ALICE triggerless readout, software trigger and online reconstruction

David Rohr for the ALICE Collaboration

Realtime Workshop, Giessen, 2024

8.4.2024

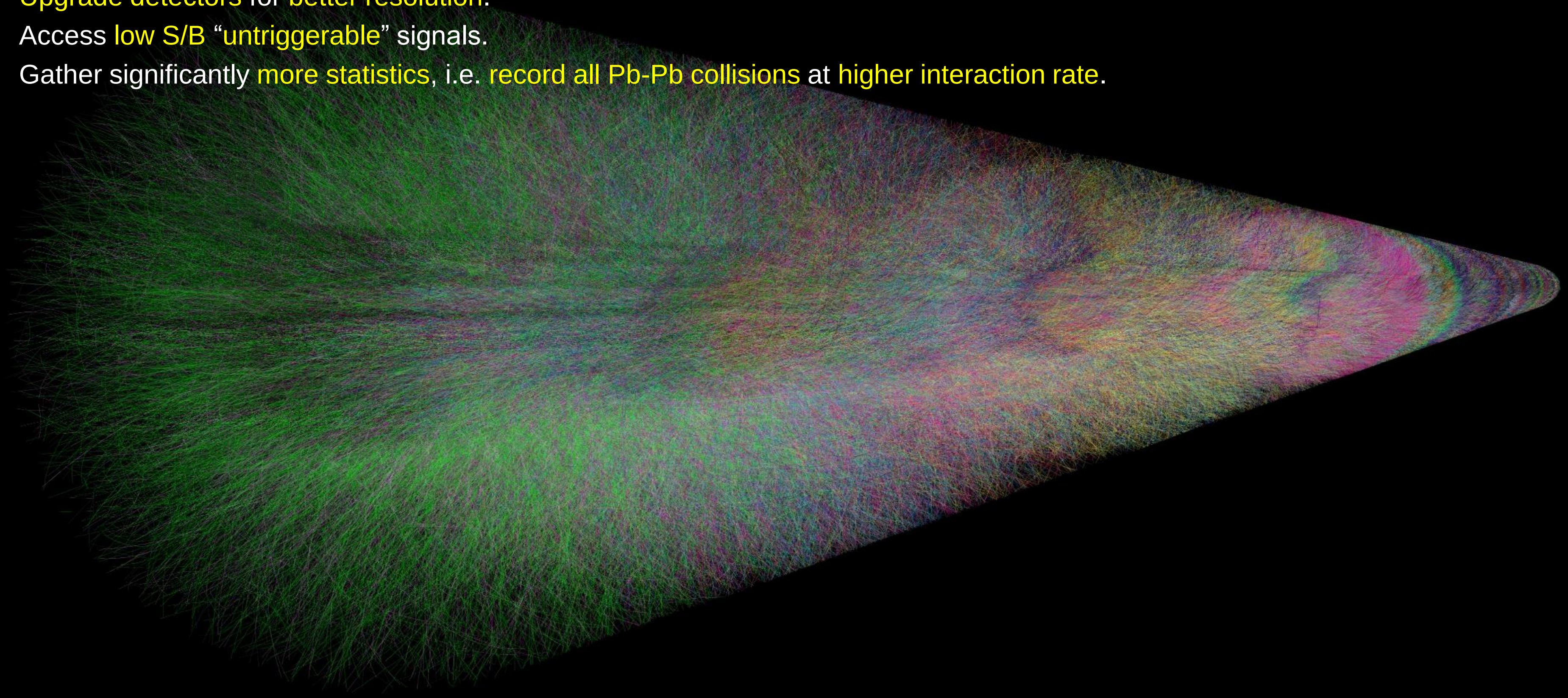
drohr@cern.ch

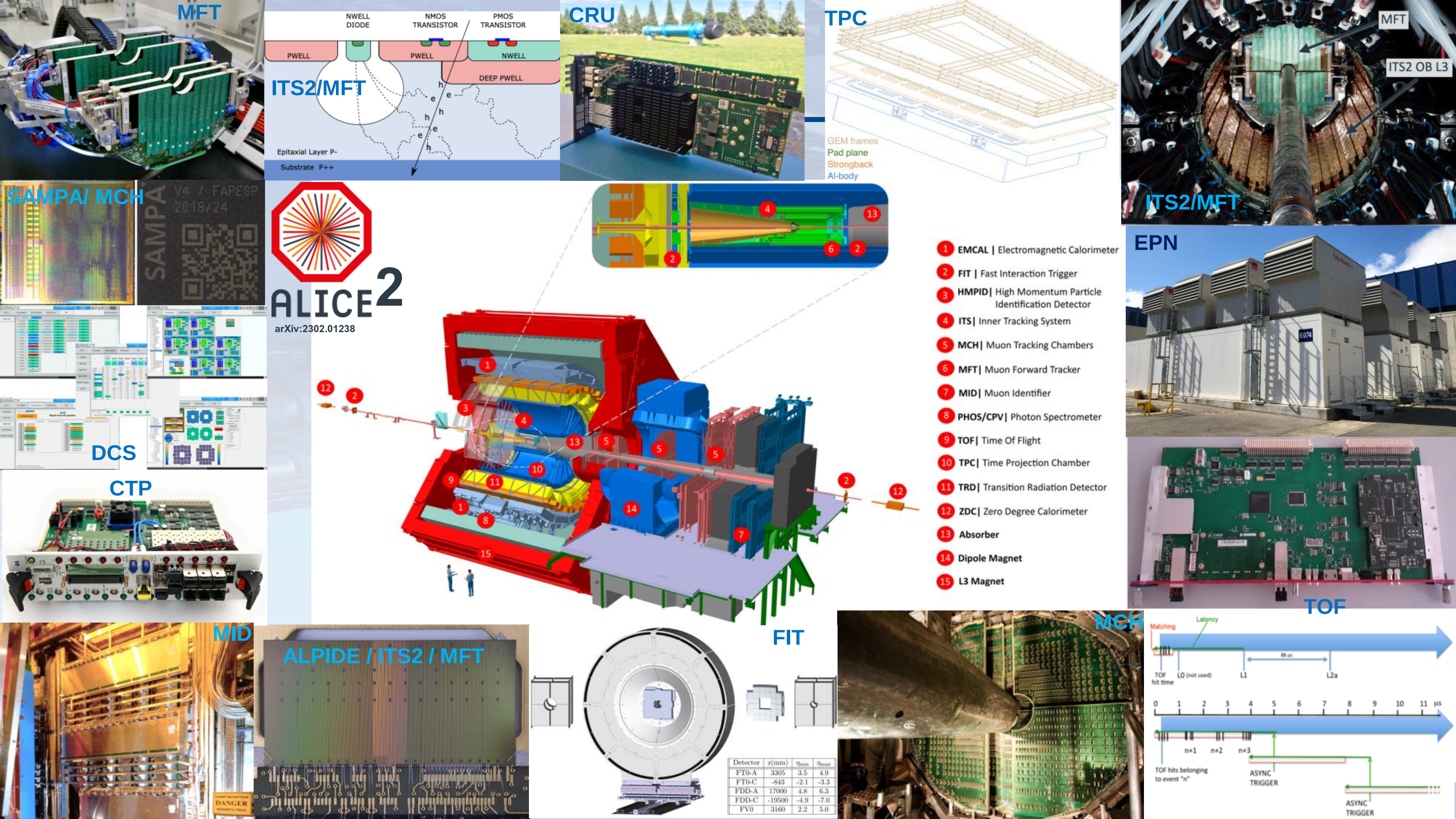
ALICE Goals for Run 3

Upgrade detectors for better resolution.

Access low S/B “untriggerable” signals.

Gather significantly more statistics, i.e. record all Pb-Pb collisions at higher interaction rate.

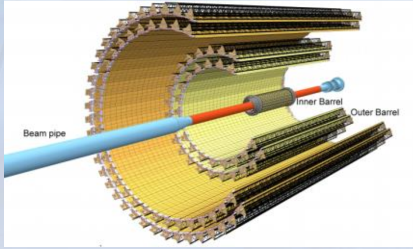




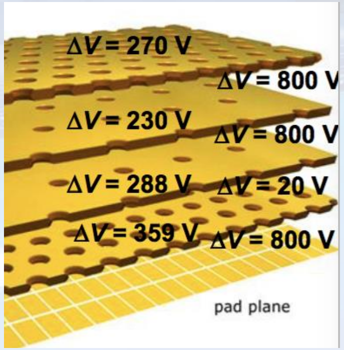
Detector	z(mm)	η_{min}	η_{max}
FT0-A	3305	3.5	4.9
FT0-C	-843	-2.1	-3.3
FDD-A	17000	4.8	6.3
FDD-C	-19500	-4.9	-7.0
FV0	3160	2.2	5.0

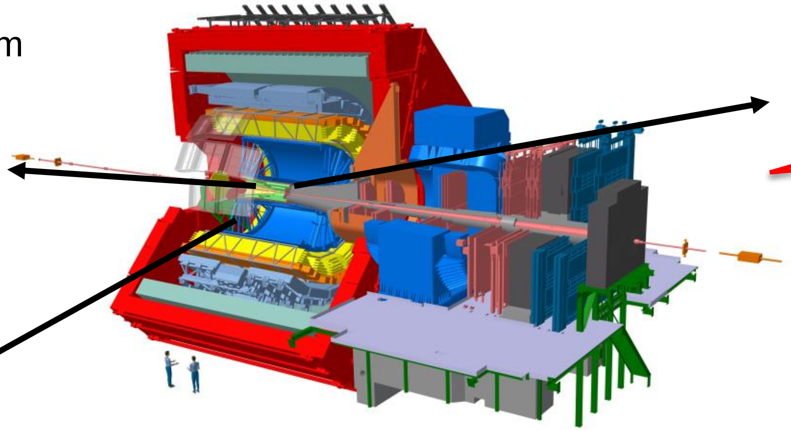
LS2 ALICE Upgrades

All-pixel Inner Tracking System

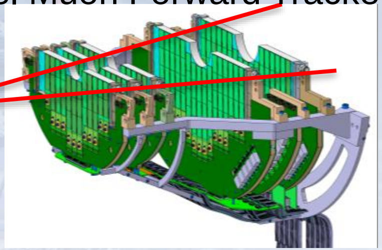


GEM-based
TPC readout





Pixel Muon Forward Tracker

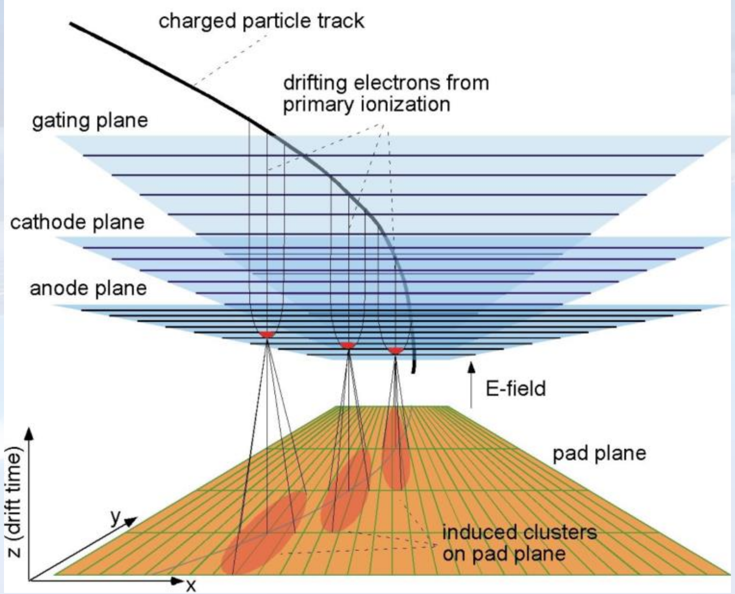


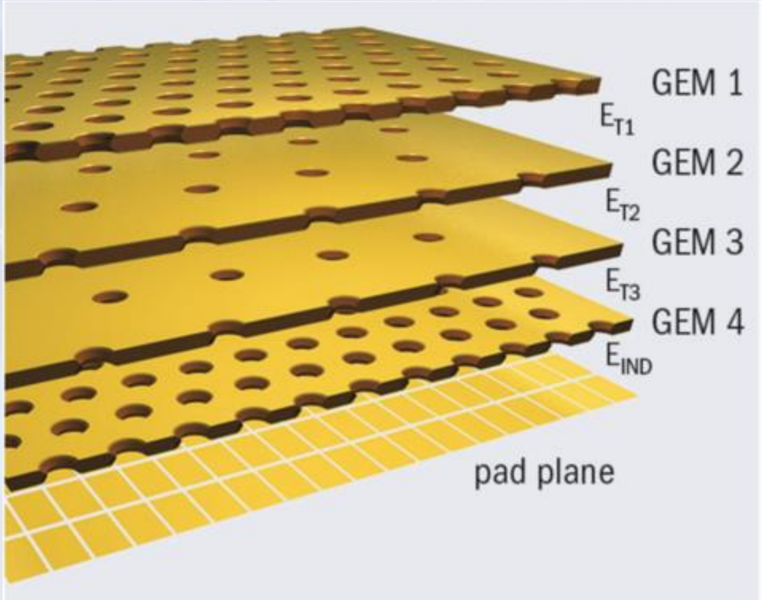
- **New detectors:**
  - Improve tracking resolution at low p_T
    → thinner, more granular
  - Enable continuous read-out
- **New online-offline computing system for synchronous and asynchronous processing**

… and much more:
- Fast Interaction Trigger
- New 50x faster readout system
- Readout upgrade of MUON, TOF, EMCAL, PHOS

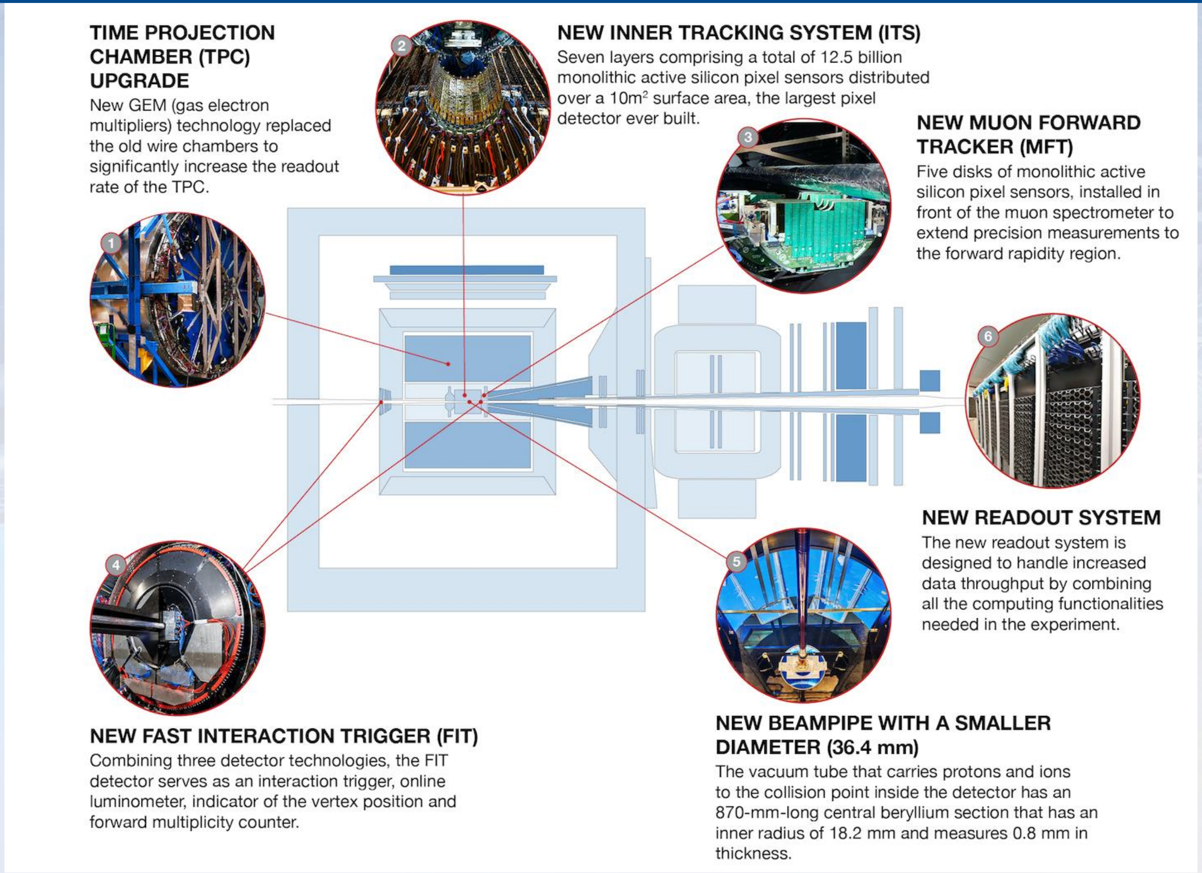
ALICE TPC upgrades and implications

- **Need continuous TPC (Time Projection Chamber) readout to store full minimum bias sample.**
  - TPC of **Run 1 and 2** used **MWPC** (Multi Wire Proportional Chambers) readout and **gating grid** to **suppress ion back flow**.
  - Gating grid **limits readout to ~3 kHz**, prevents continuous readout.
  - → **Replace MWPCs with GEMs** (Gas Electron Multiplier),
    Intrinsic ion back flow blocking (**99%**), no gating grid.





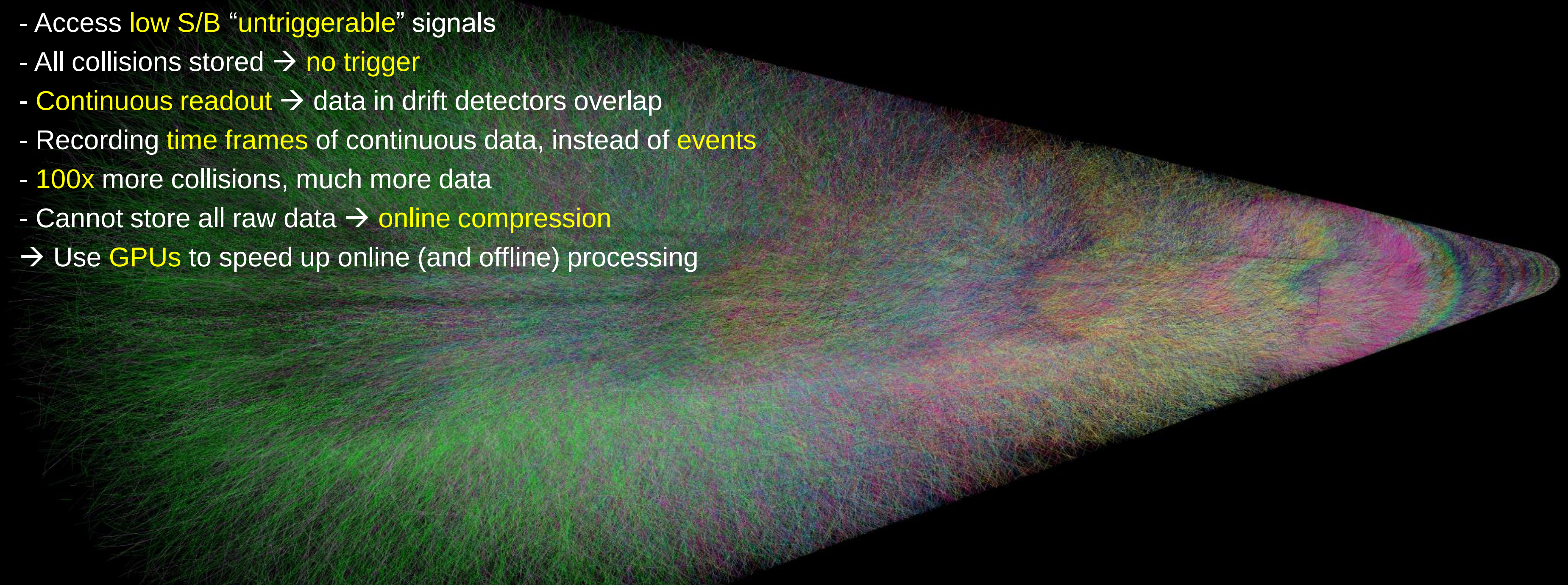
LS2 ALICE Upgrades



ALICE in Run 3

Targeting to record large minimum bias sample.
- Access low S/B “untriggerable” signals
- All collisions stored → 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

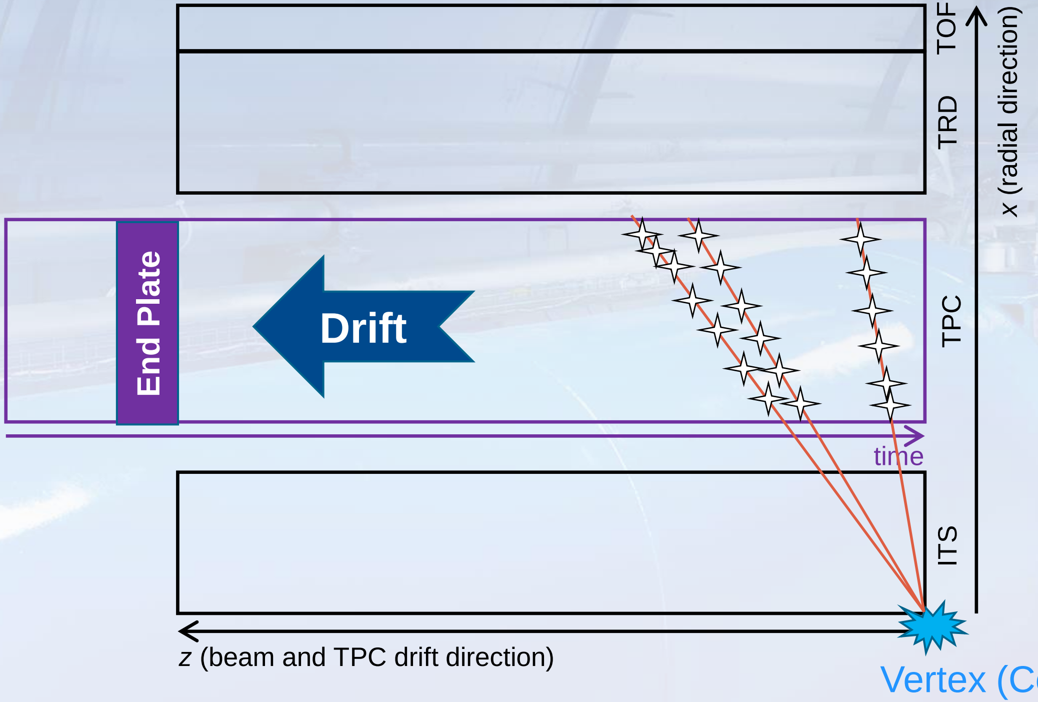


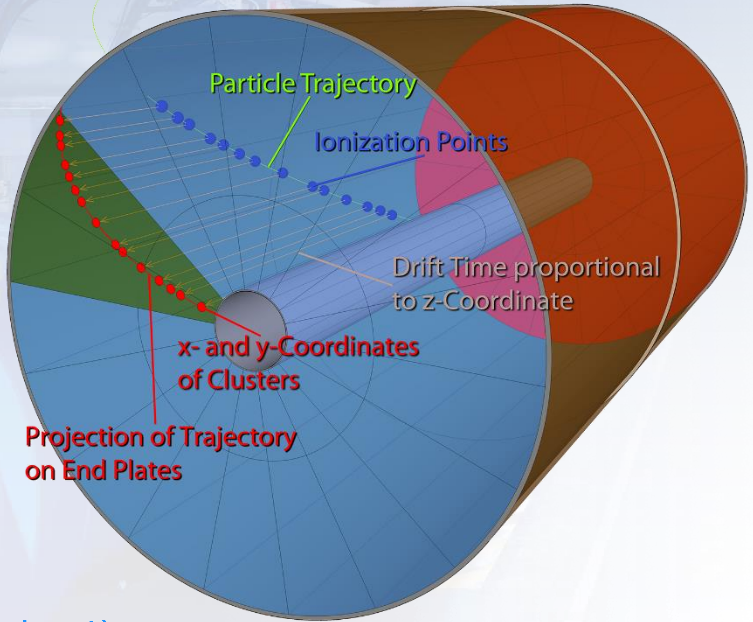
- Overlapping events in TPC with realistic bunch structure @ 50 kHz Pb-Pb.
- Timeframe of 2 ms shown (will be 10 – 20 ms in production).
- Tracks of different collisions shown in different colors.

The tracking challenge

- **Tracking continuous data…**
  - The TPC sees **multiple overlapped collisions** (shifted in time).
  - Other detectors know the (rough) time of the collision.

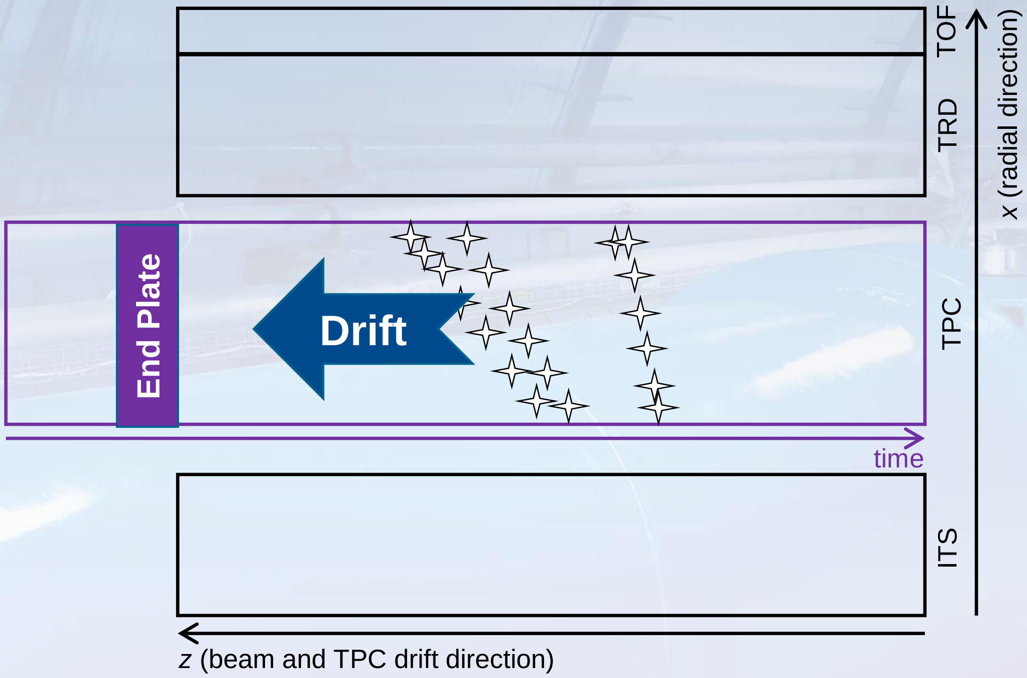
- **Problem: TPC clusters have no defined *z*-position but only a time. They can be shifted in *z* arbitrarily.**



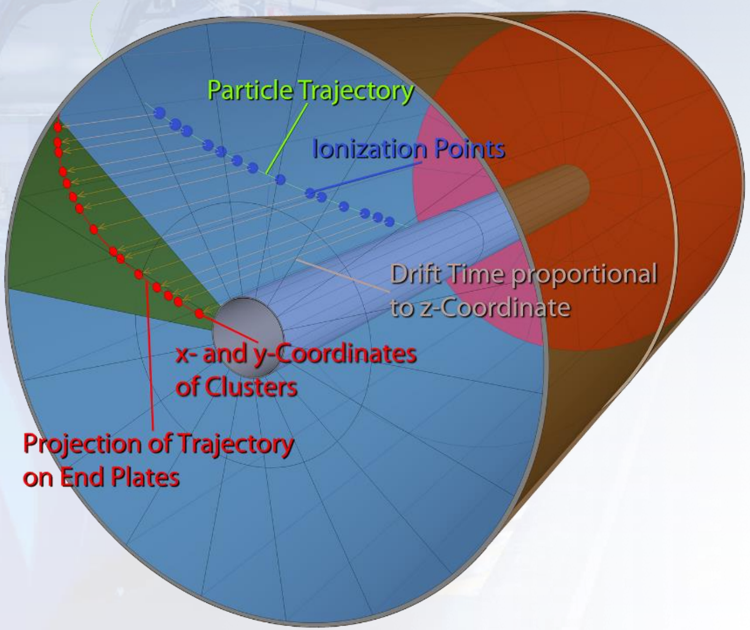


The tracking challenge

- **Tracking continuous data…**
  - The TPC sees **multiple overlapped collisions** (shifted in time).
  - Other detectors know the (rough) time of the collision.

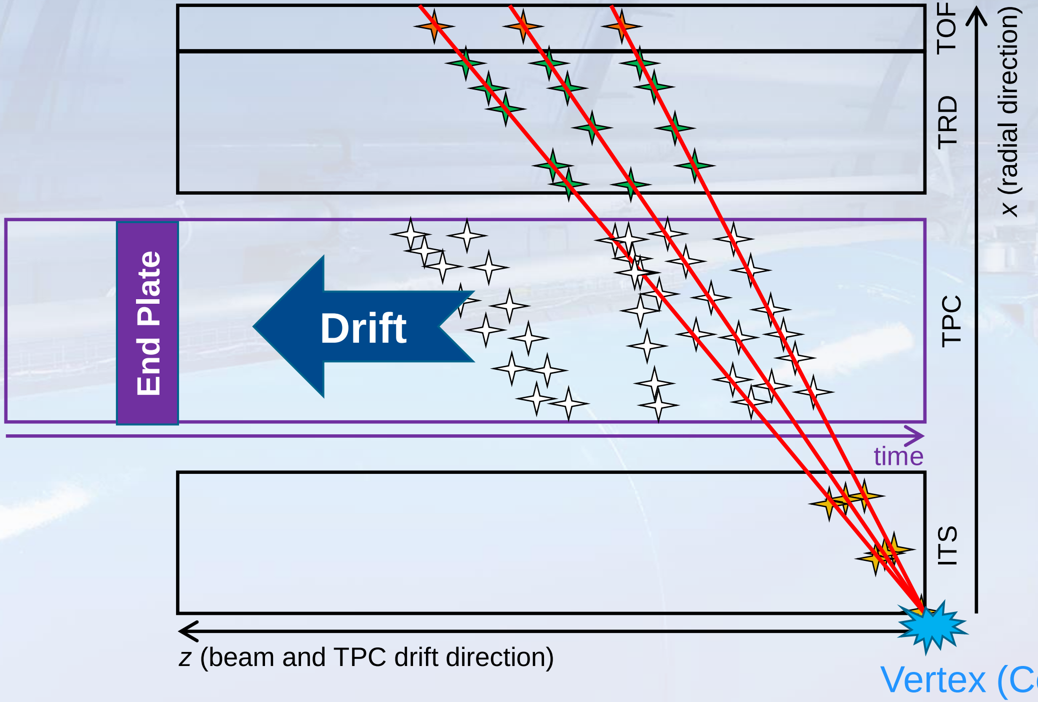


- **Problem: TPC clusters have no defined *z*-position but only a time. They can be shifted in *z* arbitrarily.**

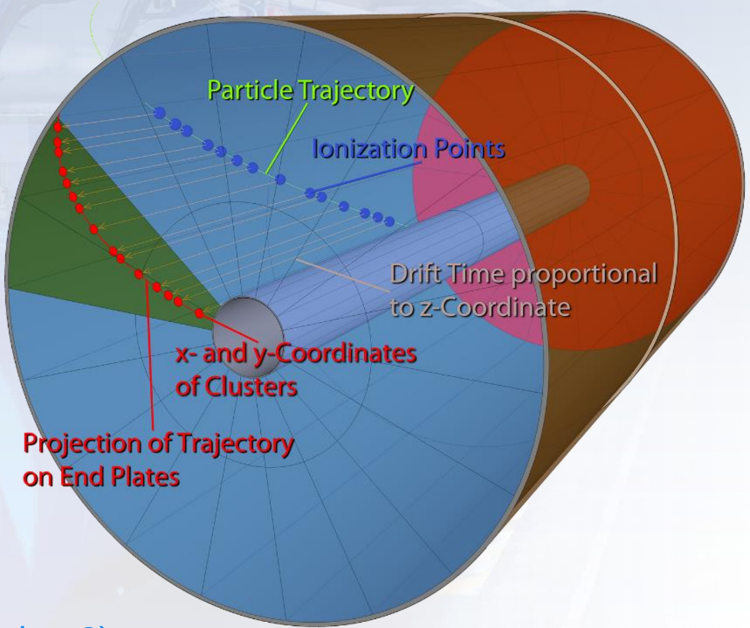


The tracking challenge

- **Tracking continuous data…**
  - The TPC sees **multiple overlapped collisions** (shifted in time).
  - Other detectors know the (rough) time of the collision.

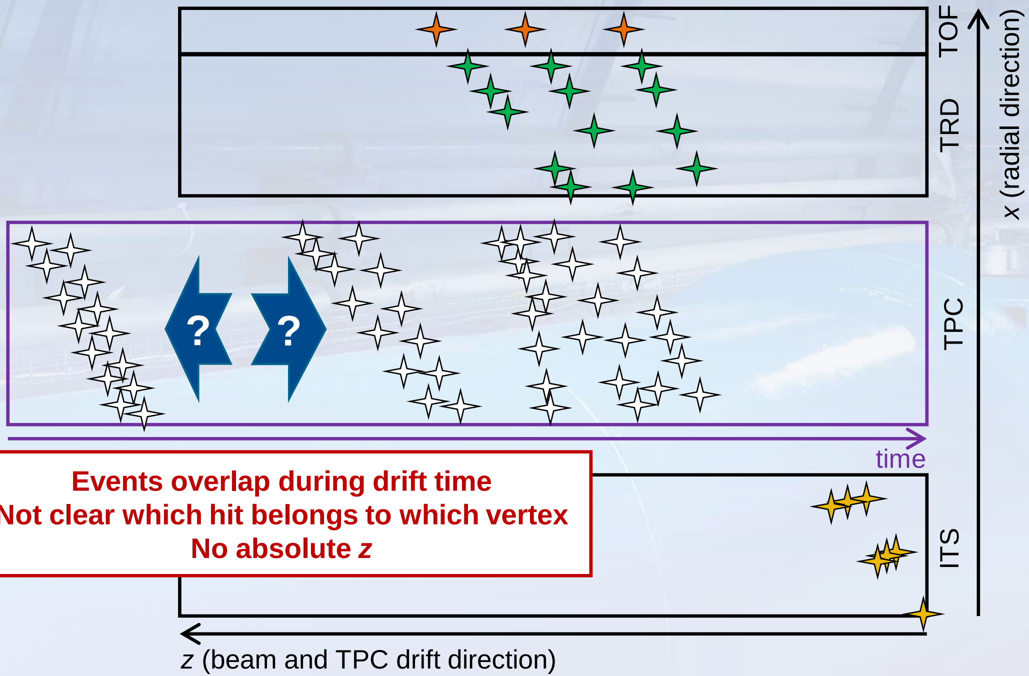


- **Problem: TPC clusters have no defined *z*-position but only a time. They can be shifted in *z* arbitrarily.**

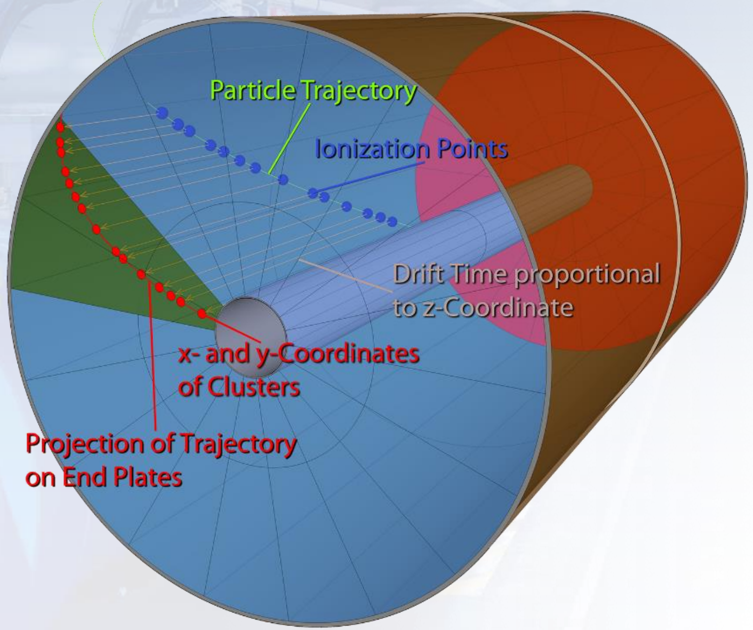


The tracking challenge

- **There are 2 (related) main challenges caused by continuous readout / space charge distortions**
  - How to assign a z-position to a cluster?
  - How to apply SCD corrections (inhomogeneous magnetic field, cluster error parameterization) if z is now known.

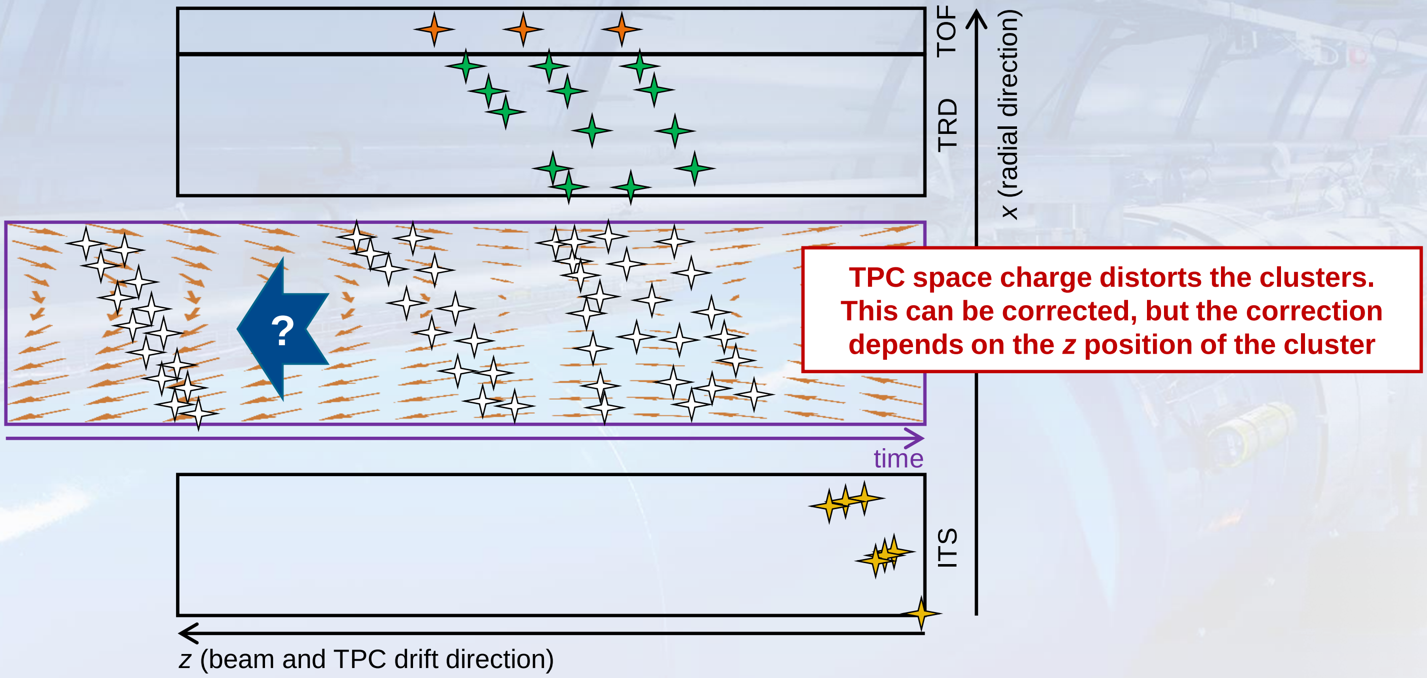


Events overlap during drift time
Not clear which hit belongs to which vertex
No absolute *z*



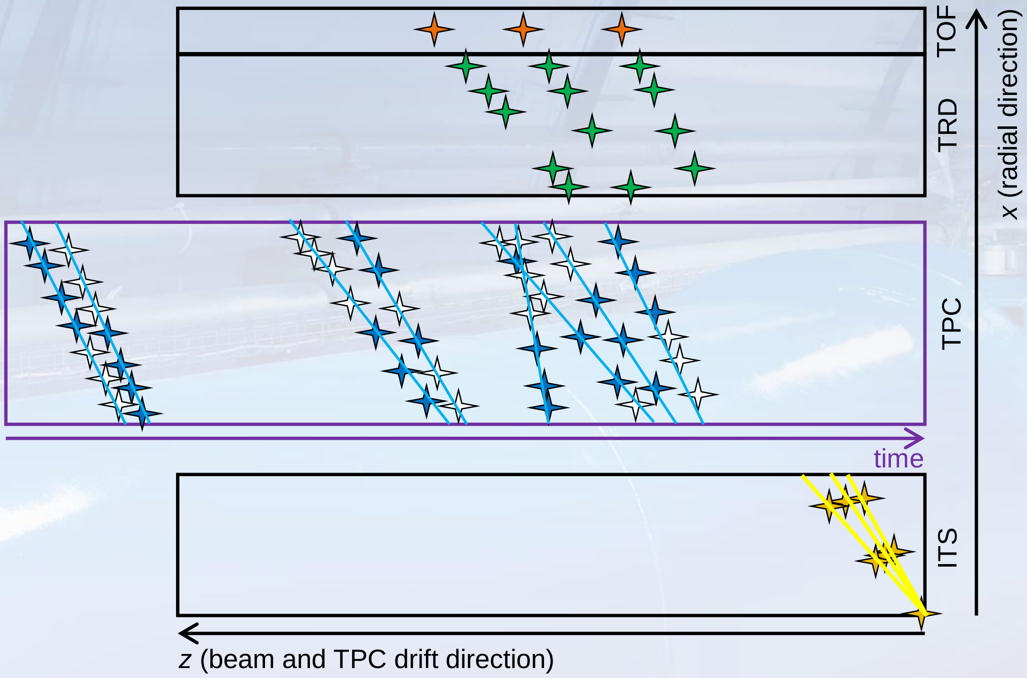
The tracking challenge

- **There are 2 (related) main challenges caused by continuous readout / space charge distortions**
  - How to assign a z-position to a cluster?
  - How to apply SCD corrections (inhomogeneous magnetic field, cluster error parameterization) if z is now known.



TPC Tracking

- **There are 2 (related) main challenges caused by continuous readout / space charge distortions**
  - How to assign a z-position to a cluster?
  - How to apply SCD corrections (inhomogeneous magnetic field, cluster error parameterization) if z is now known.



- Standalone ITS tracking.
- Standalone TPC tracking, scaling *t* linearly to an arbitrary *z*.

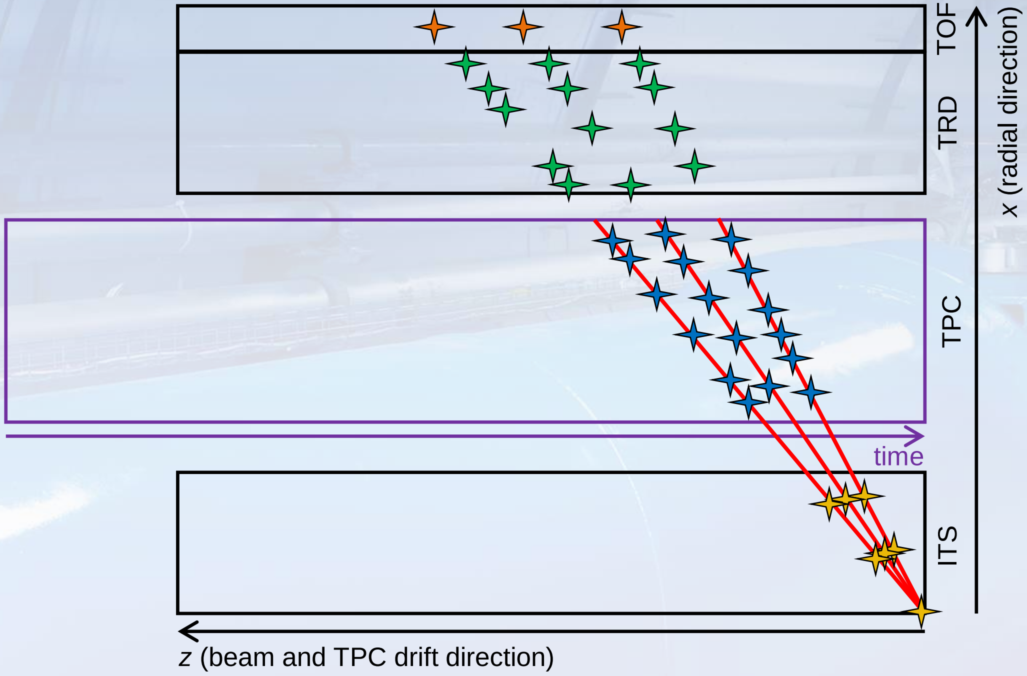
Precise tracking needs *z* for:

- Cluster error parameterization
- Inhomogeneous B-field
- Distortion correction

Effects smooth →
irrelevant for initial trackletting

TPC Tracking

- **There are 2 (related) main challenges caused by continuous readout / space charge distortions**
  - How to assign a z-position to a cluster?
  - How to apply SCD corrections (inhomogeneous magnetic field, cluster error parameterization) if z is now known.



- Standalone ITS tracking.
- Standalone TPC tracking, scaling t linearly to an arbitrary z.
- Extrapolate to $x = 0$, define $z = 0$ as if the track was primary.
- Track following to find missing clusters. For cluster error parameterization, distortions, and B-field, shift the track such that $z = 0$ at $x = 0$.
- Refine $z = 0$ estimate, refit track with best precision
- For the tracks in one ITS readout frame, select all TPC tracks with a compatible time (from $z = 0$ estimate).
- Match TPC track to ITS track, fixing z-position and time of the TPC track.
- Refit ITS + TPC track outwards.

Processing considerations

- **TPC standalone tracks cannot have a precise time stamp / vertex assignment on their own, but only after matching to other detectors.**
- **Event reconstruction cannot process a “single event / collision” by design:**
  - We know only after the tracking which track belongs to which collision.
  - And for tracks not originating clearly from a primary vertex, this is only known with a certain probability.

- **Data unit for the processing cannot be an “event” like in Run 2.**
- **Instead, we record / process time frames with a configurable length of up to 256 drift times.**
  - Smaller drift times leas to more statistic loss due to effects at the time frame boundaries.
  - Larger time frames need more memory for the processing.
  - Current compromise is 32 drift times per TF (~2.5 ms of continuous data).

- **Note that this reduces / simplifies the processing rates (not data rates) a lot.**
  - In run 2 pp we could have several kHz of event rate.
  - No we have ~350 Hz of TF rate.
  - This simplifies the scheduling, and makes sure that we send fewer but larger data chunks around.
  - Also helps with parallelism in the processing, with larger data chunks processed at once.

Readout process

- **During the readout, data is organized in heart beat frames (HBF) of ~90 us each.**
  - Each HBF can consist of multiple pages with 8 kb each.
  - The data distribution software on the readout nodos aggregates the HBFs into TFs.
  - For the detectors / readout, everything is just a continuous stream of HBFs.

- **Is all of ALICE triggerless?**
  - Actually not, several of the detectors were upgraded for full native continuous read out.
  - But some “legacy” detectors still require a trigger.
  - The CTP tries to trigger these detectors for minimum-bias, i.e. to record all collisions.
    - Or if the rate is limited, for the largest possible subset of collisions.
    - For instance, the scheme for the TRD foresaw ~40 kHz trigger rate in Run 3, compared to 50 kHz maximum interaction rate, i.e. only 80% of the events would have TRD contribution.
  - With multiple such detectors, the CTP will ensure to trigger the same subset.

- **LHC runs ~half a year of pp compared to 3 weeks of Pb-Pb → We get more pp data then Pb-Pb, even at the relatively low ALICE interaction rates of 500kHz / 1MHz**
  - Cannot store all pp data.
  - → ALICE performs CTF skimming: All pp data is stored to disk first, but then it is skimmed after data taking using physics analysis triggers to decide which collisions to keep permanently.

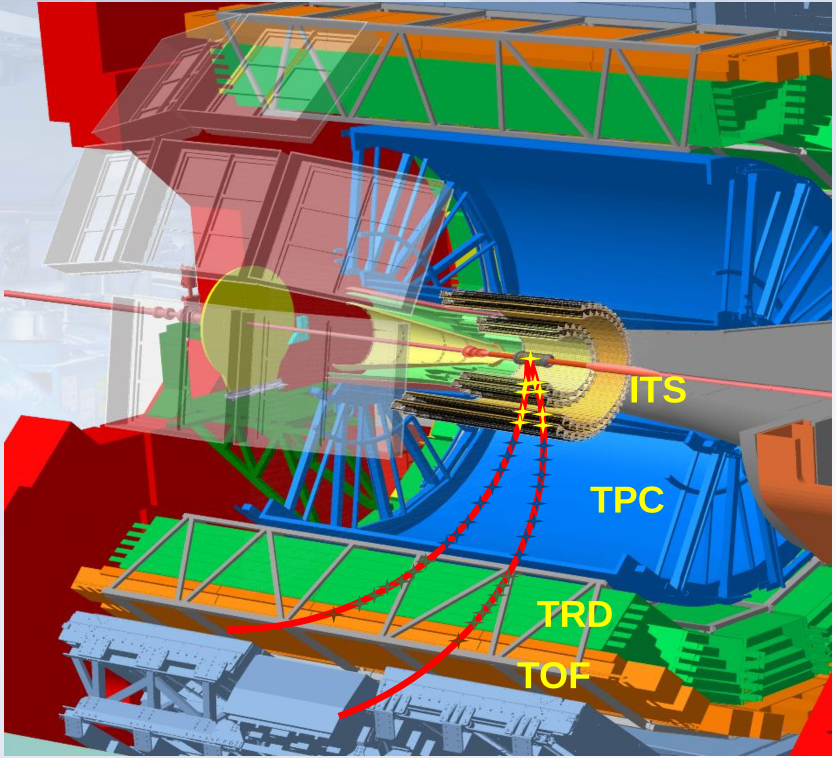
Readout process

- **ALICE uses the Common Readout Unit (CRU) card to receive the optical links from the detectors in the readout farm.**
  - The FPGA-based card is developed by LHCb (PCIe40), the CRU firmware is developed by ALICE.
  - Some legacy detectors with low rate still use the C-RORC card (ALICE’s readout card of Run 2).

- **Detectors can**
  - either send HBFs directly,
  - or a “user logic” in the CRU creates HBFs out of
  the data send by the detectors.
  - E.g. the TPC sends just a stream of raw ADC values, the CRU performs common-mode correction, ion tail filtering, and zero suppression, and then packages the data into HBFs.
    - This is an example of local processing happening already in the FPGA.

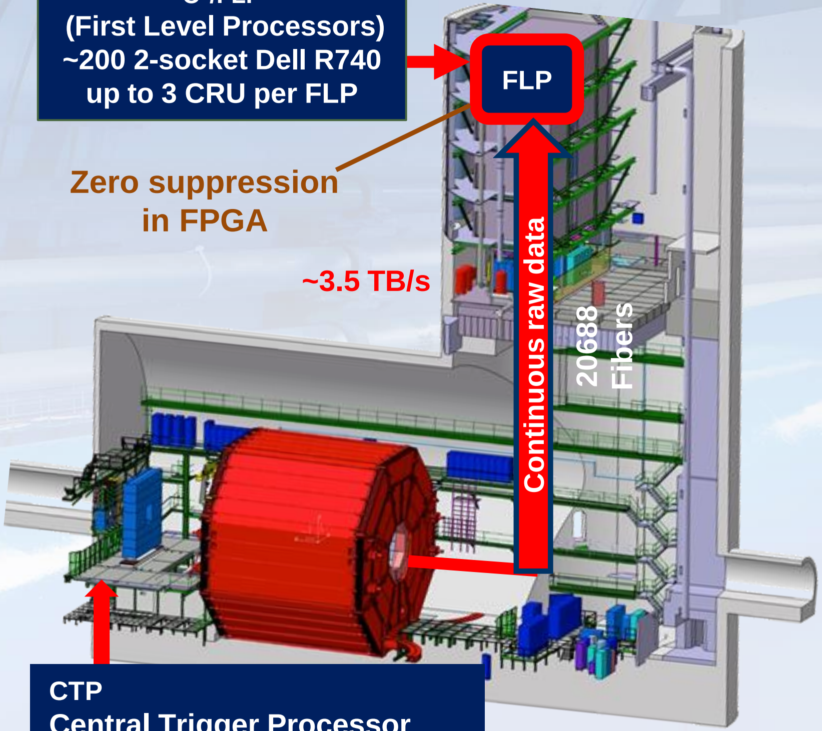
The ALICE detector (barrel region) in Run 3

- **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)

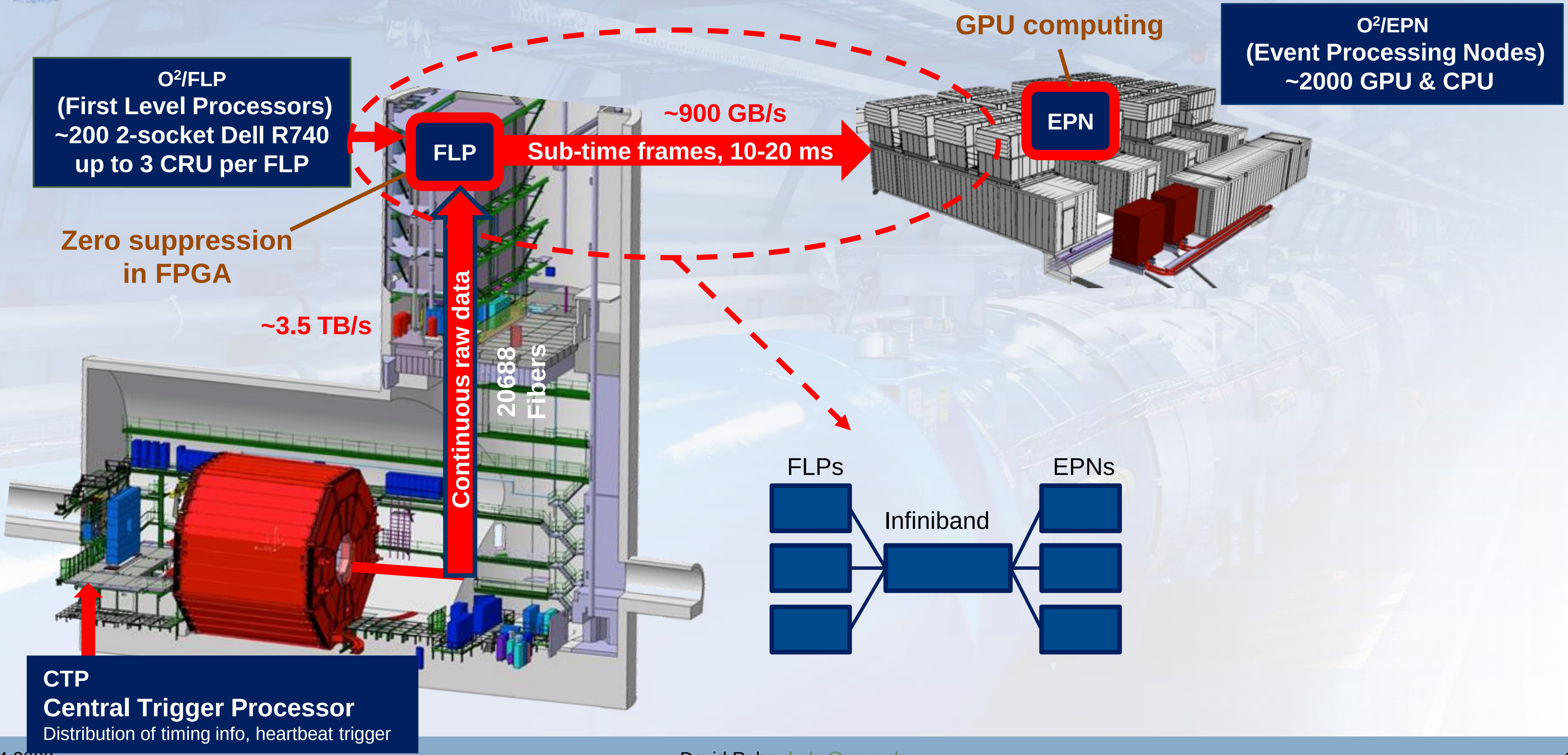


- **ALICE performs continuous readout.**
- **Native data unit is a time frame: all data from a configurable period of data up to 256 LHC orbits.**
  - Default was ~11 ms (128 LHC orbits) before 2023.
  - Current default is **~2.8 ms** (32 LHC orbits)

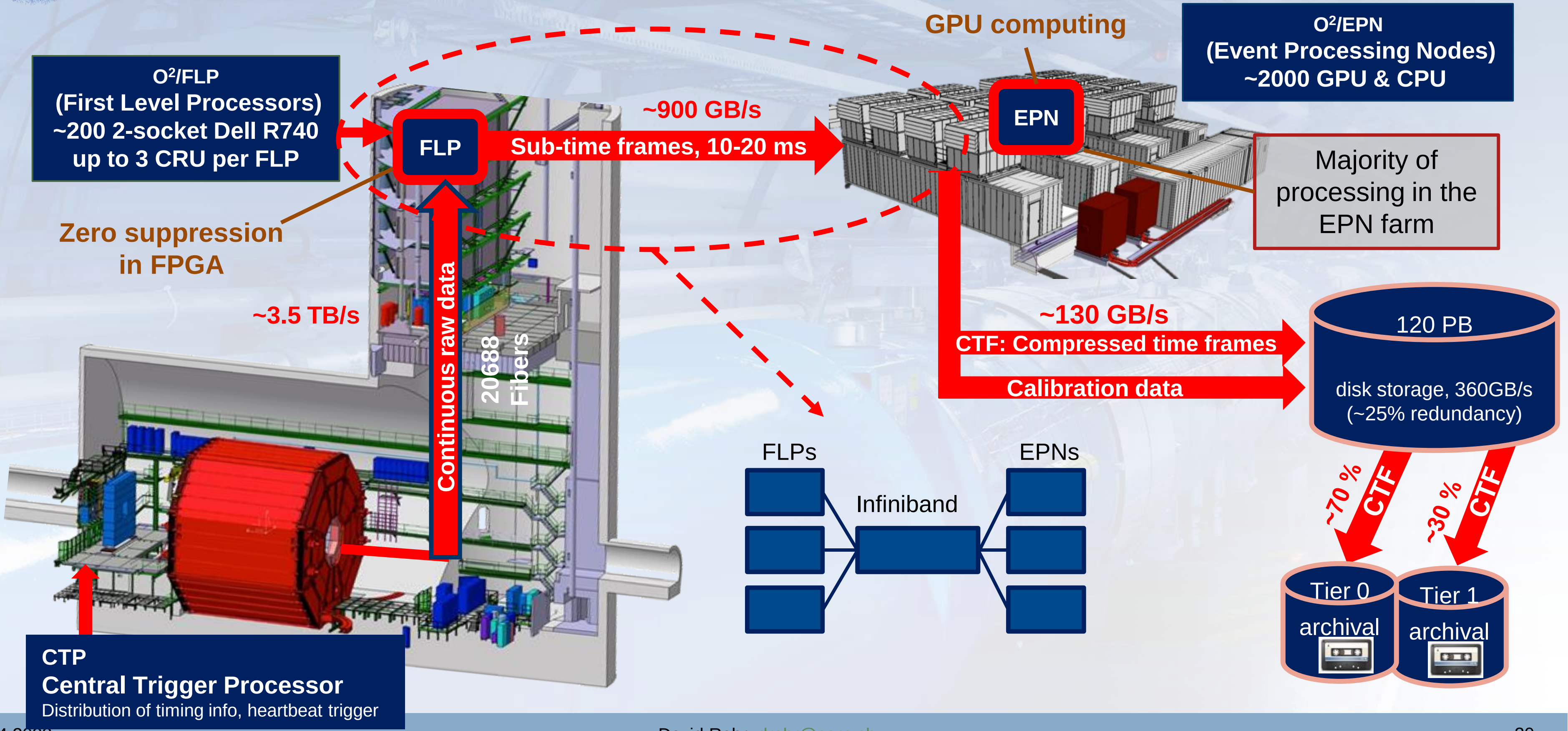
ALICE Raw Data Flow in Run 3



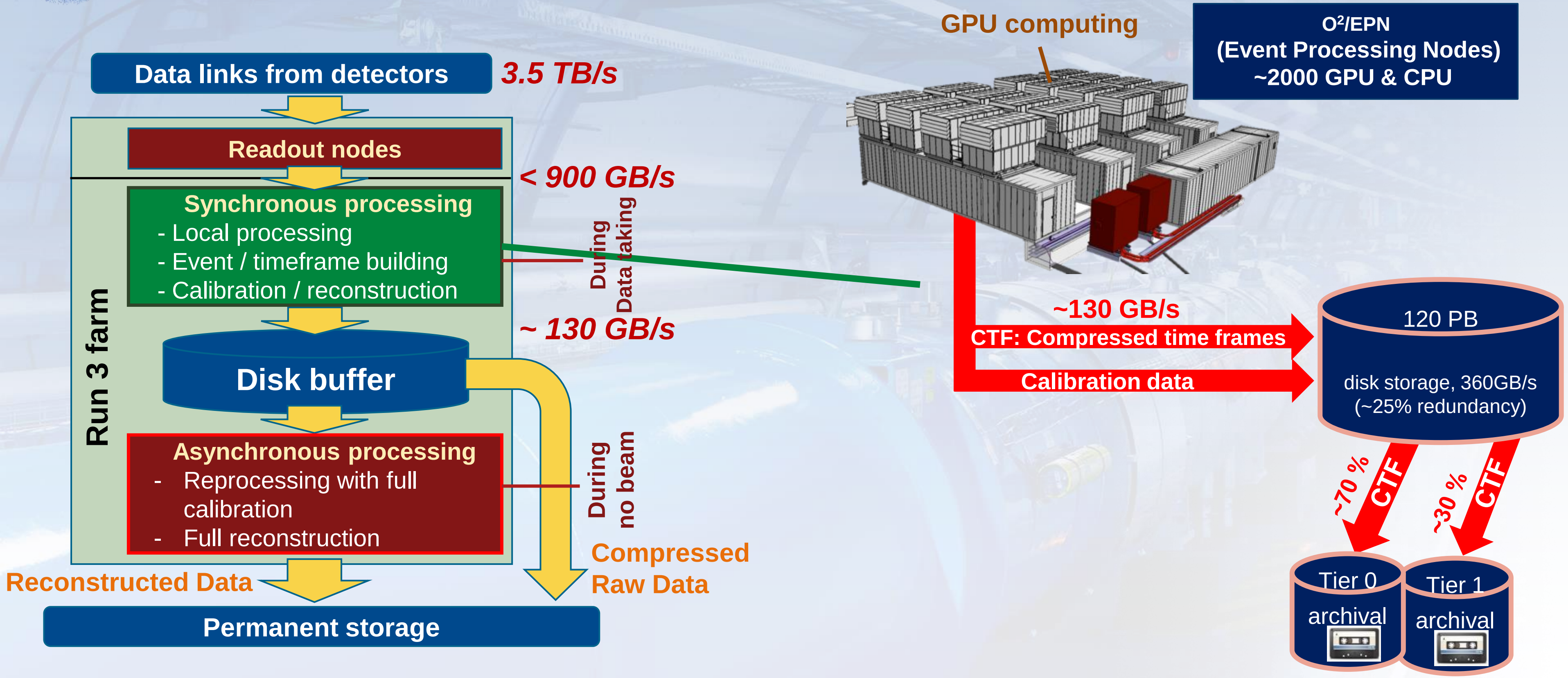
ALICE Raw Data Flow in Run 3



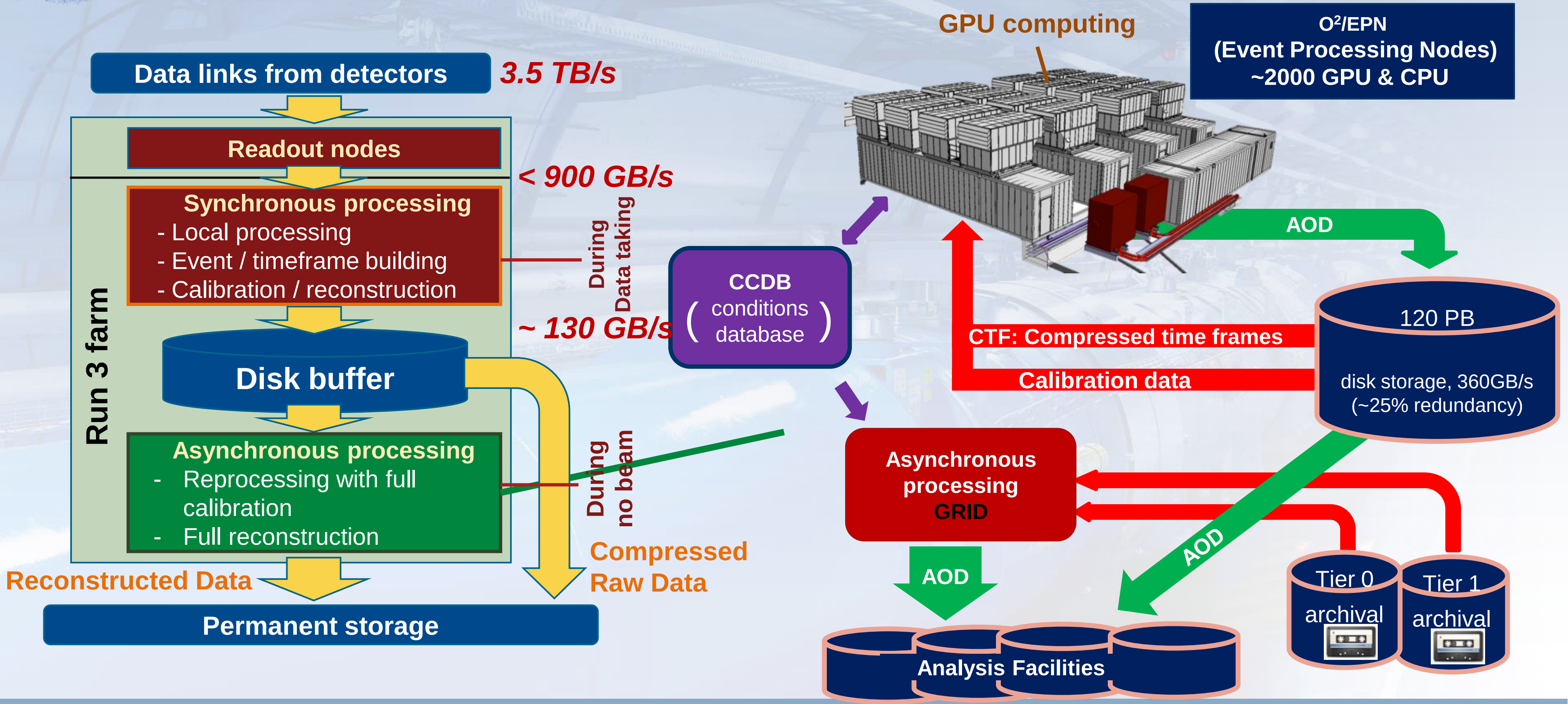
ALICE Raw Data Flow in Run 3



Synchronous and Asynchronous Processing



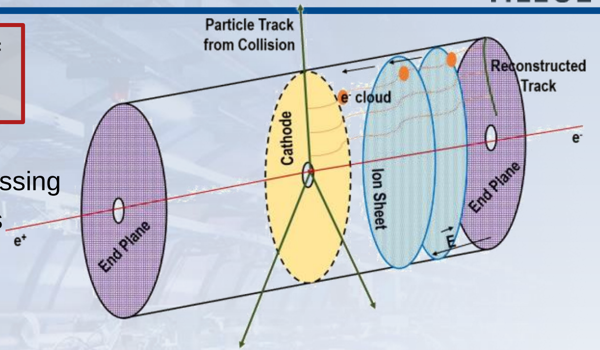
Synchronous and Asynchronous Processing



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

Needs tracking of 1% of tracks

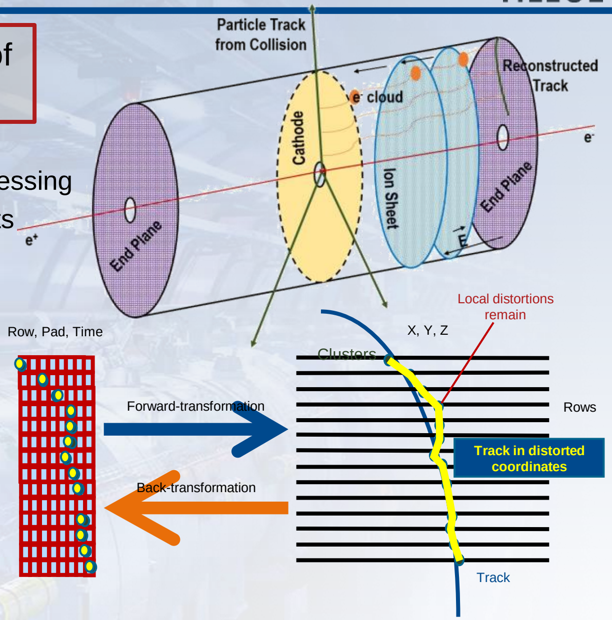


O² Processing steps

- **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

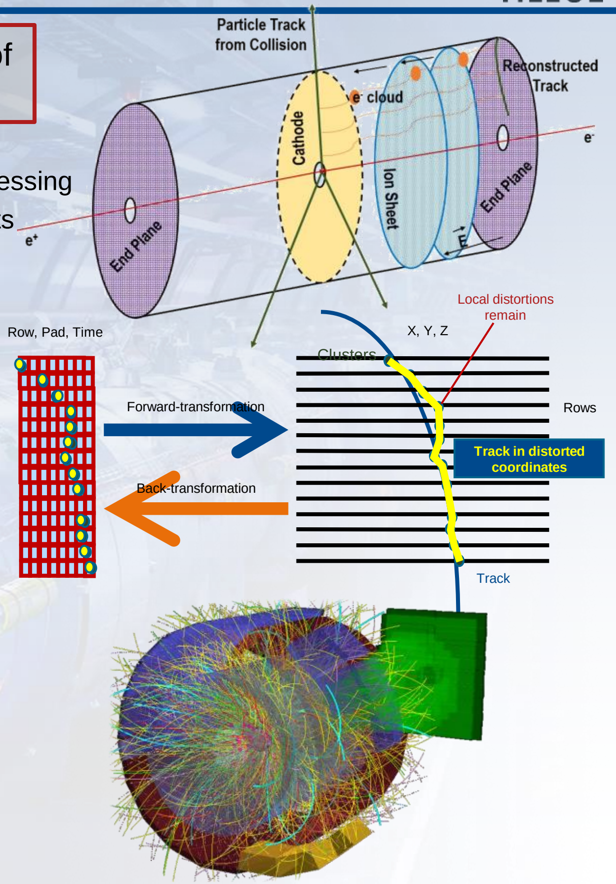


O^2 Processing steps

- **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

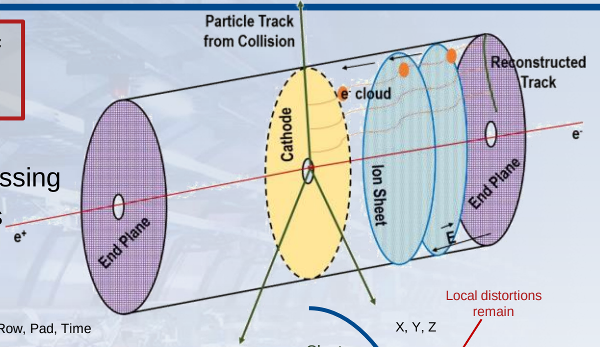


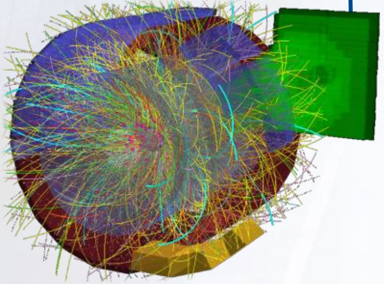
O² Processing steps

- **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
  - **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

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.

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** :
  - **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.

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!

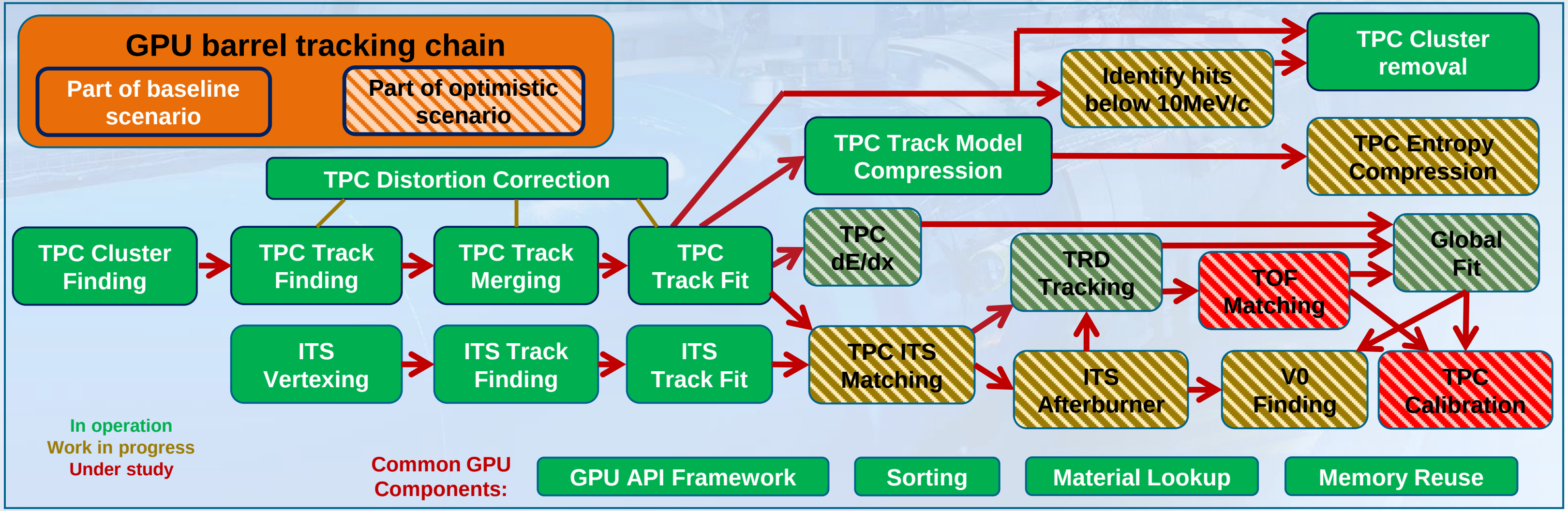
- **GPUs** well suited for **TPC** reco (from Run 1 and 2 experience).
- **GPUs** provide the **required compute power**.
  - Time frame concepts yields large enough GPU data chunks.
- Following up **2 scenarios** for EPN GPU processing:

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

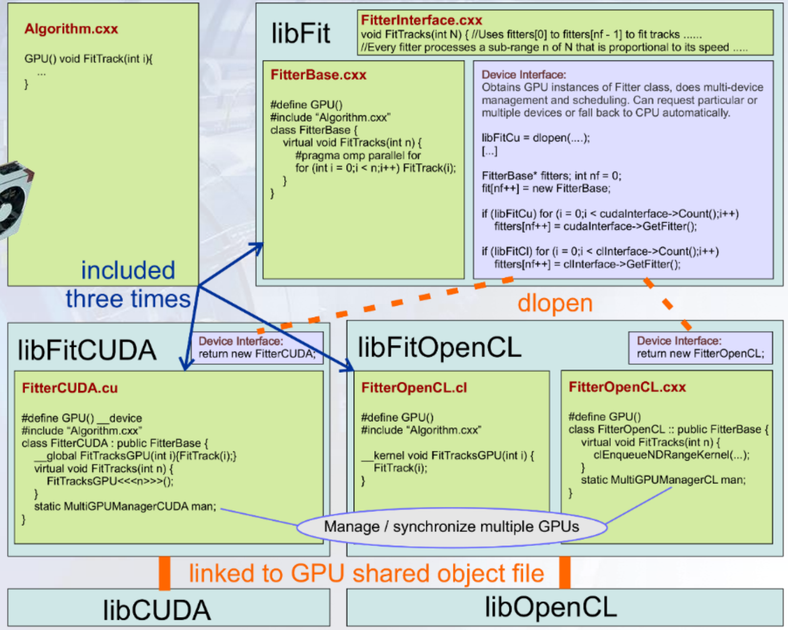
- **Central barrel tracking chosen as best candidate for optimistic scenario for asynchronous reco:**
  - Mandatory **baseline scenario** includes everything that must run on the GPU during synchronous reconstruction.
  - **Optimistic scenario** includes everything related to the barrel tracking.



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)



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.
- **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).

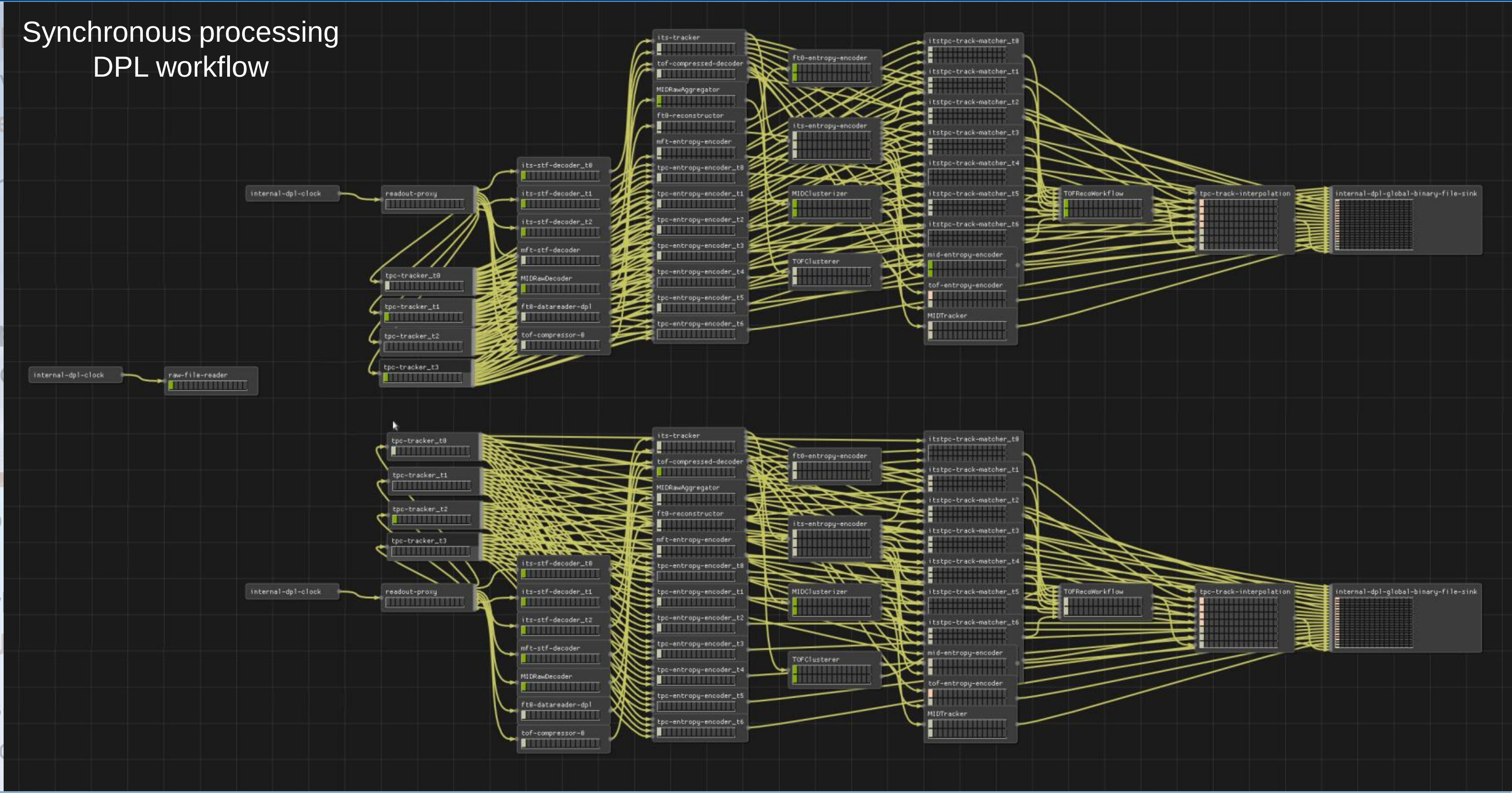
Implementation details

- **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

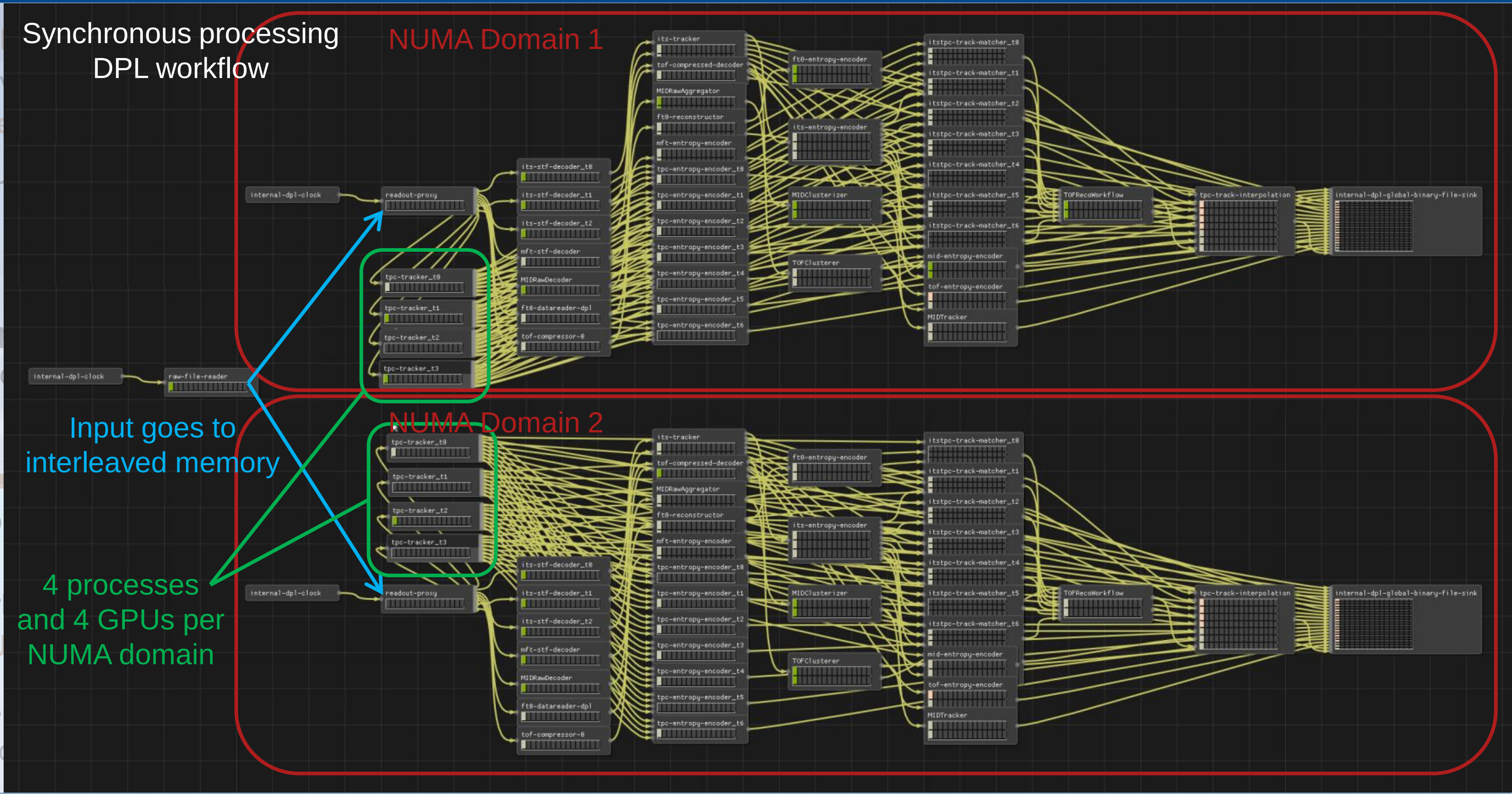
- Multiple G
  - Less ser
  - Synergie
- Splitting t
  → 2 virtual
  - Still only
- GPUs are
  - Host pro
- GPUs can
  - With me
  - With sep
- Benchmar
  - ~50 CPU
- Selected s
  - Supermi

Synchronous processing
DPL workflow



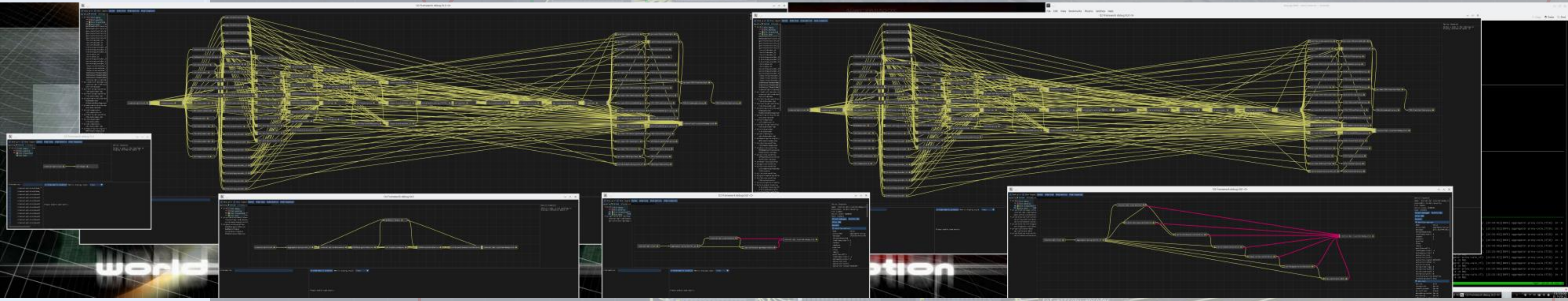
Implementation details

- Multiple 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
  - Supermi



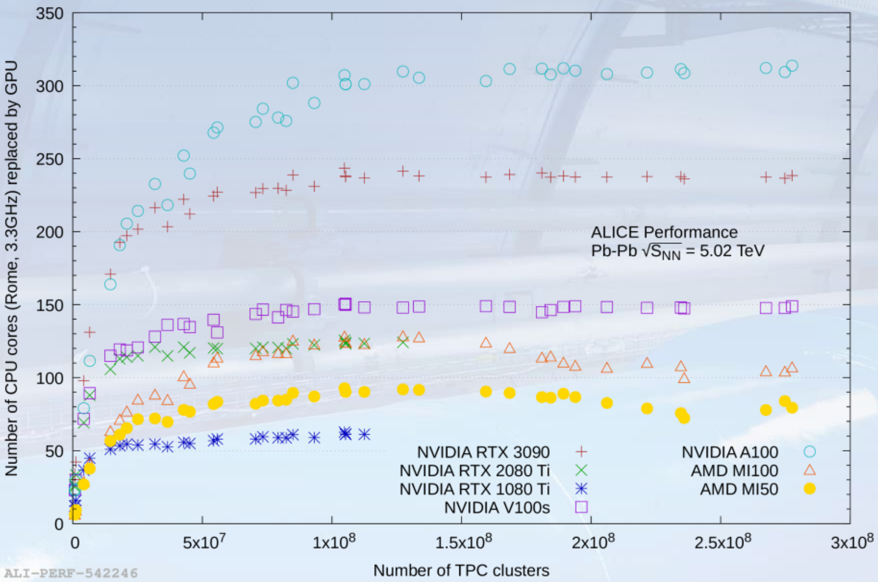
Implementation details

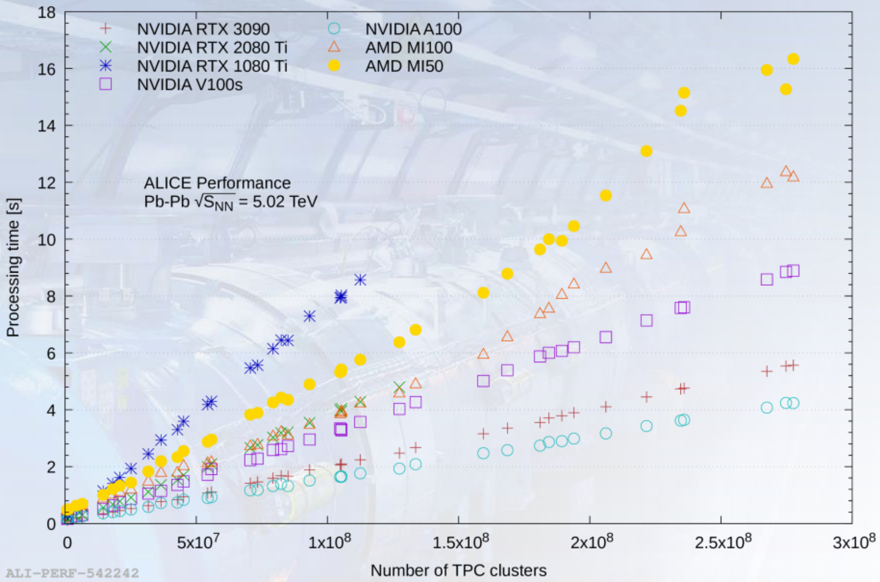
To illustrate the complexity:
Full synchronous workflow including Quality Control and Calibration



Synchronous processing performance

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





- ALICE uses 2240 MI50 and 560 MI100 GPUs in the EPN farm.
- 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 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 has switched to continuous read out in Run.**
  - Enables the **storage of all events**, can access low S/B signals.
  - ~**100x more data** than in Run 2 (50 kHz interaction rate v.s. 500 Hz trigger rate).
  - Required an **upgrade** of the **detectors**, **readout systems**, and **computing scheme**.
- **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**.