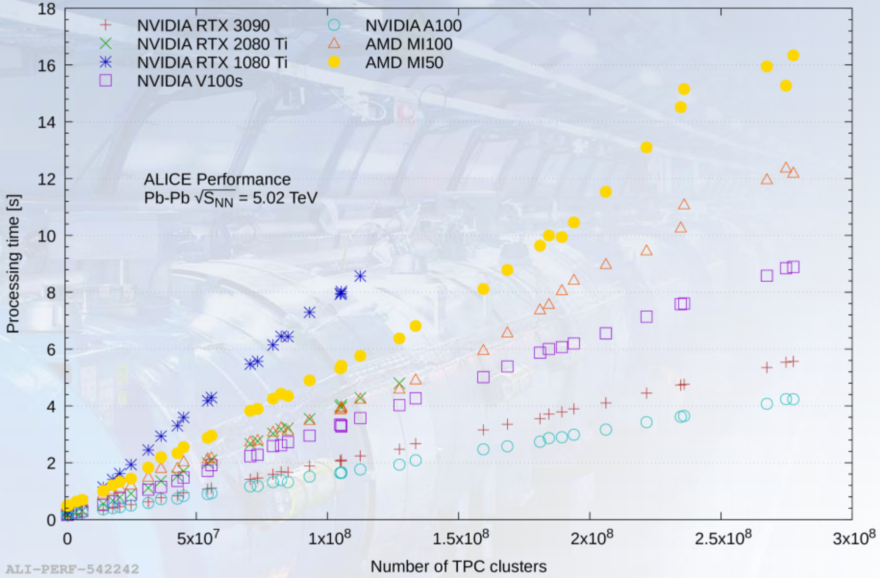
- Multiple Gl
  - Less ser
  - Synergie
- Splitting th
  → 2 virtual
  - Still only
- GPUs are
  - Host pro



Synchronous processing performance

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





- **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).

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

ALICE

- **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%.

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 (11ms, 1 instance)	Time per TF (11ms, 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 (11ms, 1 instance)	Time per TF (11ms, 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**).
- **Synchronous processing successful in 2021 - 2023.**
  - **pp** data taking and **low-IR Pb-Pb** went **smooth** and as expected, but not causing full compute load.
  - **Full rate** will come with Pb-Pb in **October 2023**.
    - **50 kHz Pb-Pb** processing **validated** with data replay of **MC** data (~ **30% margin**).
- **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**.