# ALICE Online Processing Software and Experience from LHC Pilot Beam Data Taking

**David Rohr**

**Software & Computing Round Table**

*5.4.2022*

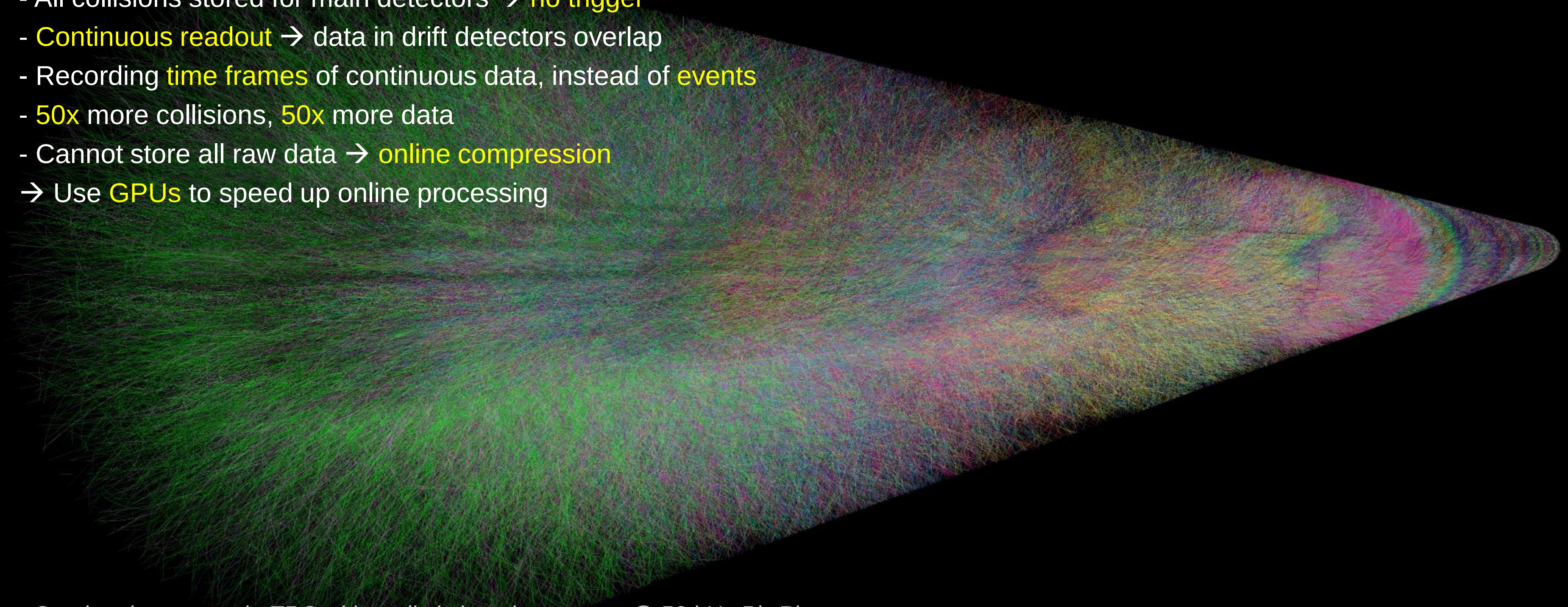
*drohr@cern.ch*

# 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
- 50x more collisions, 50x more data
- Cannot store all raw data → online compression

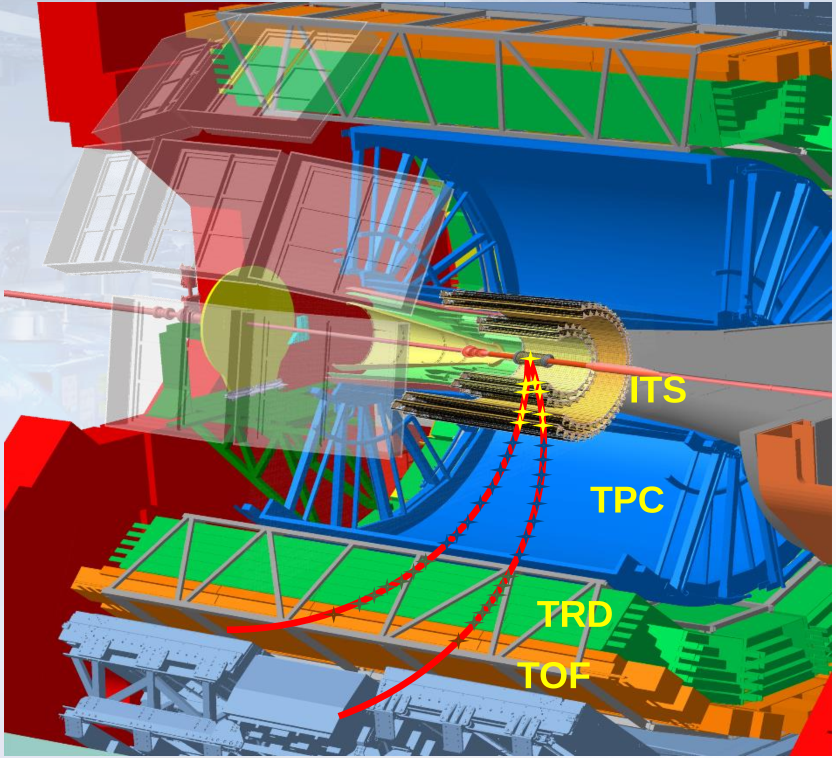
→ Use GPUs to speed up online processing



- 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.

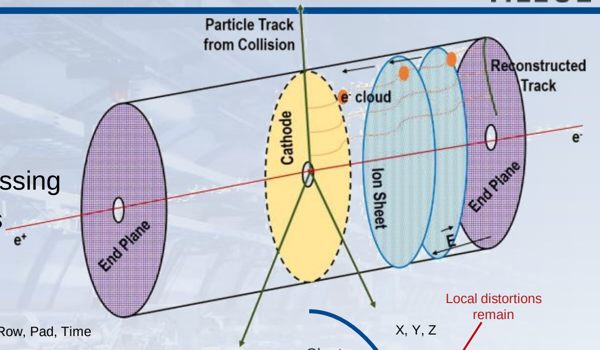
# The ALICE detector in Run 3

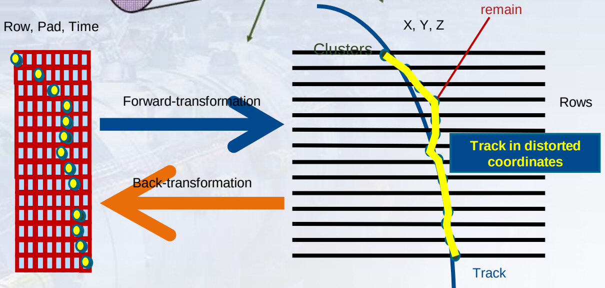
- **ALICE uses mainly 3 detectors for barrel tracking: ITS, TPC, TRD + (TOF)**
  - **7 layers ITS** (Inner Tracking System – silicon tracker)
  - **152 pad rows** TPC (Time Projection Chamber)
  - **6 layers TRD** (Transition Radiation Detector)
  - **1 layer TOF** (Time Of Flight Detector)
- **Several major upgrades before Run 3:**
  - The TPC is equipped with a GEM readout
  - The ITS is completely replaced by 7 layers of silicon pixels
  - **Major computing upgrade in the O² project**
    - **Merges online and offline processing in the same software framework. Same code (with different cuts / parameters) running online and offline**
- **Drivers behind design decisions:**
  - Search for rare signals imposes large increase in statistics wrt. Run 1+2
  - Triggered TPC readout insufficient
    - Huge out-of-bunch pile up during the TPC drift time
  - → Need continuous readout

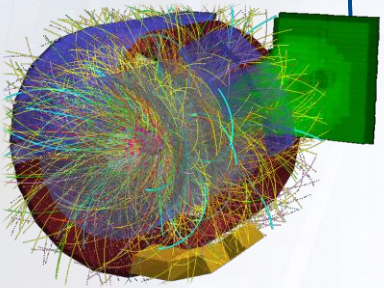


# $O^2$ Processing steps

- **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%**)



- **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

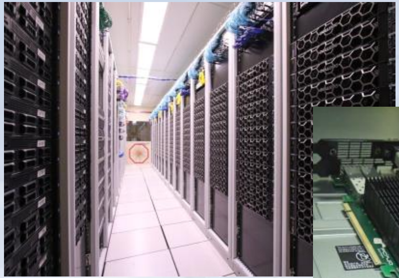
# $O^2$ Project organization

- $O^2$ is composed of 3 projects: EPN, FLP, PDP:

**FLP** Farm in CR1: 202 Servers
Readout + local processing on CPU and FPGA

**EPN** Farm in CR0: 250 Servers / 2000 GPUs
Global processing on CPU and GPU

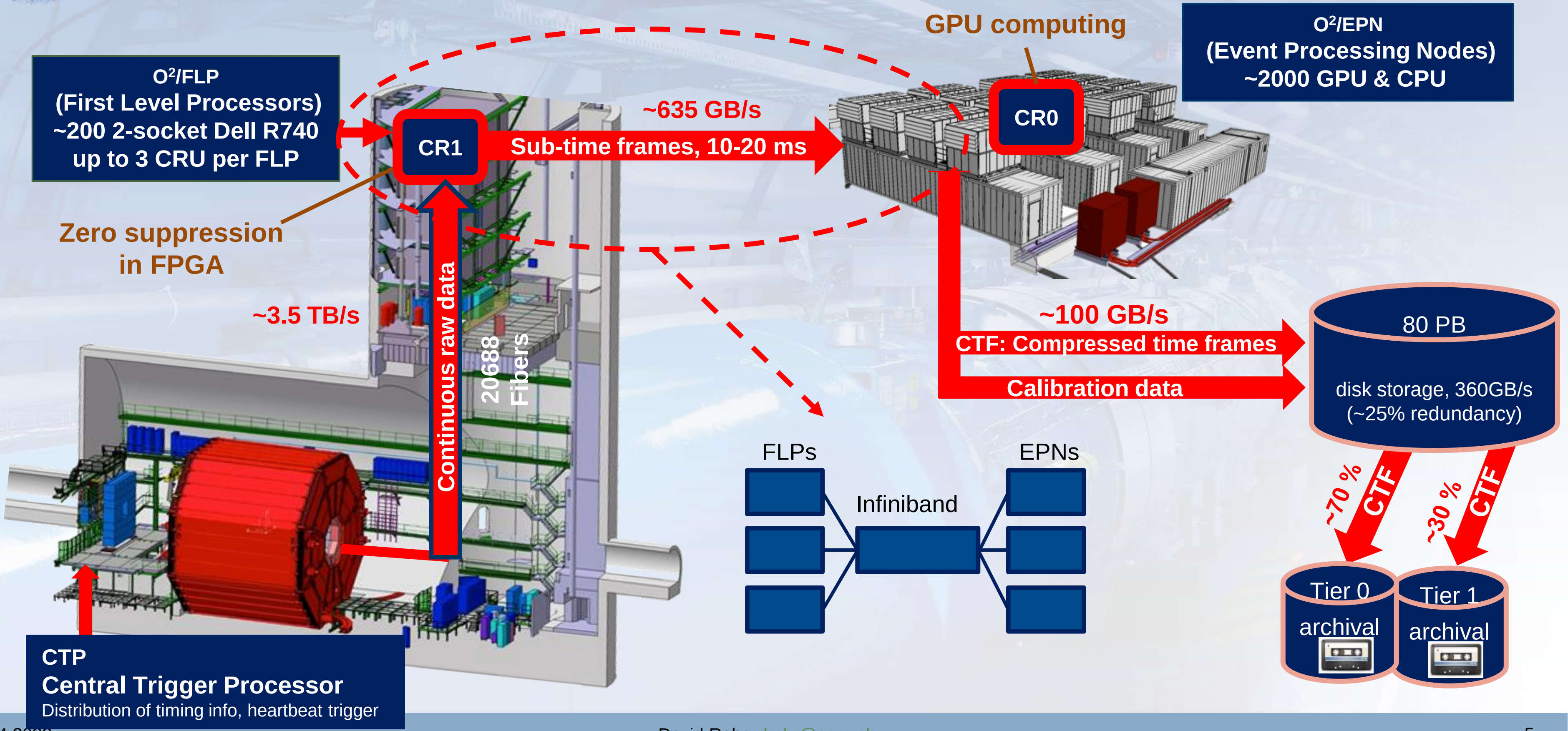
**detector data**



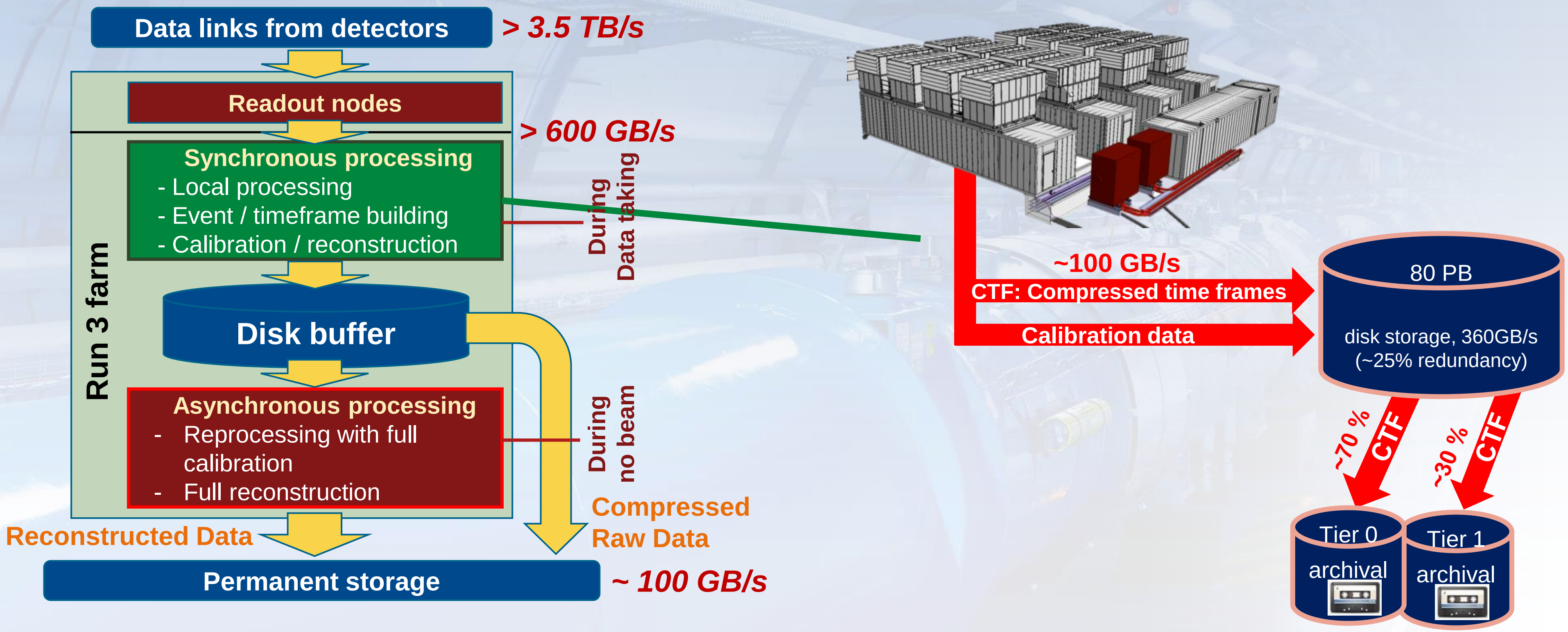
**to storage**

**Reconstruction software developed by PDP**

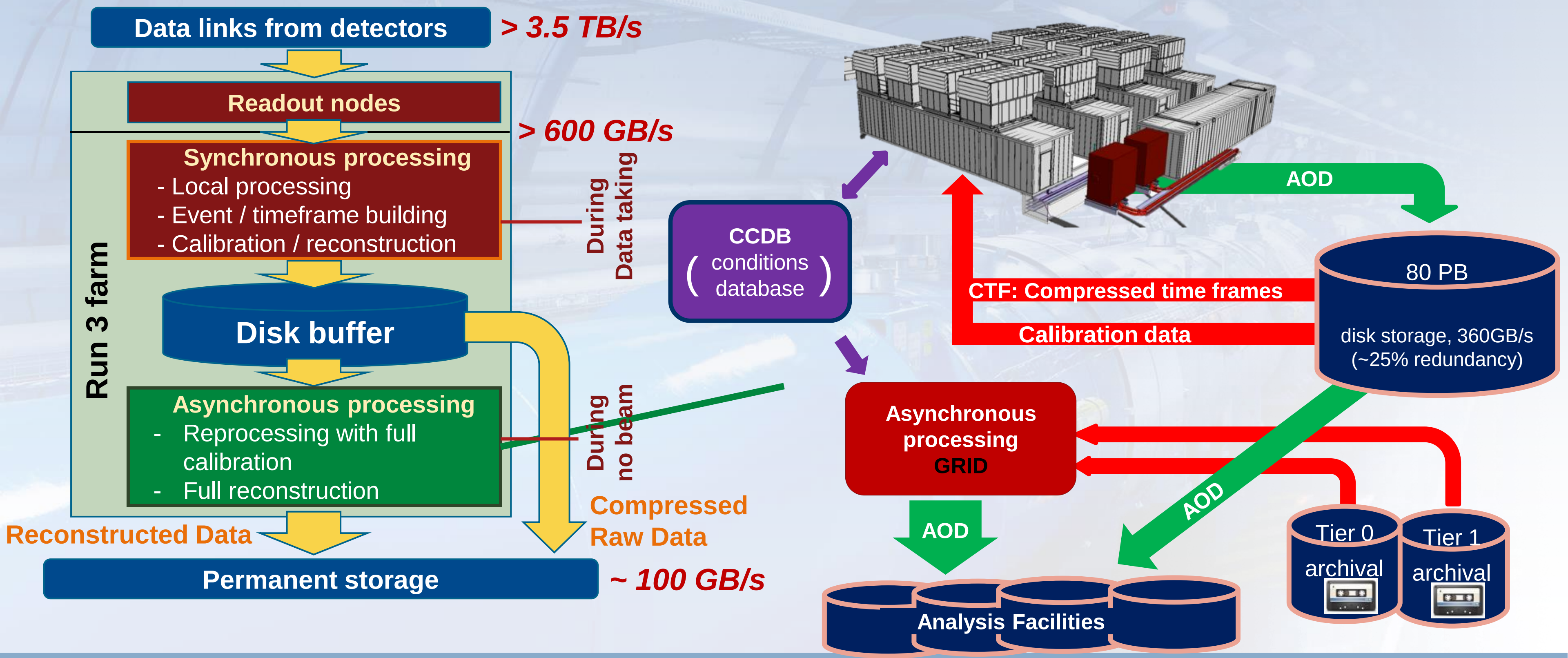
# ALICE Raw Data Flow in Run 3



# Synchronous and Asynchronous Processing



# Synchronous and Asynchronous Processing



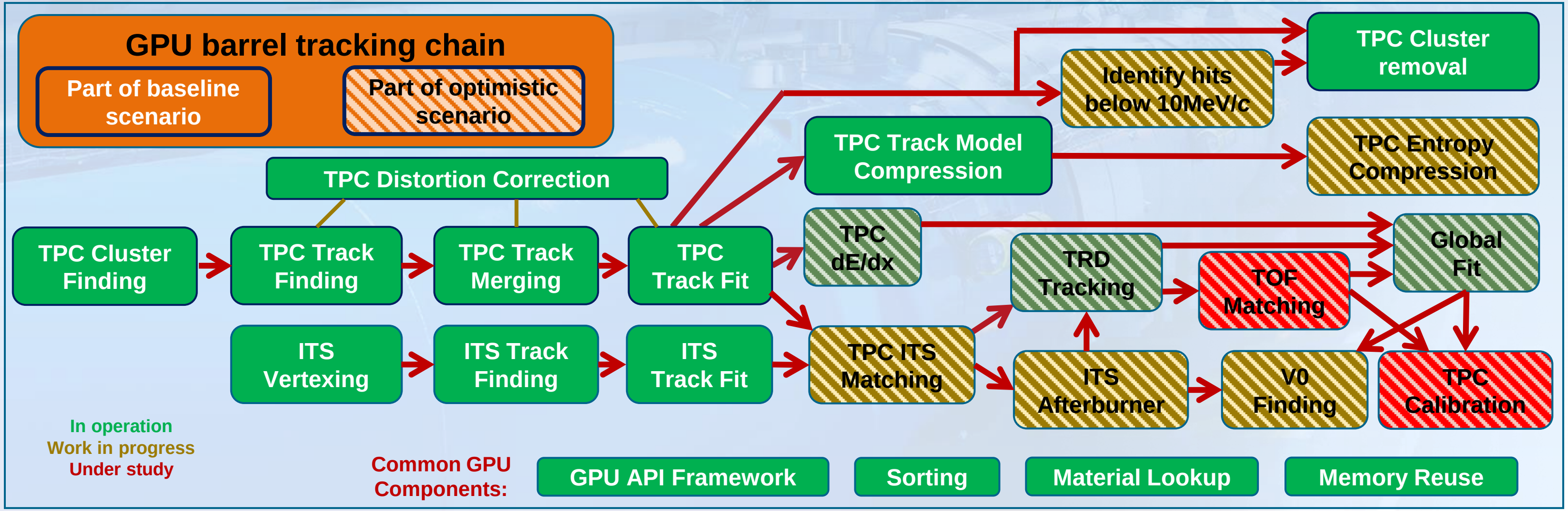
# Usage of GPU for synchronous and async processing

**Baseline GPU solution**
**(fully available today):**
**TPC + part of ITS tracking on GPU**

- This is the mandatory part to keep step with the data taking during the synchronous reconstruction
  - Aiming for ~20% margin
- Caching the raw data is impossible, i.e. if we were not fast enough here, we would need to reduce the event rate

**Optimistic GPU solution**
**(what we are aiming for eventually):**
**Run full barrel tracking on GPU**

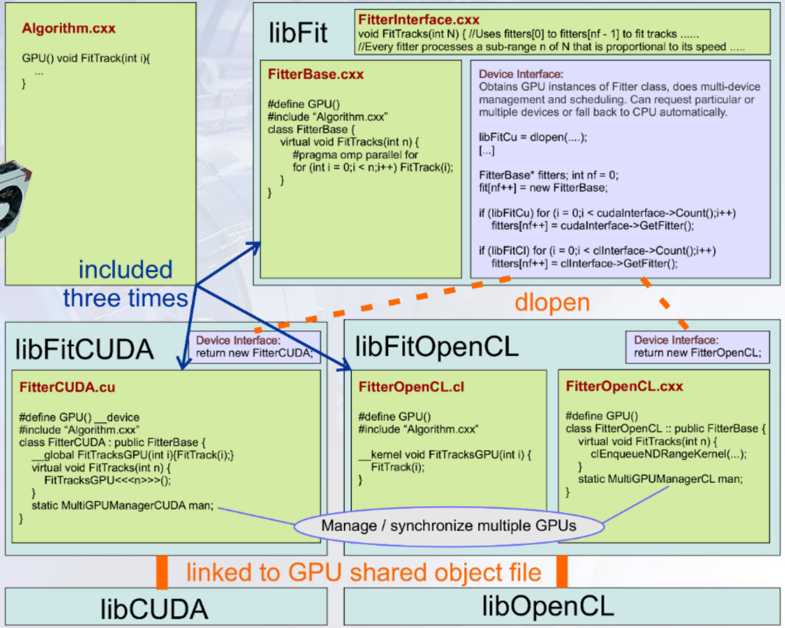
- This aims to make the best use of the GPUs in the asynchronous phase
- Does not affect the synchronous processing
- If we cannot use the GPUs for a large part of the asynchronous reconstruction on the EPN, the processing would be CPU bound while the GPUs would be idle



# Compatibility with several GPU frameworks

- **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)

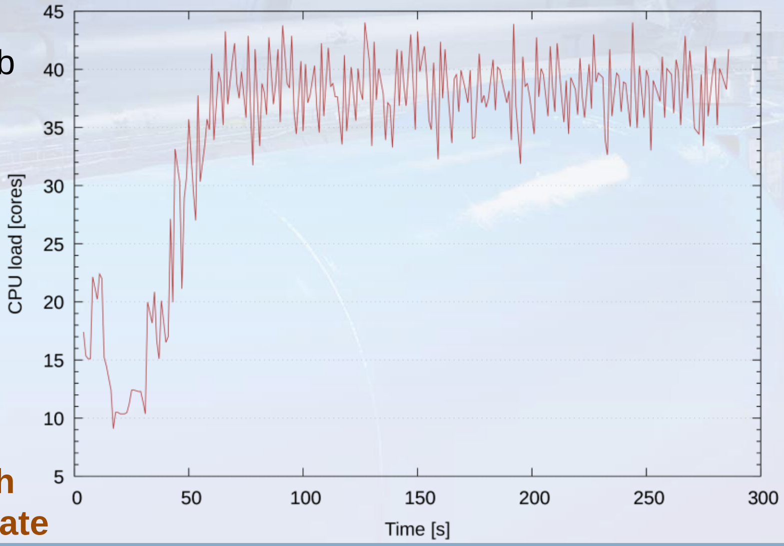
# Synchronous processing full system test results

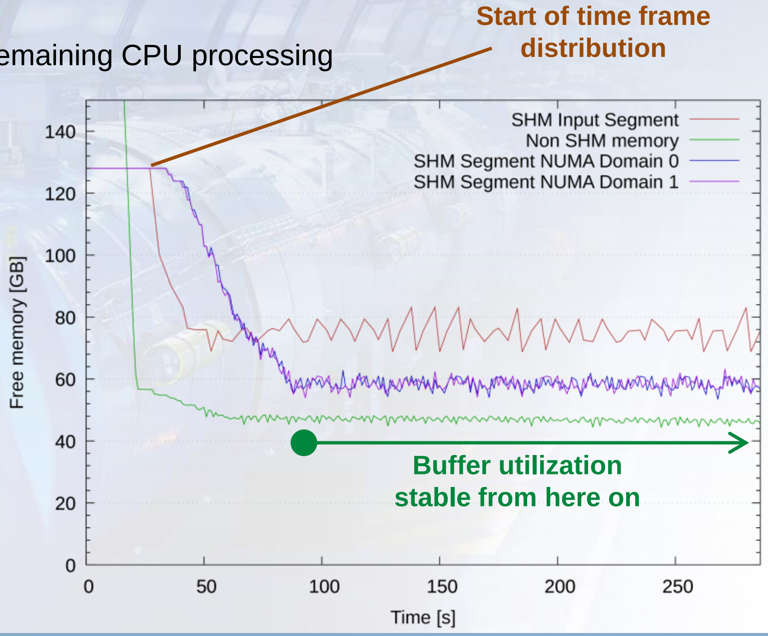
- **Full system test setup**:
  - 1 Supermicro server, 8 * AMD MI50 GPUs, 2 * 32 core Rome CPU, 512 GB of memory
  - Replaying data at 1/250 of the rate expected during 50 kHz Pb-Pb, measuring CPU load, memory load, temperatures
  - If memory doesn't increase over time → no backpressure → server can sustain the rate
- **Load / memory usage**:
  - Max memory consumption **280 GB**, max. CPU load **44 cores**
  - **Final setup** needs **+10 GB** / **6 cores** for the network transfer and **+20%** for remaining CPU processing

Test replaying Pb-Pb data at 50kHz Interaction rate

CPU load and memory utilization reach a plateau after 1-2 minutes of startup time

**→ $O^2$ can cope with highest expected rate**



# Pilot beam data taking

- DISCLAIMER: Obviously I cannot present any physics results here. And no nice plots...
- This my my personal perception of what went well, and what could have been better.

# Pilot beam data taking

- **ALICE is designed for 50 kHz Pb-Pb.**
- **The pilot beam is few kHz pp, if at all.**
  - → Even less interaction rate than what we expect for Pb-Pb.
  - → Computationally, it is not inexpensive, it is basically zero!
  - Or is it?
    - If you have a noisy detector since e.g. a pedestal is wrong / missing, you might be surprised that you cannot reconstruct the “zero” data on your laptop...
    - You might need some extra work to find signals, e.g. an event display without some kind of trigger might show... ...noise and 0 tracks.
- **I think we should not call it “experience” from pilot beam.**
  - Neither “lessons learned” from pilot beam.
  - But rather “reminders” from pilot beam.
    - Many issues have actually been seen before, but solutions have been postponed, neglected, or the problem was just forgotten.
    - Run 2 commissioning is some years in the past...

# Pilot beam data taking

- **In general, one should start as early as possible with global commissioning.**
  - Many issues are considered irrelevant if they appear only every couple of hours.
  - With 20+ systems, there are suddenly multiple problems per hour...
- **Everything that can affect global runs, particularly from central systems (a minor detector might just be switched off, but you cannot turn of the readout), should be investigated.**

- **But these were some global comments – this talk is about software:**
  - ALICE Run 3 reconstruction during the pilot been went (actually to my surprise) rather smooth.
  - I had expected more issues operating on the raw data of so many detectors when they get real beam data for the first time.
    - With so much online processing in modern physics experiments, the online system operations must change.
    - Pure readout usually does not suffer from data-driven problems (except if the data headers are corrupt).
    - In an HLT, you might be able to switch of something that fails for a while.
    - ALICE in Run 3 does full online processing and compression of data. Only the compressed data is stored.
      - If processing fails for one detector, there is no data for that detector – or no data at all (depending on how severely it fails).
  - Rapid development / software deployment turned out paramount!
    - Of course there must be a compromise wrt. stability.

# Pilot beam data taking

- **For software deployment on the farm, we are actually still discussing the final scheme internally, but I could imagine something like:**
  - We can have multiple versions installed in parallel (mandatory) and different partitions of detectors can use different software versions in parallel.
  - There should be one “golden” software that is well maintained, and updated weekly at most (except for critical problems): this provides a baseline for stable global runs (worst case with a feature disabled if it is broken in there).
  - New optional / alternative versions are deployed in parallel on demand.
    - Can be used by detectors in standalone tests (e.g. if they contain a new feature needed by the detector).
    - This is also in some sense an additional layer of online CI: better to see it in a standalone run than to break all global runs.

# Pilot beam data taking

- **What about processing rates / CPU usage / buffer utilization.**
  - I have created a plot for TPC GPU tracking time of pilot beam time frames, but...
    - ... this is not only totally irrelevant with such sparse data, it is even misleading.
  - From this partially noise-dominated data, you cannot simply scale to high rates.
  - Estimations from MC data gives a much better estimate!

- **Software stability can be checked: The pilot beam data set as a whole is a good benchmark for data driven bugs (ok, not for something related to high occupancy).**

- **And, a big chance that came with the pilot beam:**
  - We could run a lot of “offline” software online.
  - The software framework is the same, it is just different parameters to enable some more reconstruction steps / algorithms.
  - Since the compute-farm was basically idle.
  - One example is TPC dEdx.

- **And a big caveat:**
  - Make sure that workarounds developed for the pilot beam, under the assumption there is infinite compute power, do not live long enough to make it into real data taking!

# Summary

- $O^2$ (Online Offline processing) is the online computing scheme for ALICE in Run 3, including hardware and software, covering the data flow from the readout to the final reconstruction results
- **Main reconstruction steps:**
  - **Synchronous processing**: calibration and compression (reconstruction as much as needed)
  - **Asynchronous processing**: full final reconstruction
- **ALICE uses hardware accelerators for the processing**
  - Bulk of reconstruction runs on **GPUs** on the **EPNs**
    - **> 95%** of **synchronous** reconstruction on GPU, aiming for **60% - 85%** of **asynchronous** workload.
  - Local processing on **FPGAs** on the **FLPs**
- **EPN processing tested in full system test at 50 kHz Pb-Pb data rates: successful with 20% margin**
- **O2 successfully handled the processing of the pilot beam**
  - More a test of stability and infrastructure and not of processing performance
  - Computing-wise, the pilot beam went pretty smooth – actually better than expected.
  - This doesn't mean there were no software bugs, but we were able to fix or work around all of them very quickly.
- **Compute-intense reconstruction steps are designed to run on GPU**
  - This uses a **vendor-independent** shared source code approach
  - Can run on CPUs with **OpenMP** and on GPUs via **CUDA**, **OpenCL**, and **ROCm**
- **Synchronous processing deployed in time for pilot beam, now focusing on asynchronous reconstruction**