GPU-accelerated track reconstruction in the ALICE High Level Trigger

David Rohr *for the ALICE Collaboration*
Frankfurt Institute for Advanced Studies

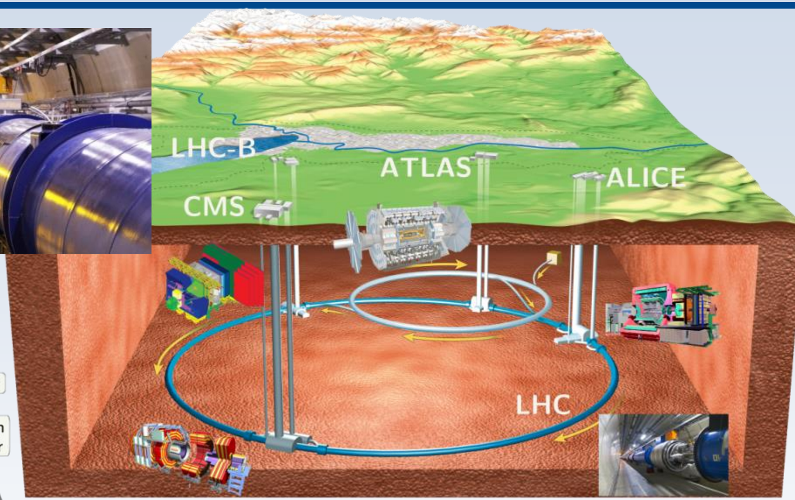
CHEP 2016, San Francisco

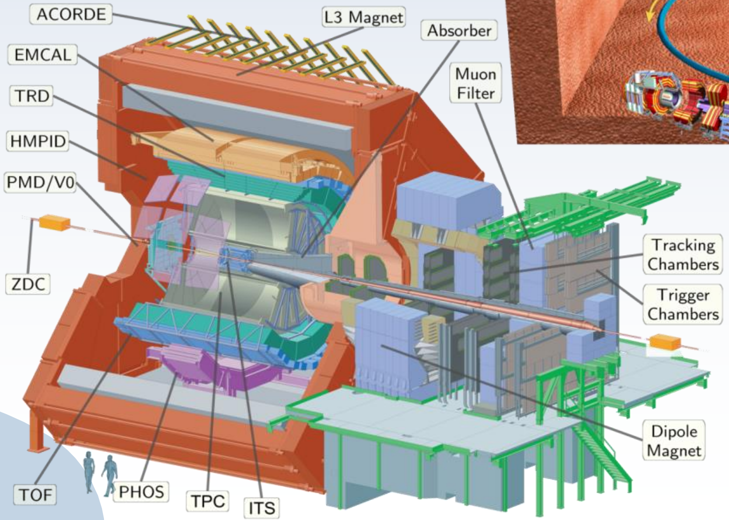
13.10.2016

SPONSORED BY THE
Federal Ministry of Education and Research

ALICE at the LHC

- The Large Hadron Collider (LHC) at CERN is today‘s most powerful particle accelerator colliding protons and lead ions.
- ALICE is one of the four major experiments, designed primarily for heavy ion studies.
- The Time Projection Chamber (TPC) is ALICE’ primary detector for track reconstruction.
- The High Level trigger (HLT) is an online compute farm for real-time data reconstruction for ALICE.





Track reconstruction in ALICE

- **The HLT performs online reconstruction of all events recorded by the ALICE detector in real time.**

- **Tracking is the most time consuming task in online event reconstruction.**

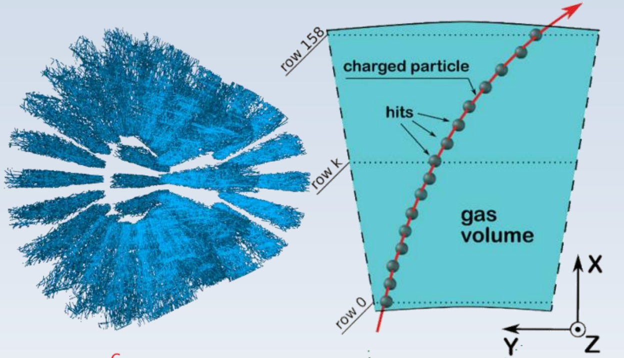
- **We use GPUs as hardware accelerators to speed up tracking and save costs on the online compute farm.**

- **GPU Tracking originally developed for Run 1.**
  - Implementation not necessarily optimal for nowadays GPUs.
  - We want to improve GPU utilization for Run 2/3, and use available GPU capacity for new features.

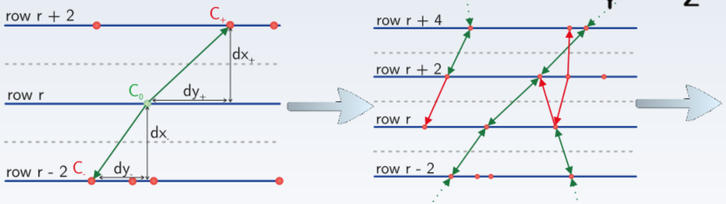
- **Current tracker sufficient for all Run 2 scenarios.**
  - Instead of improving performance for the current GPU generation, we rather aim at new features.
  - Current Run 2 computing farm can also be used as playground for Run 3.

Tracking Algorithm

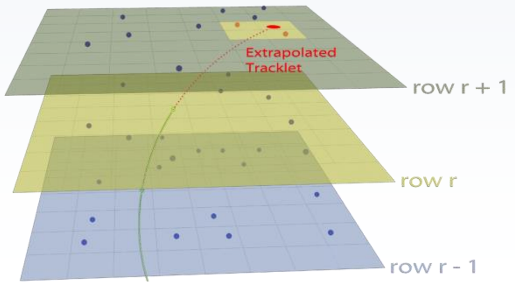
- **TPC Volume is split in 36 sectors.**
  - The tracker processes each sector individually.
  - Increases data locality, reduce network bandwidth, but reduces parallelism.
  - Each sector has 160 read out rows in radial direction.
  - Tracking runs in 2 phases:



- **1. Phase: Sector-Tracking (within a sector)**
  - Heuristic, combinatorial search for track seeds using a **Cellular Automaton**.
    - A) Looks for three hits composing a straight line (**link**).
    - B) Concatenates links.



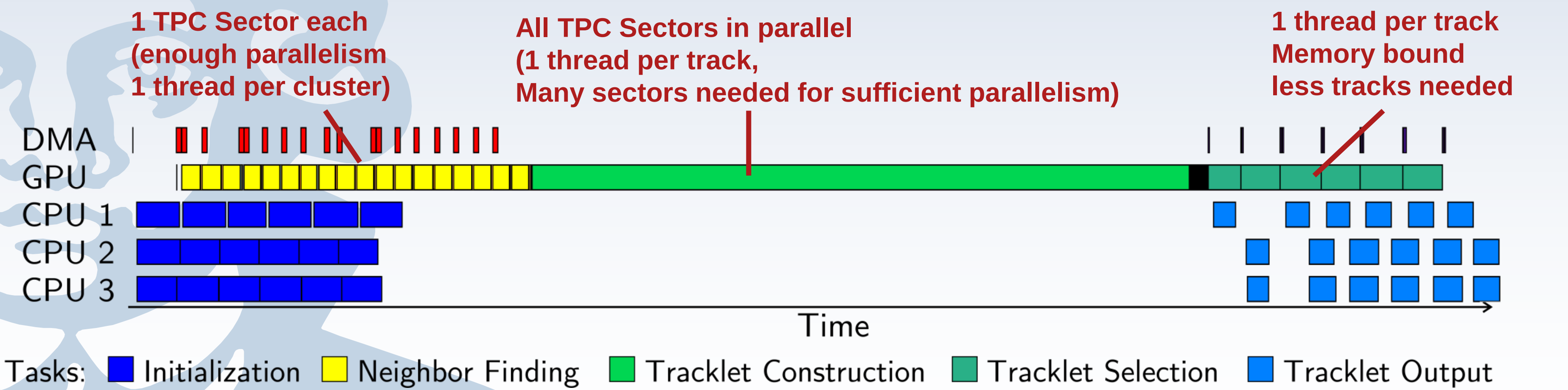
  - Fit of track parameters, extrapolation of track, and search for additional clusters using the **Kalman Filter**.



- **2. Phase: Track-Merger**
  - Combines the track segments found in the individual sectors.

New processing scheme needed

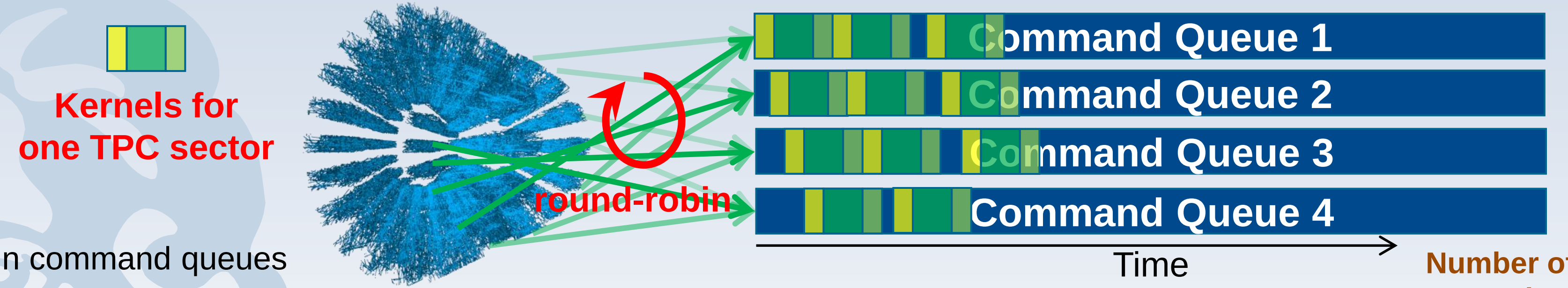
- **The task scheduling for the tracking was originally developed for GTX285 GPUs.**
  - Original scheme limited because old GPUs could not execute 2 different kernels at a time.
- **1st step of tracking is local in one TCP sector, processing of sectors arranged in a pipeline.**
- **Some steps, in particular tracklet construction cannot exploit enough parallelism in one sector.**
  - → Combined processing of multiple sectors.



- **The pipeline ensures that the GPU does not idle,**
  - **BUT, utilization within a single kernel is not necessarily optimal.**

New processing scheme

- **Problem: Too few tracks (and too few clusters in one sector) to load all compute units of modern GPUs.**



- **Idea:**
  - Use n command queues
  - Queues processing for all TPC sector on the queues in a round-robin fashion.
  - Each kernel will always only process one step for one sector, occupy only few GPU cores.
  - GPU scheduler will place multiple kernels concurrently.

Number of queues is a parameter, can match 8 hardware queues on AMD for instance.

- **DMA transfer back to host needs to know number of found tracks.**
  - In order to avoid synchronization, we copy an estimated upper bound of tracks.
    - If too many are copied, doesn't matter, there is plenty of DMA bandwidth and tracks are small.
    - If too few are copied, we can fetch the remaining ones in a second go.
  - → Only one synchronization at the very end of processing is needed.
- **First test shows already 20% faster processing with a simple modification.**

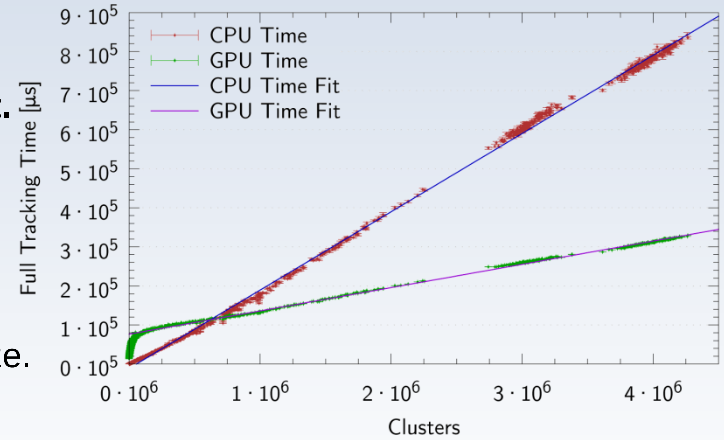
Current setup & maximum rates

- **A simple alternative to increase GPU utilization.**
- **We can run multiple instances of the GPU tracker on multiple events in parallel (without further tuning).**
  - GPU parallelization also over events, on top of tracks / clusters.
  - Tracking time of 1 instance: **145 ms** (Full central PbPb).
  - Tracking time of 2 instances: **220 ms** (**110 ms** / event).
  - Speedup because of better GPU resource usage. Even a full central PbPb event can no longer utilize all ALUs of modern GPUs (this was different some years ago when we started to use GPUs in the HLT).
    - → The speedup is much larger for smaller events.
- **Currently deployed in the HLT for Run 2: Maximum HLT tracking rate is 40.000.000 tracks / second.**

- **Only events with all detectors in**
  - pp (PbPb Reference run, Run 244364, **TPC**, ITS, EMCAL, V0, ZDC): 4.5 kHz (**Limit: CPU**)
  - pp (13 TeV, 25 ns, Run 239401, **TPC**, ITS, EMCAL, C0, ZDC): 2.4 kHz (**Limit: RCU2 bandwidth**)
  - PbPb (Max Luminosity, Run 245683, **TPC**, ITS, EMCAL, V0, ZDC): **950 Hz** (**Limit: RCU2 bandwidth**)
  - **PbPb (Run 245683, local TPC Reco only, no data transport): 2.5 kHz (Limit: GPU)**
- **GPU resources are used at maximum to 45% (assuming max TPC read out).**
  - **Use available GPU resources for other reconstruction tasks.**

Concurrent event processing

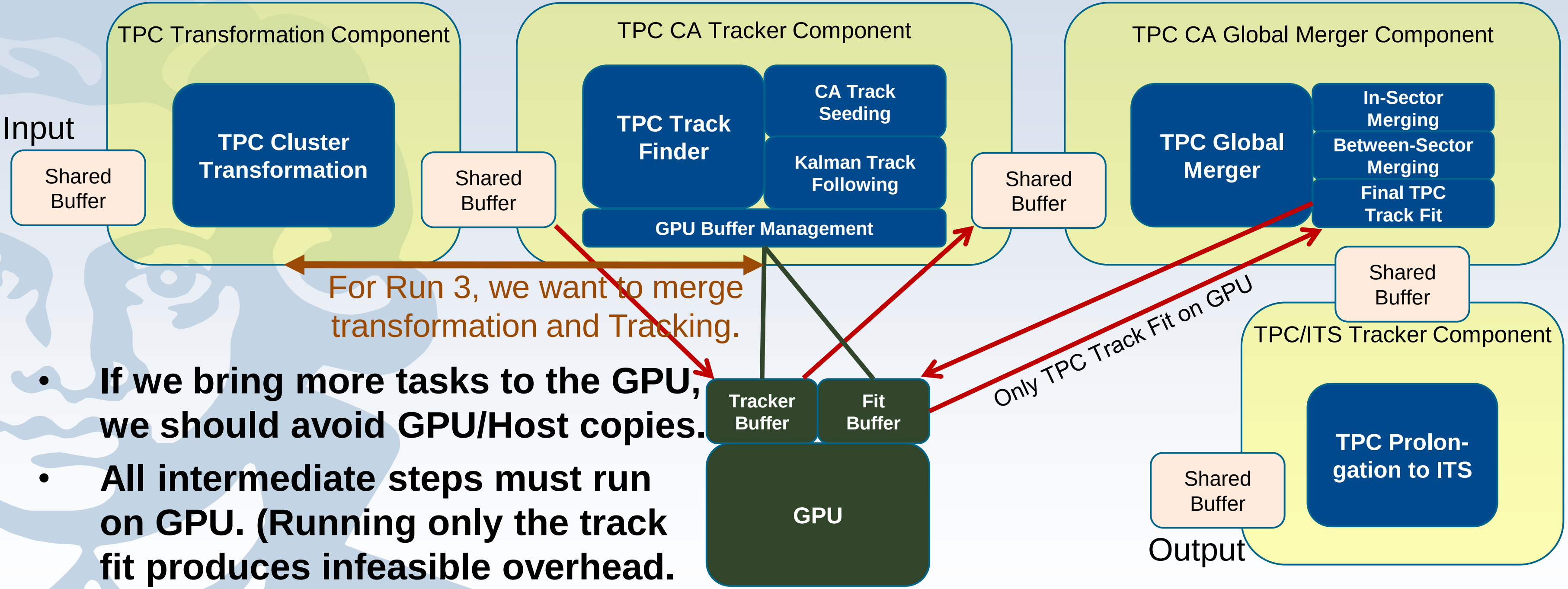
- **We want to try new features needed for O2 already now in the HLT (e.g. online calibration).**
- **GPU Memory usage of TPC tracking is below 1 GB, GPUs in ALICE HLT have 6 GB, in some years 32+ GB.**



- **At very high rates, processing all events individually is inefficient.**
  - E.g. ALICE HLT framework currently limited at 6 kHz.
  - It is better to combine multiple events, and process them jointly.
  - ALICE will inherently do this with time frames in continuous read out.
  - This will also make sure the GPUs are fully utilized.
  - This is possible, because tracking time goes linear with input data size.

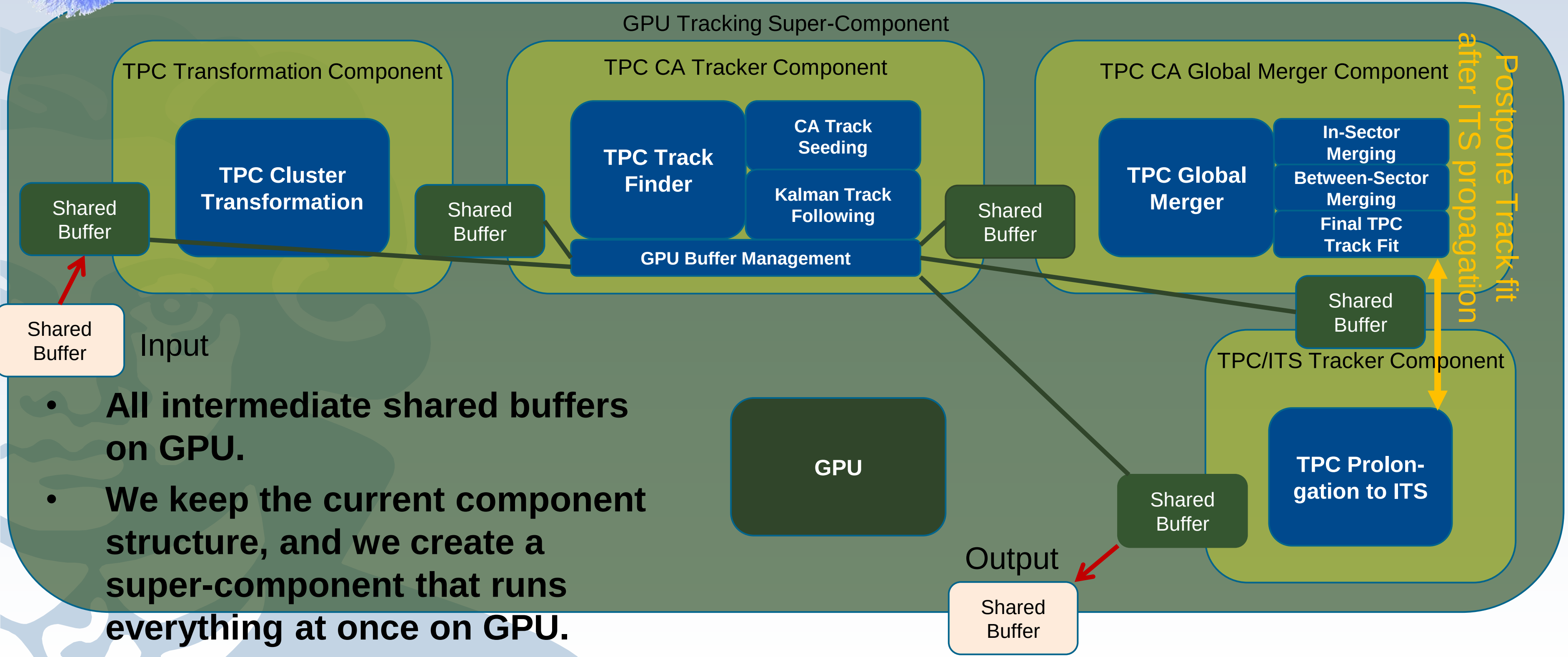
- **Depending on time frame size, we might need to stream the time-frame through the GPU in slices (along z).**
  - We can use GPU scheduling queue as presented in optimized Run2 scheme.
  - From Run 1 / 2 experience, we know that pipelines processing of TPC subvolumes works very well.
  - GPU memory is large enough to hold large slices offering sufficient parallelism.

Current HLT TPC / ITS Tracking



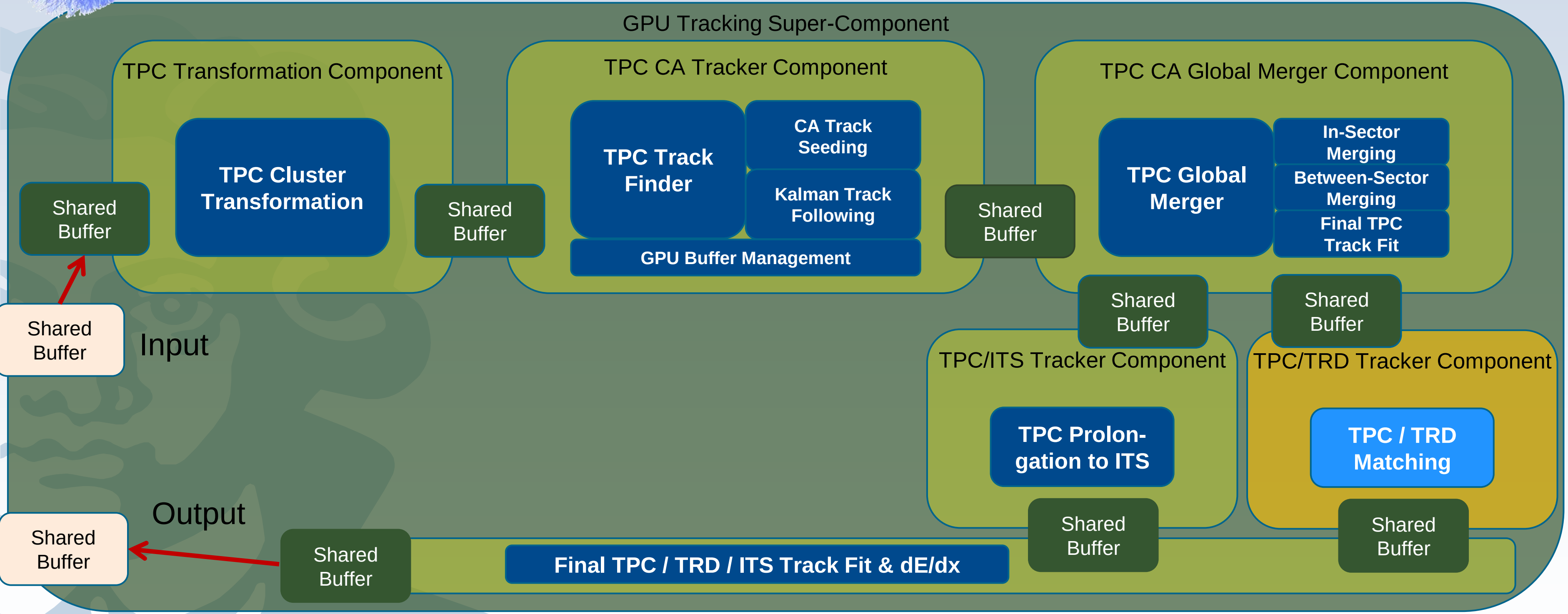
- **If we bring more tasks to the GPU, we should avoid GPU/Host copies.**
- **All intermediate steps must run on GPU. (Running only the track fit produces infeasible overhead.**
- **We have to evaluate which (consecutive) components can use GPU efficiently.**
  - The entire tracking chain seems a good candidate.

Next developments in tracking



- **All intermediate shared buffers on GPU.**
- **We keep the current component structure, and we create a super-component that runs everything at once on GPU.**

Next developments in tracking



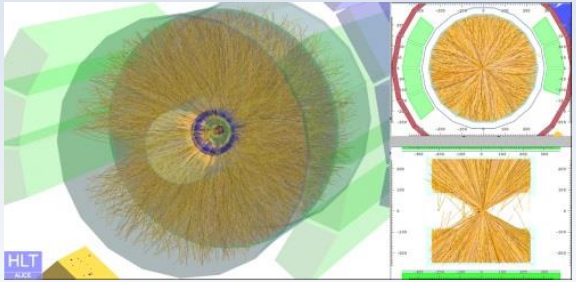
- **TRD prolongation could run in parallel to ITS prolongation, final track fit afterward.**
- **We could add dE/dx to final track fit. New track-based compression needs refit suited for GPUs.**

Summary

- **HLT track reconstruction fast enough to cope with all trigger scenarios in Run 2 and with the maximum TPD DDL link rate.**

- **Tracker has a common source code for CPU / OpenCL / CUDA yielding consistent results.**

- **180 compute nodes with GPUs in the HLT**
  - Since 2012 in 24/7 operation, no problems yet.



- **Cost savings compared to an approach with traditional CPUs:**
  - About 500.000 US dollar during ALICE Run I.
  - Above 1.000.000 US dollar during Run II.
  - Mandatory for future experiments, e.g. CBM (FAIR, GSI) and ALICE upgrade with >1TB/s data rate.
  - Can be used to test new online tracking features for Run III.

- **We are now looking into optimizations for new GPU architectures, but not yet specific to one model.**
  - Plan to bring more components onto the GPU, reduce PCIe transfer, keep component structure.
  - Using GPUs with more memory, we are confident to process timeframes similarly to events today.