Online data processing with GPUs in ALICE during LHC Run 3

David Rohr

GPU Day

20.6.2022, Budapest

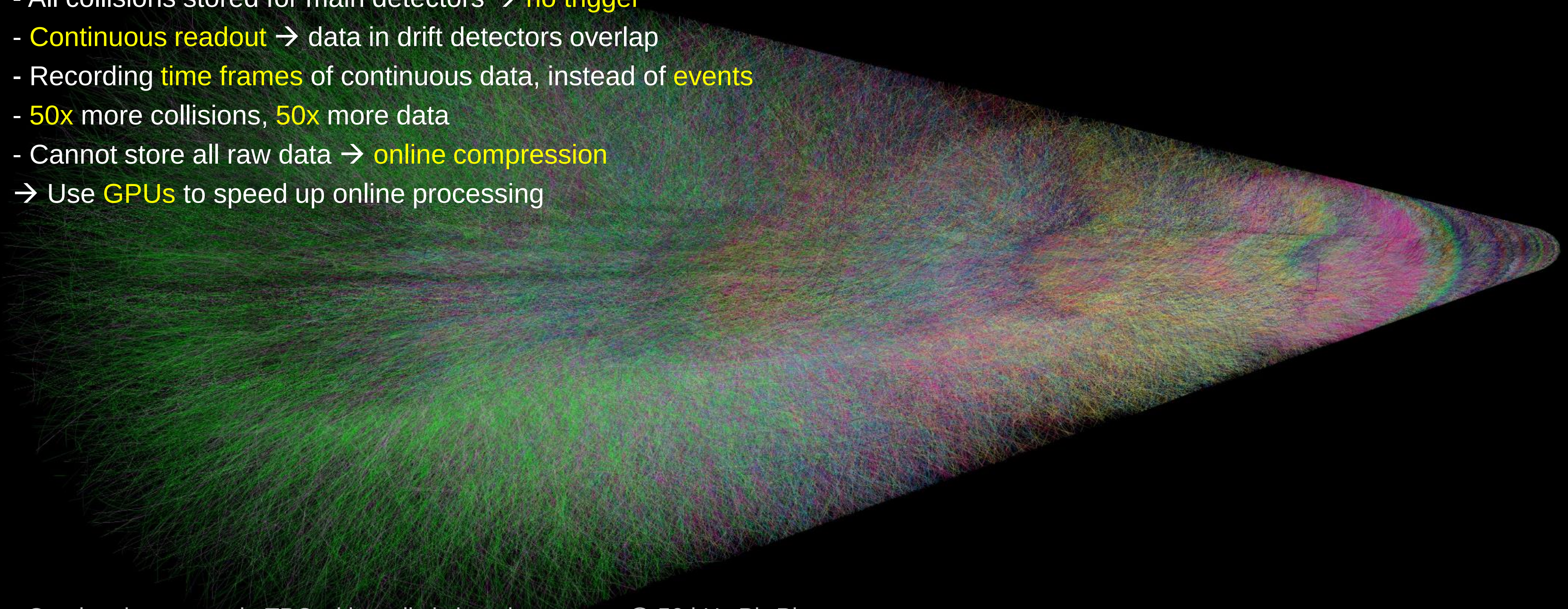
drohr@cern.ch

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

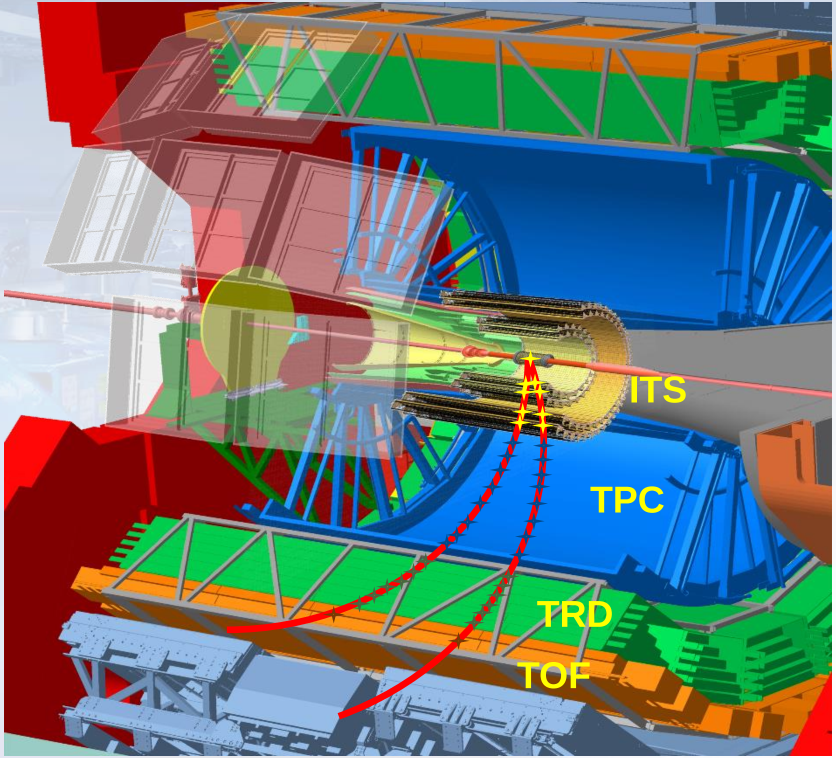
→ Use GPUs to speed up online processing



- 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.

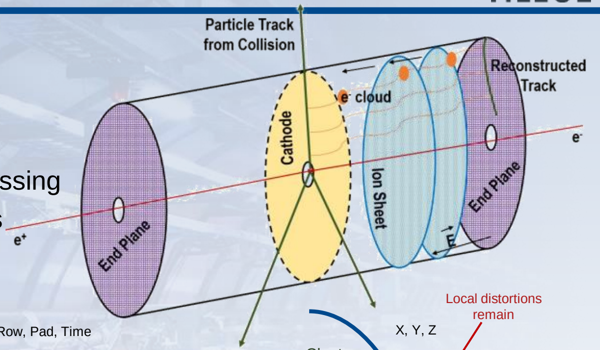
The ALICE detector in Run 3

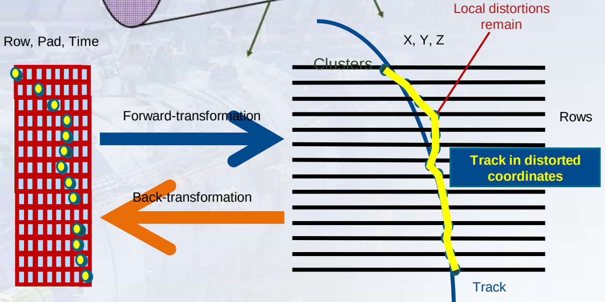
- **ALICE uses mainly 3 detectors for barrel tracking: ITS, TPC, TRD + (TOF)**
  - **7 layers ITS** (Inner Tracking System – silicon tracker)
  - **152 pad rows** TPC (Time Projection Chamber)
  - **6 layers TRD** (Transition Radiation Detector)
  - **1 layer TOF** (Time Of Flight Detector)
- **Several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

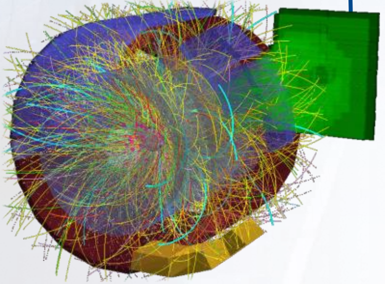


O² Processing steps

- **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%**)







- **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

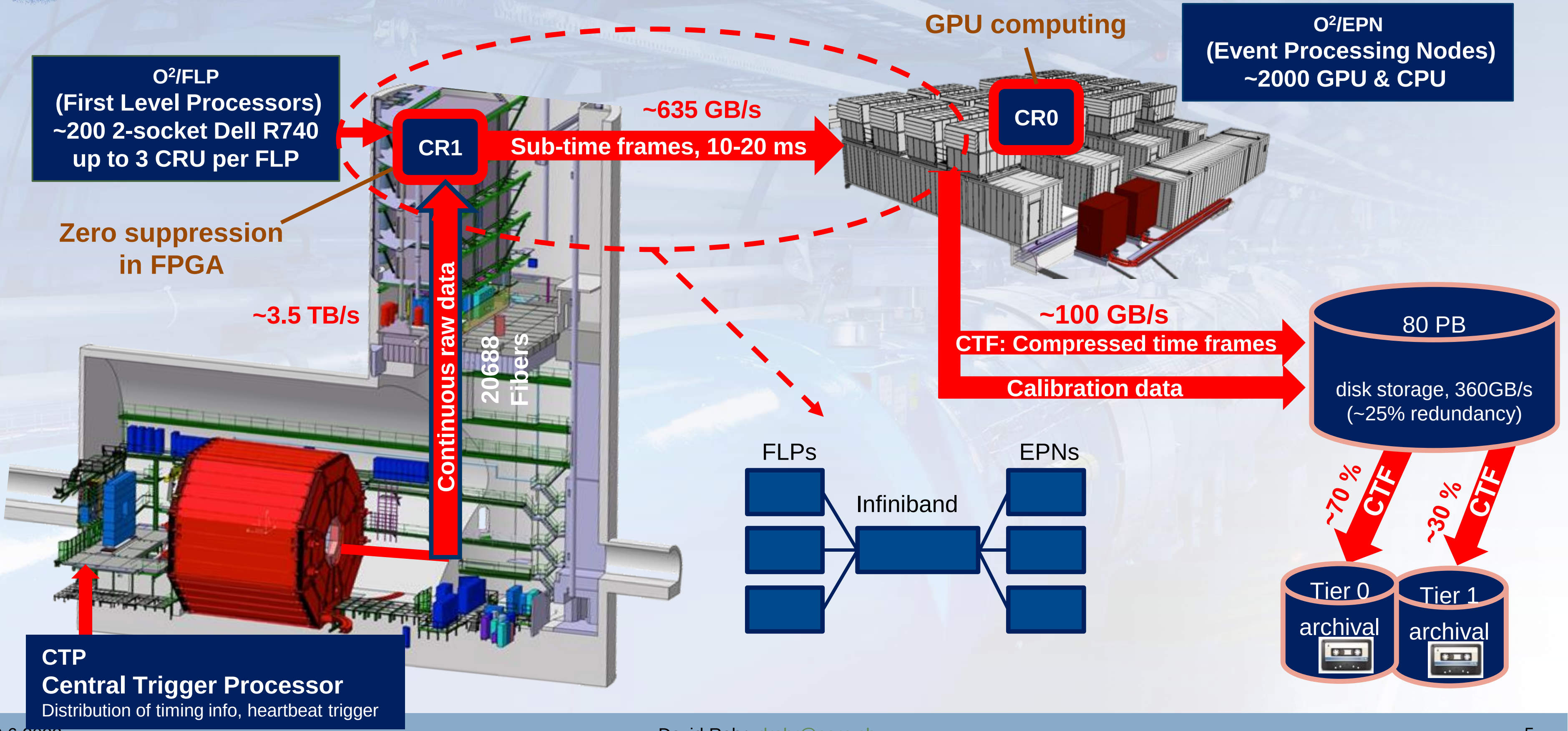
EPN Farm in CR0: 250 Servers / 2000 GPUs
Global processing on CPU and GPU

detector data

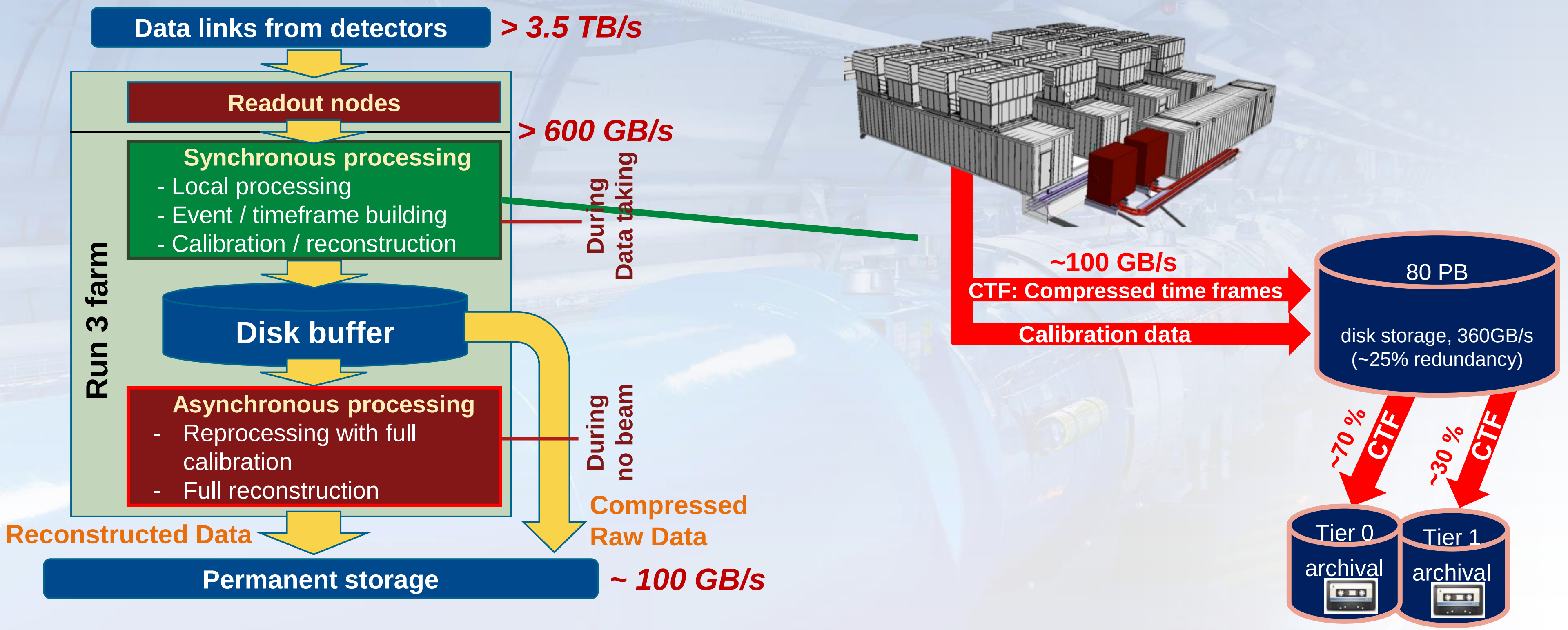
to storage

Reconstruction software developed by PDP

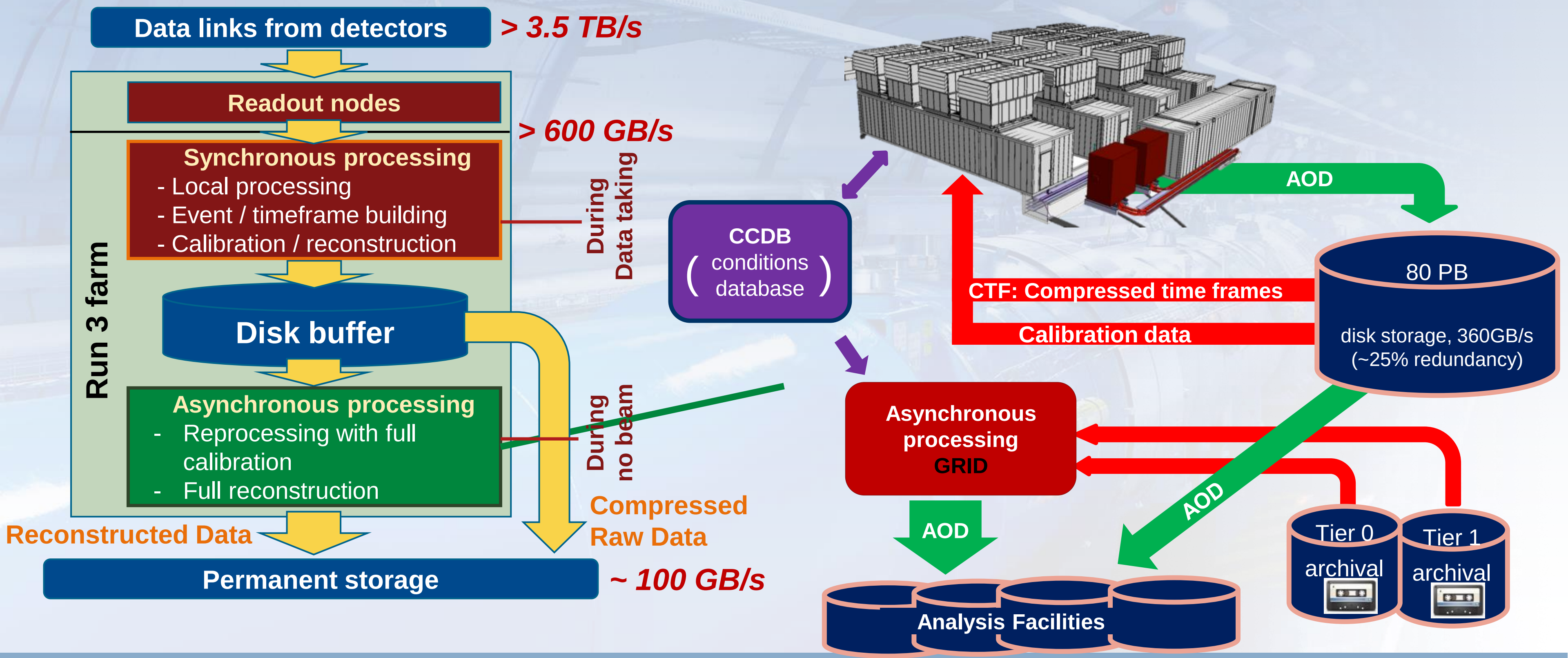
ALICE Raw Data Flow in Run 3



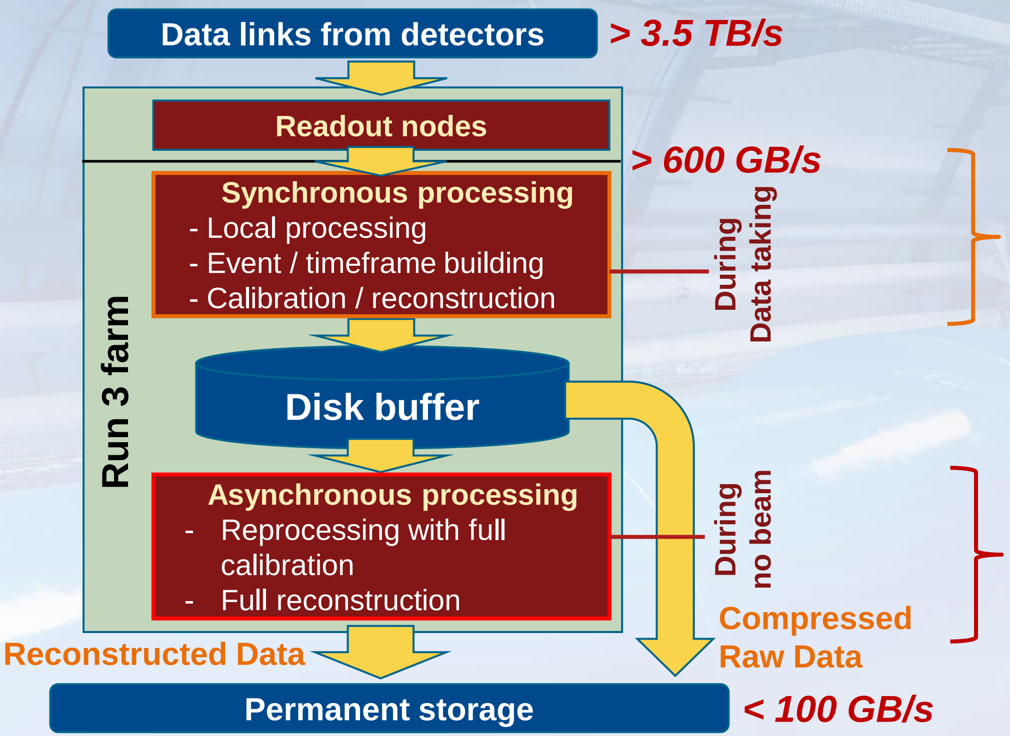
Synchronous and Asynchronous Processing



Synchronous and Asynchronous Processing



Synchronous and Asynchronous Processing



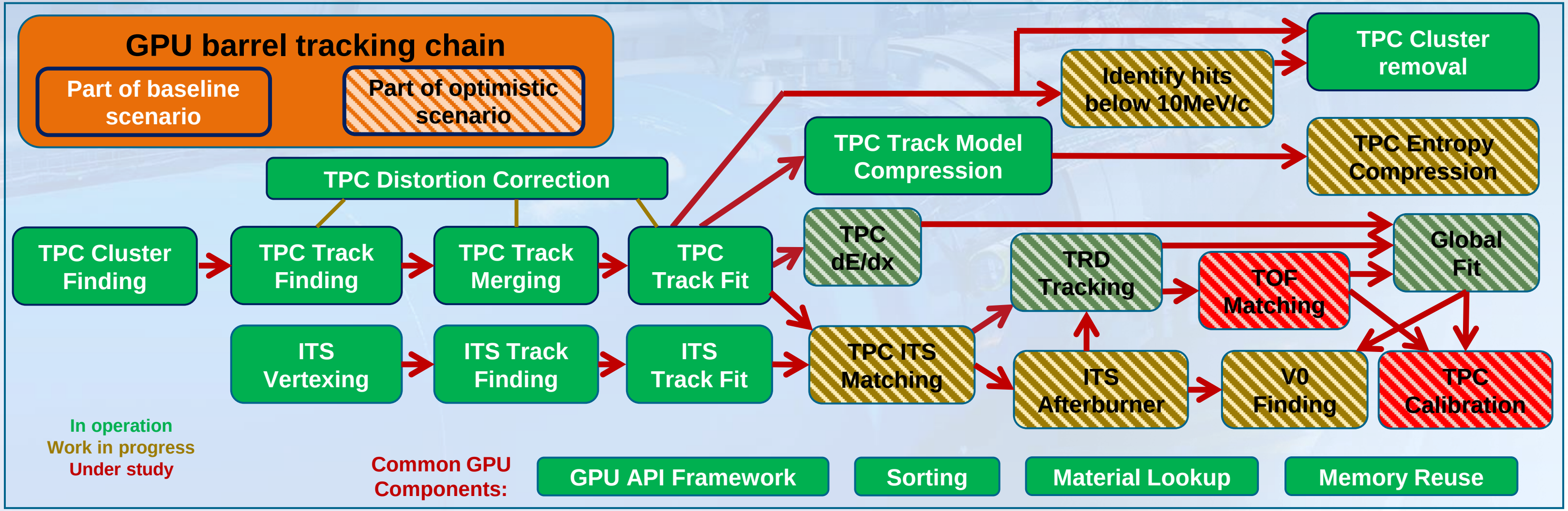
- **Calibration: Tracking for ITS / TPC / TRD / TOF for ~1% of tracks.**
- **Data compression: track-model compression requires full TPC tracking for all collisions.**
  - → TPC tracking dominant workload during synchronous reconstruction.
  - → Well suited to run on **GPUs**, EPN farm designed for best TPC clusterization / tracking / compression performance.

- **No clear single computational hot-spot.**
- **TPC reconstruction important but not dominant.**
  - Actually faster than in the synchronous phase: no clusterization / no compression / less hits after hit removal in synchronous phase overcompensates the slowdown of more elaborate fits.
- **Full reconstruction for all other detectors.**

→ **More heterogeneous workload.**

Central barrel global tracking chain

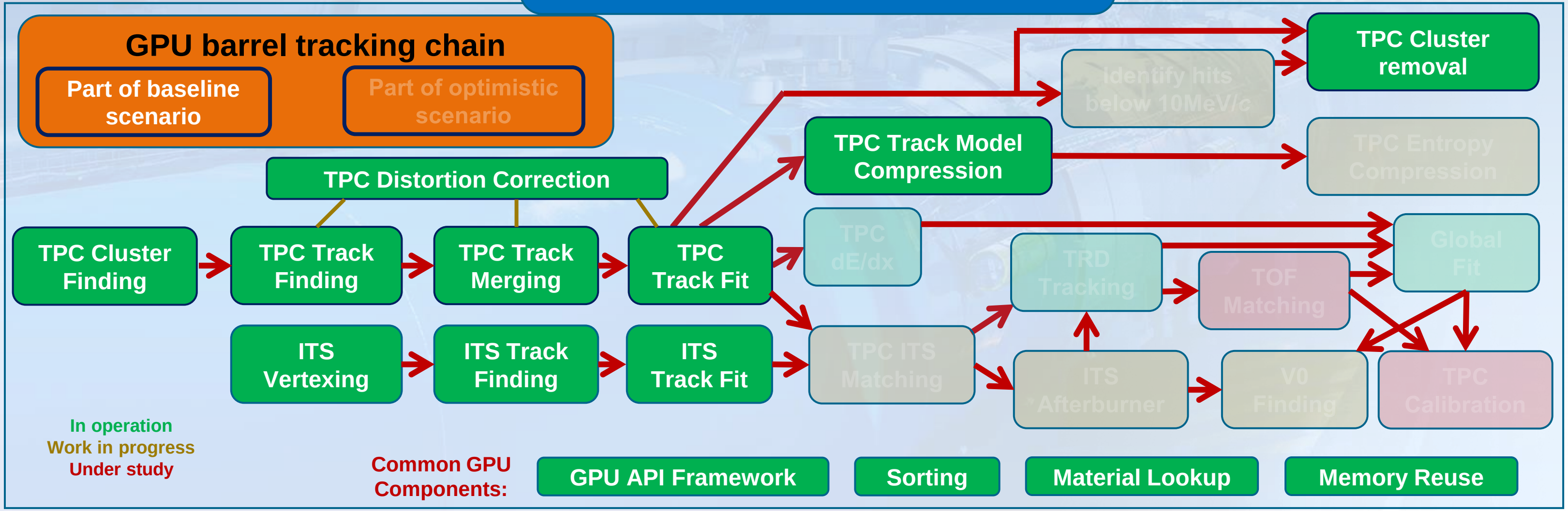
- **Overview of reconstruction steps considered for GPU-offload:**
  - Mandatory **baseline scenario** includes everything that must run on the GPU during synchronous reconstruction.
  - **Optimistic scenario** includes everything related to the barrel tracking.



Central barrel global tracking chain

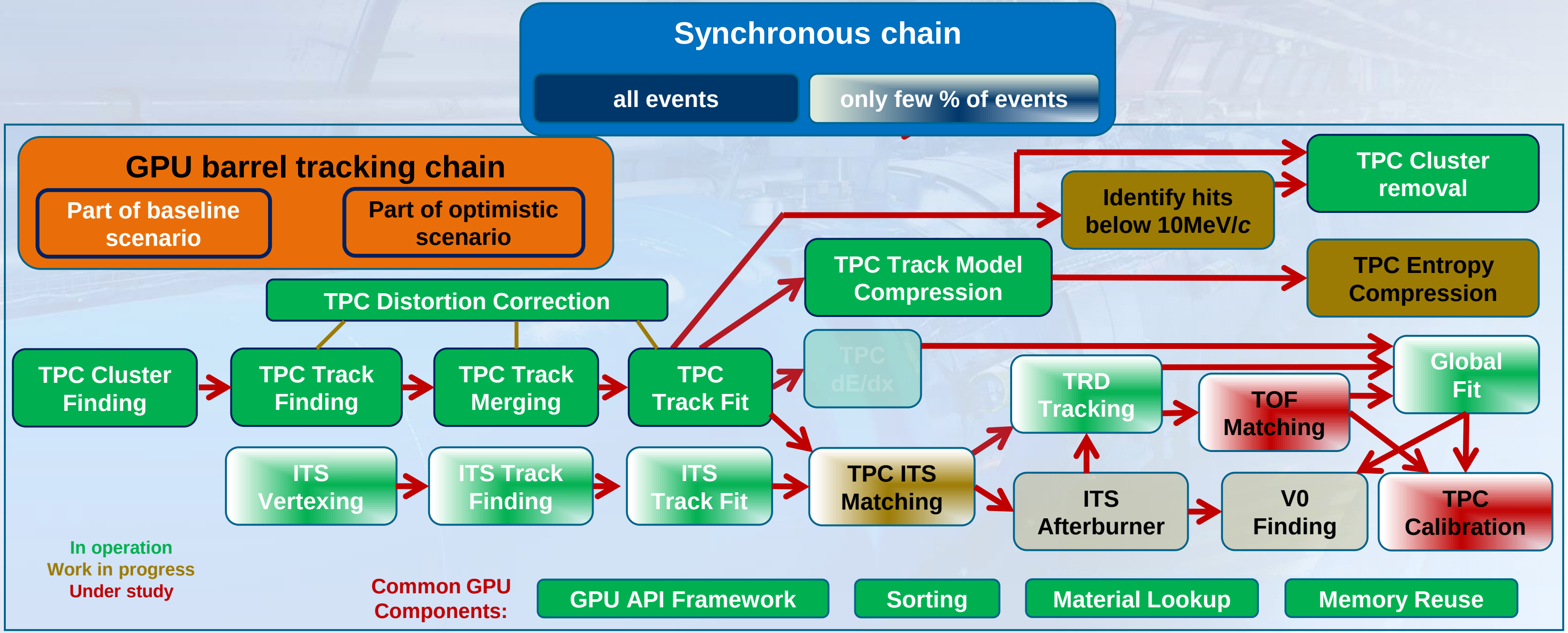
- **Baseline scenario fully implemented (module some improvements e.g. distrotion correction).**
  - Not mandatory to speed up the synchronous GPU code further, but we should try nonetheless.
  - If we add / improve reconstruction steps, we have to speed it up accordingly to remain in the 2000 GPU budget.
  - Worst case, can always trade higher speed for worse tracking resolution and less compression.
    - Risky in compression strategy B (see later).

Baseline scenario
(ready except for 1 optional component)



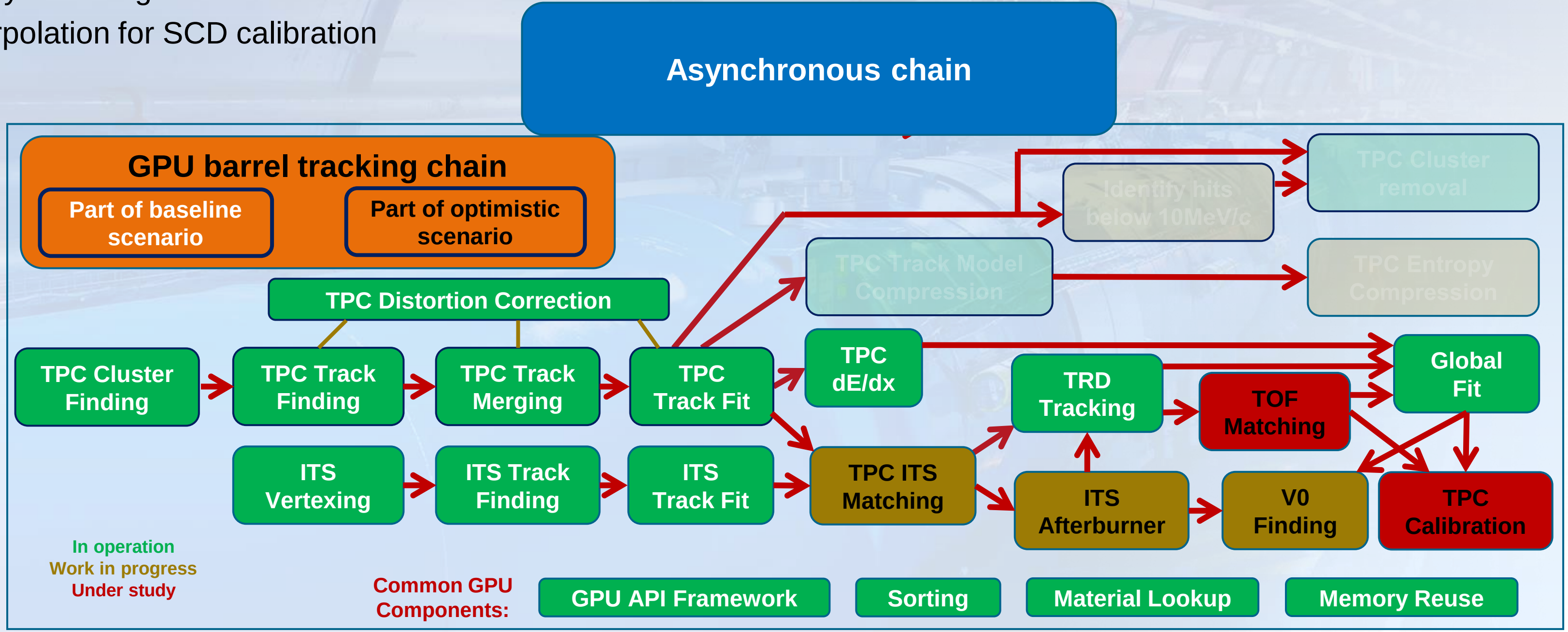
Central barrel global tracking chain

- **Baseline scenario fully implemented (module some improvements e.g. distrotion correction).**
  - 2 optional parts still being investigated for sync. reco on GPU: TPC entropy encoding / Looper identification < 10 MeV.

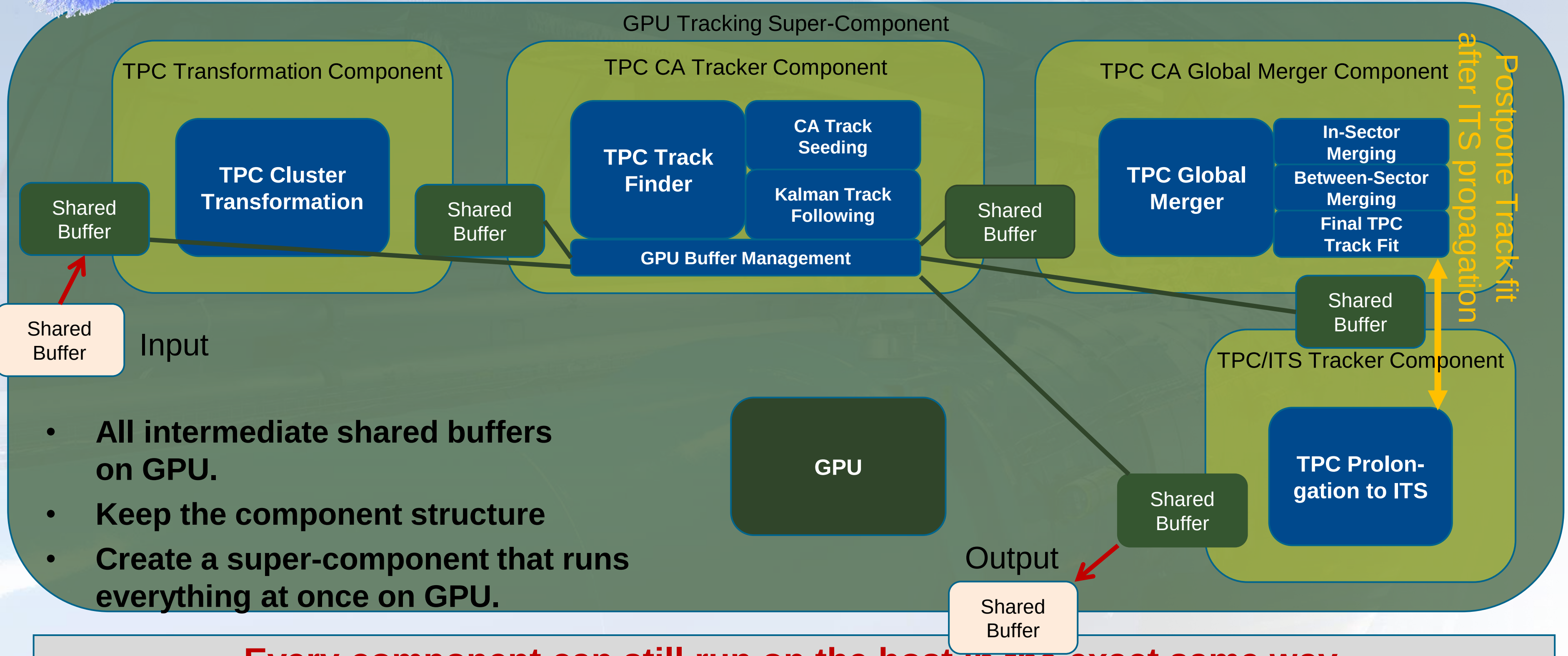


Central barrel global tracking chain

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



Message passing based approach, on host and GPU



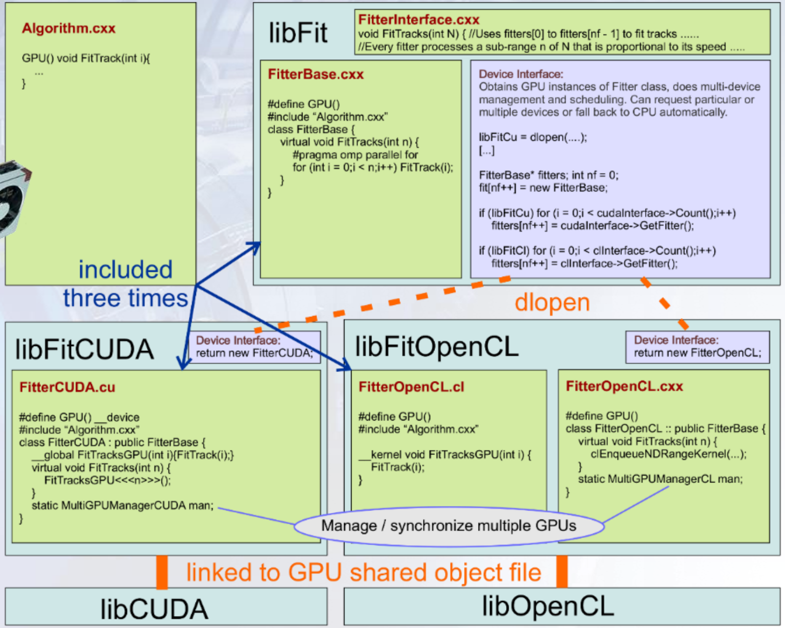
- **All intermediate shared buffers on GPU.**
- **Keep the component structure**
- **Create a super-component that runs everything at once on GPU.**

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

Compatibility with several GPU frameworks

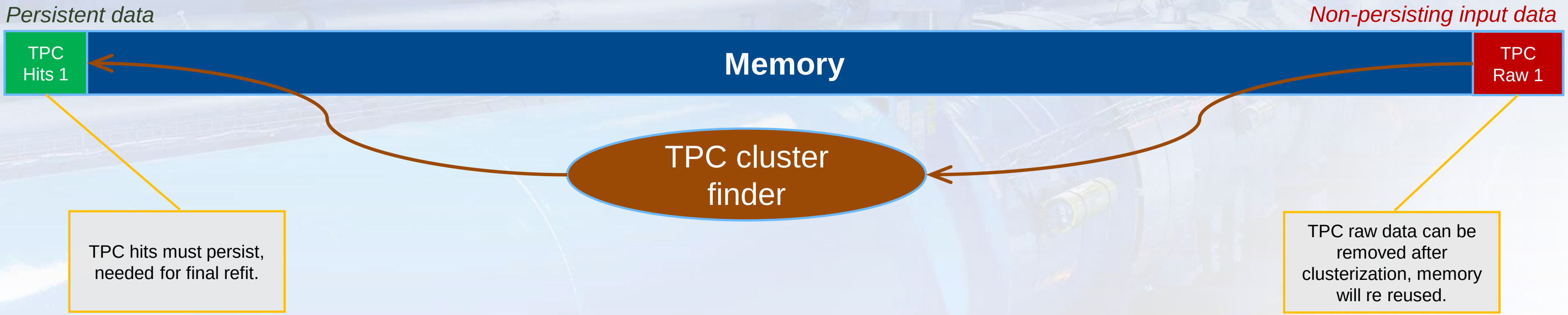
- **Generic common C++ Code compatible to CUDA, OpenCL, HIP, and CPU (with pure C++, OpenMP, or OpenCL).**
  - OpenCL needs clang compiler (ARM or AMD ROCm) or AMD extensions (TPC track finding only on Run 2 GPUs and CPU for testing)
  - Certain worthwhile algorithms have a vectorized code branch for 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)



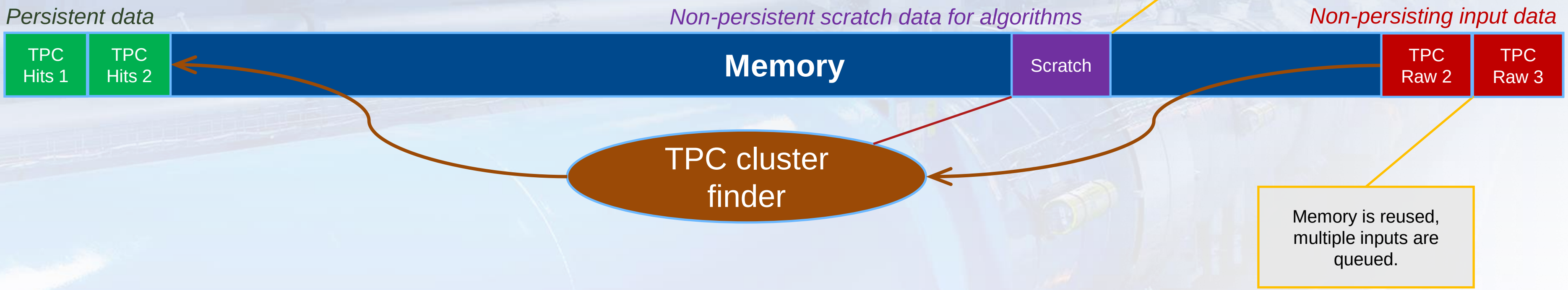
Memory requirements

- **ALICE reconstructs timeframes (TF) independently (**128 – 256 orbits → ~10 - ~20 ms → ~500 - ~1000 collisions**).**
  - One TPC drift time of data not reconstructible at TF border (~ 90 us) → < 0.5 – 1 % of statistics lost (baseline is 0.5 %).
  - Timeframe should fit in GPU memory. If not, could use kind of ring buffer, or reduce TF length to 128 orbits.
  - Trying to avoid the ring buffer approach, could be added later if needed.
- **Custom allocator: grabs all GPU memory, gives out chunks manually, memory will be reused when possible.**
  - Classically: reuse memory between events, collisions are not that large.
  - ALICE reuses memory between different algorithms in a TF, possibly also between independent collisions.
  - Some memory must persist during timeframe processing.



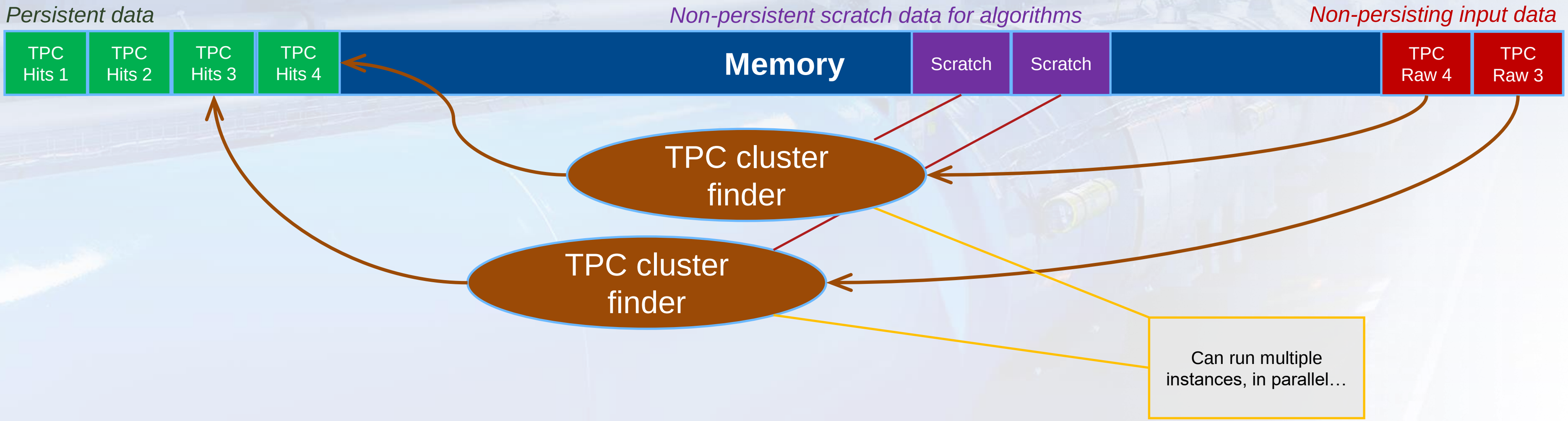
Memory requirements

- **ALICE reconstructs timeframes (TF) independently (128 – 256 orbits → ~10 - ~20 ms → ~500 - ~1000 collisions).**
  - One TPC drift time of data not reconstructible at TF border (~ 90 us) → < 0.5 – 1 % of statistics lost (baseline is 0.5 %).
  - Timeframe should fit in GPU memory. If not, could use kind of ring buffer, or reduce TF length to 128 orbits.
  - Trying to avoid the ring buffer approach, could be added later if needed.
- **Custom allocator: grabs all GPU memory, gives out chunks manually, memory will be reused when possible.**
  - Classically: reuse memory between events, collisions are not that large.
  - ALICE reuses memory between different algorithms in a TF, possibly also between independent collisions.
  - Some memory must persist during timeframe processing.



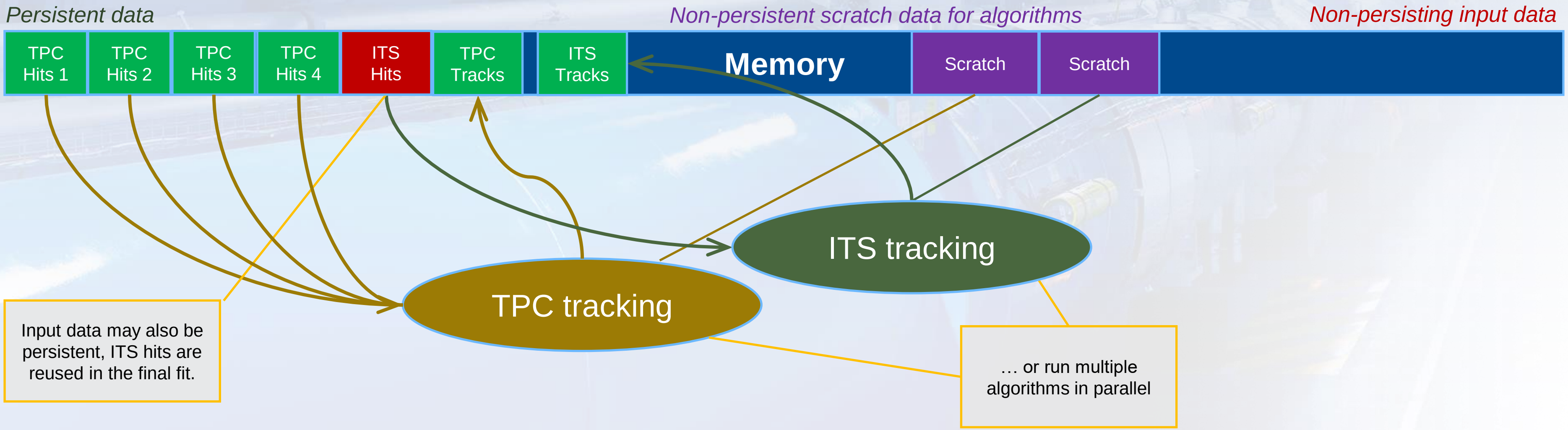
Memory requirements

- **ALICE reconstructs timeframes (TF) independently (128 – 256 orbits → ~10 - ~20 ms → ~500 - ~1000 collisions).**
  - One TPC drift time of data not reconstructible at TF border (~ 90 us) → < 0.5 – 1 % of statistics lost (baseline is 0.5 %).
  - Timeframe should fit in GPU memory. If not, could use kind of ring buffer, or reduce TF length to 128 orbits.
  - Trying to avoid the ring buffer approach, could be added later if needed.
- **Custom allocator: grabs all GPU memory, gives out chunks manually, memory will be reused when possible.**
  - Classically: reuse memory between events, collisions are not that large.
  - ALICE reuses memory between different algorithms in a TF, possibly also between independent collisions.
  - Some memory must persist during timeframe processing.



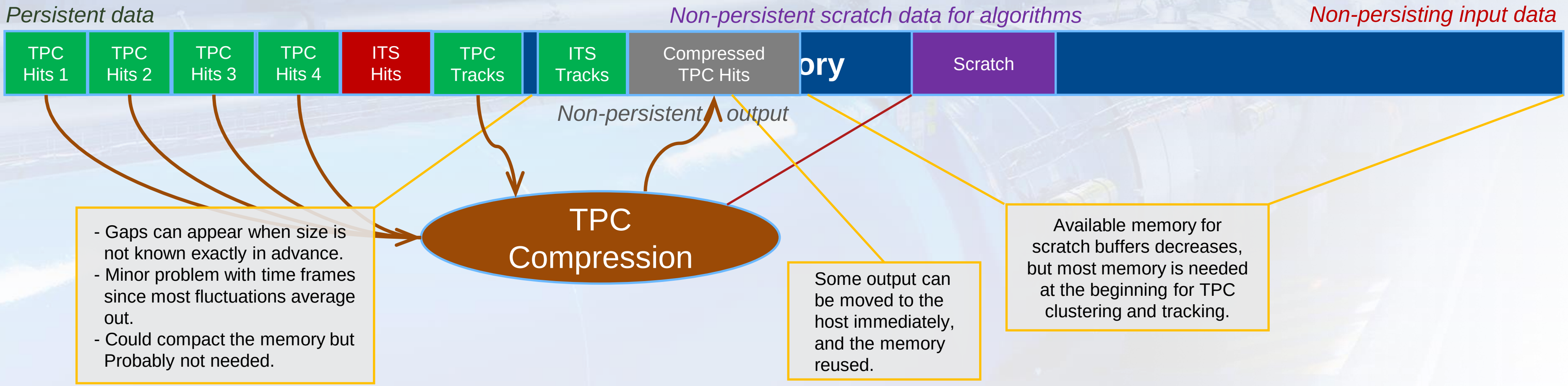
Memory requirements

- **ALICE reconstructs timeframes (TF) independently (128 – 256 orbits → ~10 - ~20 ms → ~500 - ~1000 collisions).**
  - One TPC drift time of data not reconstructible at TF border (~ 90 us) → < 0.5 – 1 % of statistics lost (baseline is 0.5 %).
  - Timeframe should fit in GPU memory. If not, could use kind of ring buffer, or reduce TF length to 128 orbits.
  - Trying to avoid the ring buffer approach, could be added later if needed.
- **Custom allocator: grabs all GPU memory, gives out chunks manually, memory will be reused when possible.**
  - Classically: reuse memory between events, collisions are not that large.
  - ALICE reuses memory between different algorithms in a TF, possibly also between independent collisions.
  - Some memory must persist during timeframe processing.



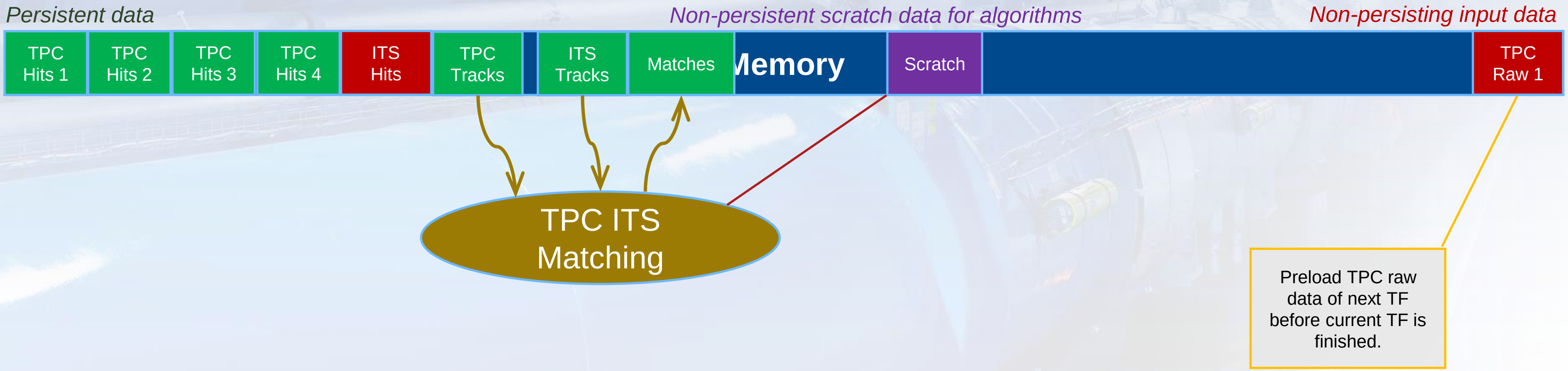
Memory requirements

- **ALICE reconstructs timeframes (TF) independently (128 – 256 orbits → ~10 - ~20 ms → ~500 - ~1000 collisions).**
  - One TPC drift time of data not reconstructible at TF border (~ 90 us) → < 0.5 – 1 % of statistics lost (baseline is 0.5 %).
  - Timeframe should fit in GPU memory. If not, could use kind of ring buffer, or reduce TF length to 128 orbits.
  - Trying to avoid the ring buffer approach, could be added later if needed.
- **Custom allocator: grabs all GPU memory, gives out chunks manually, memory will be reused when possible.**
  - Classically: reuse memory between events, collisions are not that large.
  - ALICE reuses memory between different algorithms in a TF, possibly also between independent collisions.
  - Some memory must persist during timeframe processing.

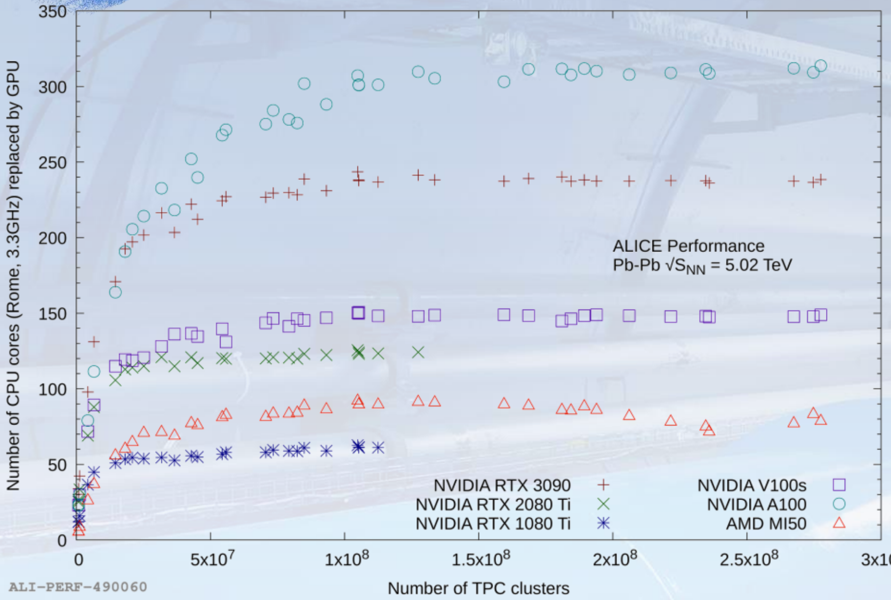


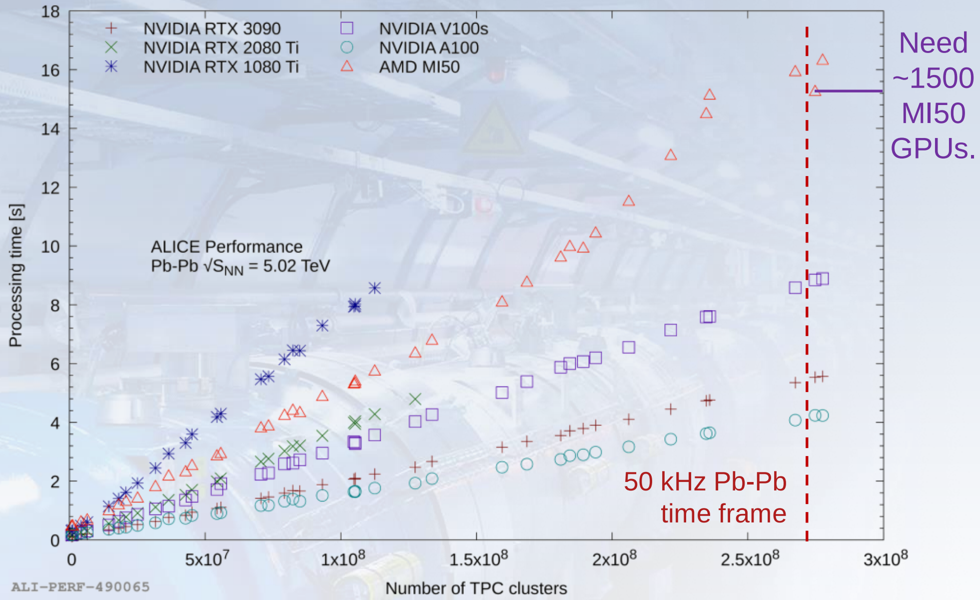
Memory requirements

- **ALICE reconstructs timeframes (TF) independently (128 – 256 orbits → ~10 - ~20 ms → ~500 - ~1000 collisions).**
  - One TPC drift time of data not reconstructible at TF border (~ 90 us) → < 0.5 – 1 % of statistics lost (baseline is 0.5 %).
  - Timeframe should fit in GPU memory. If not, could use kind of ring buffer, or reduce TF length to 128 orbits.
  - Trying to avoid the ring buffer approach, could be added later if needed.
- **Custom allocator: grabs all GPU memory, gives out chunks manually, memory will be reused when possible.**
  - Classically: reuse memory between events, collisions are not that large.
  - ALICE reuses memory between different algorithms in a TF, possibly also between independent collisions.
  - Some memory must persist during timeframe processing.



GPU Performance (standalone benchmark)





- **MI50 GPU** replaces **~80 Rome cores** in **synchronous** reconstruction.
  - Includes **TPC clusterization**, which is **not optimized** for the CPU!
- **~55 CPU cores** in **asynchronous** reconstruction (more realistic comparison).

GPU Model	Performance		GPU Model	Performance
NVIDIA RTX 2080 Ti	100.0%		NVIDIA V100s	122.7%
NVIDIA Quadro RTX 6000 (active)	105.8%		NVIDIA RTX 3090	187.3%
NVIDIA Quadro RTX 6000 (passive)	96.1%		NVIDIA T4	59.3%
NVIDIA RTX 2080	83.5%		AMD MI50	67.8%
NVIDIA GTX 1080	60.1%		AMD Radeon 7	71,2%

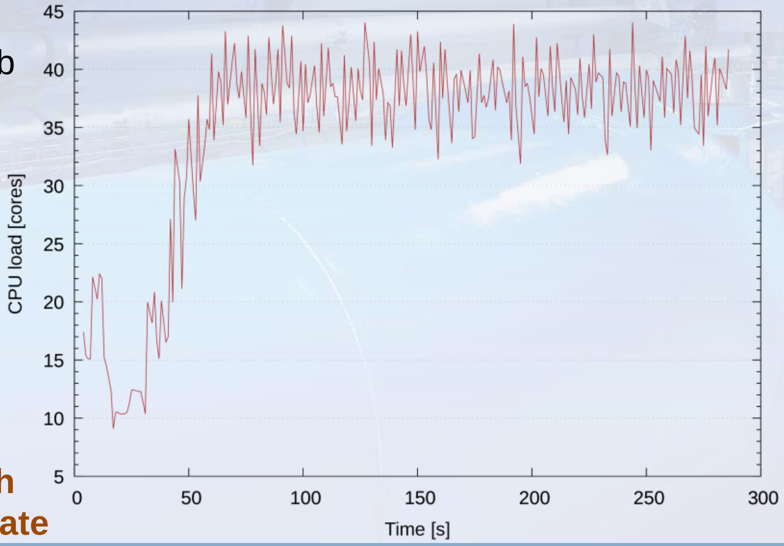
Synchronous processing full system test results

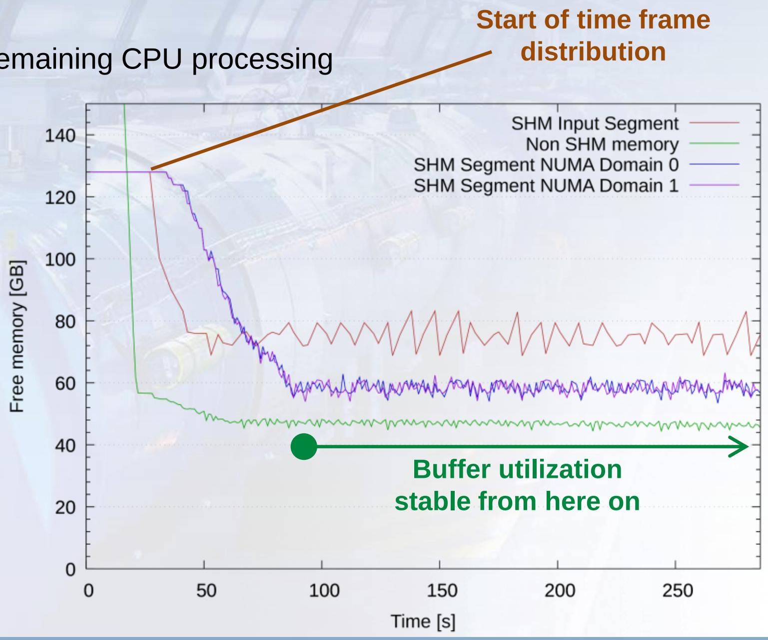
- **Full system test setup**:
  - 1 Supermicro server, 8 * AMD MI50 GPUs, 2 * 32 core Rome CPU, 512 GB of memory
  - Replaying data at 1/250 of the rate expected during 50 kHz Pb-Pb, measuring CPU load, memory load, temperatures
  - If memory doesn't increase over time → no backpressure → server can 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

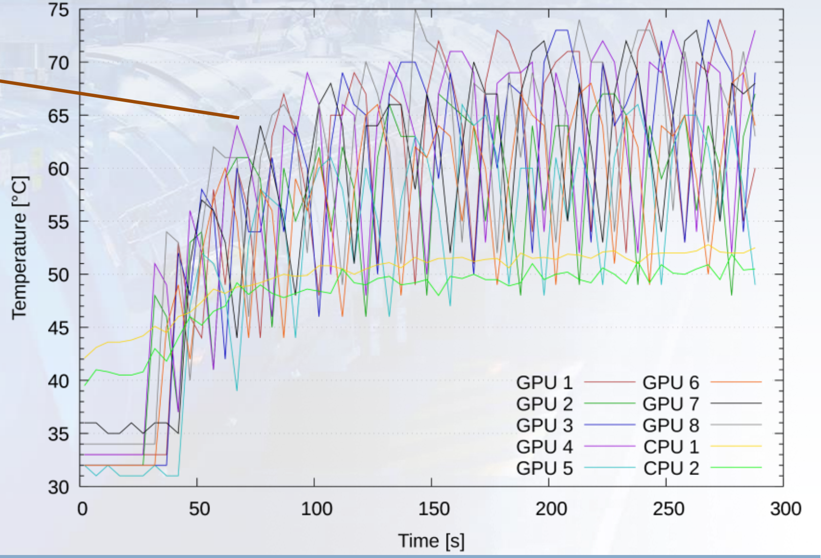




Synchronous processing full system test results

- **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.
- **Temperatures**:
  - Max GPU temperature: **75 °C**.
  - Max CPU temperature: **53 °C**.
  - (Environmental temperature: 21 °C.)

GPU temperature drops in idle phases due to rate limitation.



Fraction of workload that can use the GPU

- **Majority of synchronous processing already on GPU:**
  - GPUs fully loaded during synchronous processing
  - Perfect use of EPN farm
- **Trying to optimize GPU usage during asynchronous phase**
  - **Work in progress!**
  - **Software and threading optimizations not final yet**
  - **For a fair comparison, which fraction we can run on a GPU, we need an optimized CPU reference**
    – Can run each step on the host fully-multithreaded: Guarantees to use all cores all the time, but large granularity effects
    – Can use less cores and parallelize over processing steps: higher efficiency per processing step, but more difficult to use all cores (insufficient memory)
  - **1st corner case**: full parallel processing of all steps, 1 time frame at a time:
    – > **85%** of workload currently already on GPU
  - **2nd corner case**: everything single-threaded on CPU:
    – Only 60% of workload can currently run on GPU
    – But everything single-threaded not feasible due to insufficient memory
  - Truth is somewhere in between, with moderate threading
    – Currently work in progress to obtain good CPU reference measurements
    – Of course also still work in progress: further general software optimizations
  - For reference: for our software, around 90% of the compute capacity of the EPNs is provided by the GPU → **no need to aim for >90%**

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.**
  - The synchronous reconstruction is fully dominated by the TPC (99%) which already fully runs on the GPU, some more processes might follow.
  - In the asynchronous reconstruction, more than 80% is already on the GPU is the baseline scenario, and it would be 95% in the optimistic scenario.

Synchronous processing

Processing step	% of time
TPC Processing	99.37 %
EMCAL Processing	0.20 %
ITS Processing	0.10 %
TPC Entropy Coder	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 %
FIT Entropy Coder	0.01 %
TOF Entropy Coder	0.01 %
MFT Entropy Coder	0.01 %
TPC Calibration residual extraction	0.01 %
TOF Processing	0.01 %

Asynchronous processing

Processing step	% of time
TPC Processing	72.01 %
TRD Tracking	12.69 %
TOF-TPC Matching	9.94 %
MFT Tracking	1.69 %
ITS Tracking	0.78 %
TPC Entropy Decoder	0.73 %
Secondary Vertexing	0.69 %
ITS-TPC Matching	0.56 %
Primary Vertexing	0.14 %
TOF Global Matching	0.11 %
PHOS / CPV Entropy Decoder	0.10 %
FIT Entropy Decoder	0.10 %
ITS Entropy Decoder	0.06 %
TOF Entropy Decoder	0.05 %
MFT Entropy Decoder	0.05 %

Running on GPU in baseline scenario | Running on GPU in optimistic scenario | Preliminary numbers: some algorithms not yet complete or not optimized!

21.5.2021 David Rohr, drohr@cern.ch 10

Measurement for first corner case from May 2021
Fully multi-threaded CPU processing
(not ideal from efficiency standpoint)

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
    - Depending on the measurement, **60% - 85%** of **asynchronous** workload on the **GPU**
      - Still work in progress, aiming to further improve the GPU usage
  - 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
- **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**