Online Calibration in the HLT - first experience

David Rohr, drohr@cern.ch

Frankfurt Institute for Advanced Studies

CERN, 30.3.2016

Requirements in HLT for calibration

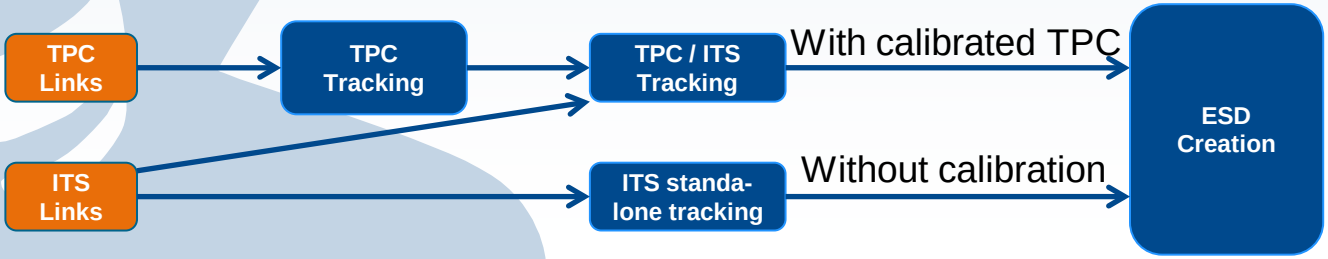
- We want to run online calibration in the HLT.
  - To avoid code duplication, we want to run the offline calibration tasks.
  - We have started with the TPC drift-time calibration as a prototype.
  - On top of the online calibration, we want to apply a feedback loop, to use the calibration results in HLT reco.

- Requirements:
  - We need TPC and ITS tracks *(Thanks to Ruben for support with Standalone ITS Tracking)*.
  - Calibration tasks might run longer than usual HLT processing.
  - The HLT framework does not support a feedback look (it is loop free).
  - We nead means to run the offline calibration task in the HLT.
  - The calibration needs to accumulate plenty of events

    → We cannot apply the calibration result as we process the events but at a later stage.

Approach for fast ITS Tracking

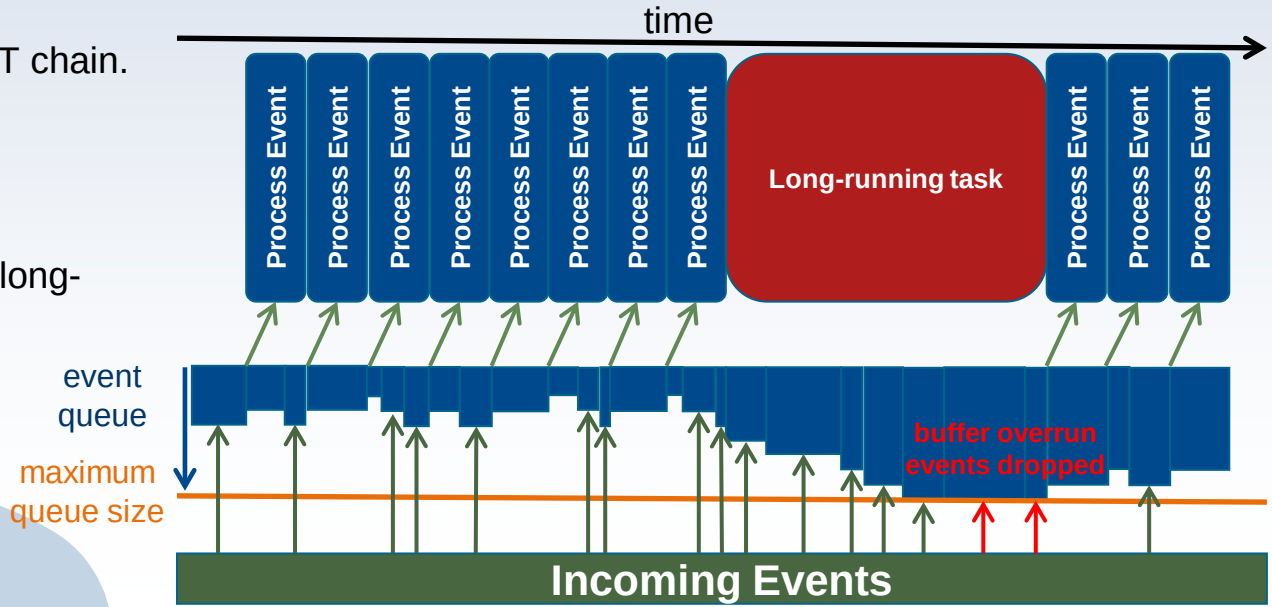
- Calibration requires TPC and ITS tracks, Luminous region needs ITS tracking (TPC is too far away and not precise enough):
  - HLT creates ITS tracks through TPC prolongation.
    (Full ITS standalone tracking is slow because of the complicated combinatoric.)
  - This might not work properly, if the TPC is not calibrated → Chicken and egg problem.
    → We run a new fast standalone ITS tracker based on SPD tracklets (by Ruben Shahoyan).
    - **This approach reduces the combinatoric a lot, hence it is fast.**
    - **It might not find all tracks.**
    - **Still, it is sufficient for the calibration (and some other tasks).**
    - HLT still uses TPC-ITS prolongation for full ITS tracking.



Challenges for Online Calibration

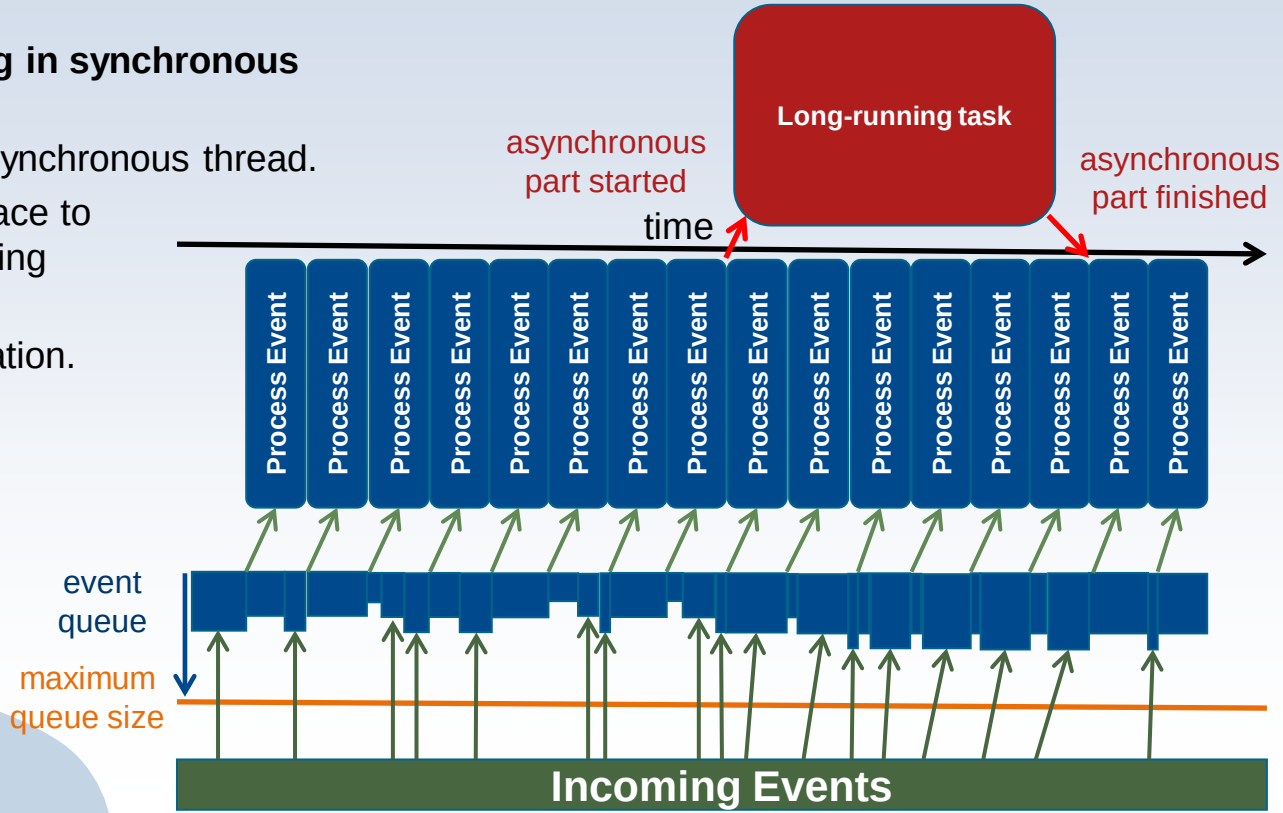
- **HLT processing based on events.**
- **Components process one event after another.**
  - → Long running infrequent tasks could make the event buffer overrun, even if the average processing time is short.
  - → This will stall the entire HLT chain.
  - → Events will be lost.

- **Examples in calibration:**
  - Accumulating events first, long-running fit later.
  - Fast event selection, long event processing.



Asynchronous Side Tasks

- **Solution: Split processing in synchronous and asynchronous part.**
  - Framework spawns an asynchronous thread.
  - It provides a simple interface to the component for offloading asynchronous tasks.
  - It handles the synchronization.



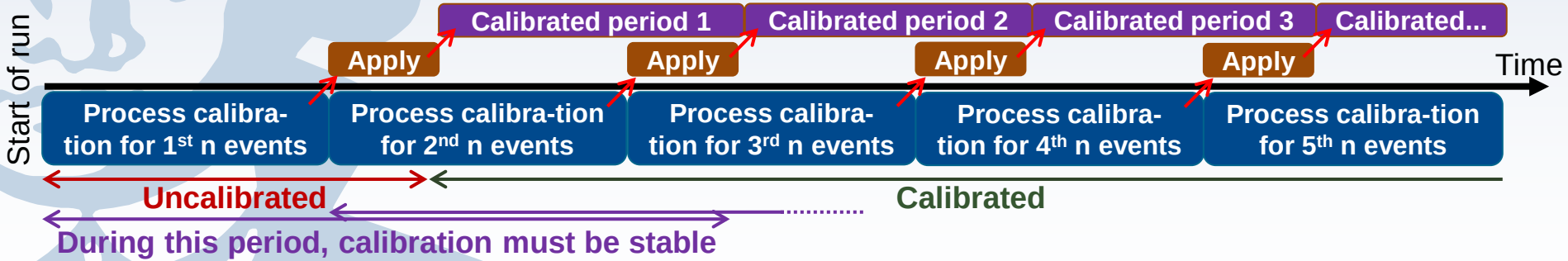
Asynchronous Threads / Processes

- Additional benefit of asynchronous components: **Resiliency to segmentation faults:**
  - We do not have the calibration code under our control.
  - A segfault there must not stop data-taking.
  - Event-synchronous processing does not work for long-running calibration tasks – stalls chain.
  - We now support asynchronous components.
    - Process as many events as possible on best effort basis.
    - Skip remaining events.
    - Processing either in asynchronous thread or asynchronous process.
      - Process-variant communicates via shared memory, still slightly slower.
      - Segmentation fault or memory leak in asynchronous process does not stop data taking.

        → **Resiliancy for fatal errors in HLT.**
      - A failing calibration component stops the calibration, but normal HLT operation continues.
      - Possibility to restart failing process..
        - (some events will be lost for calibration, processing continues transparently).

General approach for eventual calibration

- Calibration is time-dependent – drift time is calibrated in time intervals.
- Calibration needs to process in the order of n = 5000 events (Pb-Pb) per time interval.
  - The calibration task is slow, calibrating 5000 events can take minutes.
  - We should reconstruct after the calibration, but we cannot cache all events that long.
    - → There is no way we can use the calibration for the reconstruction of the first events we record.
  - But, the calibration is stable over a certain time period (here we assume 15 minutes for the drift time):
    - We can run the calibration for some time, and use the result for the next time period, as long as it is stable.
    - → **Reconstruction of the first events is not calibrated, but for all following events it is.**



- **The important question is: Can we process the calibration for enough events and apply the result in time?**

→ **Solution to E**

AnalysisTasks in HLT

- **We want to run the offline tasks in the HLT.**
  - This avoids code duplication.
  - Puts some constraints on the task:
    - Performance: The task should not be too slow.
    - Environment: Not necessarily all AliRoot environment is initialized / available, i.e. HCDB instead of OCDB.
    - ESDs are a problem:

- **ESDs are ROOT objects – All HLT components run in a different process.**
  - ESDs (/ all ROOT objects) must be serialized / deserialized when passed between processes.
  - In HLT, we provide Flat-ESD: Essentially the same information in a flat data structure.

- **Unified access to ESDs and Flat-ESDs via AliVEvent. (Not all functionality for Flat-ESDs implemented yet.)**

- **HLT runs a wrapper for the AliAnalysisTask (AliHLTAnalysisManager) and provides Flat-ESDs as input.**

→ All tasks that run with AliVEvent are in principle supported by HLT.
  - Missing functionality for the Flat-ESD must be implemented when needed.
  - Performance constraints must be met.

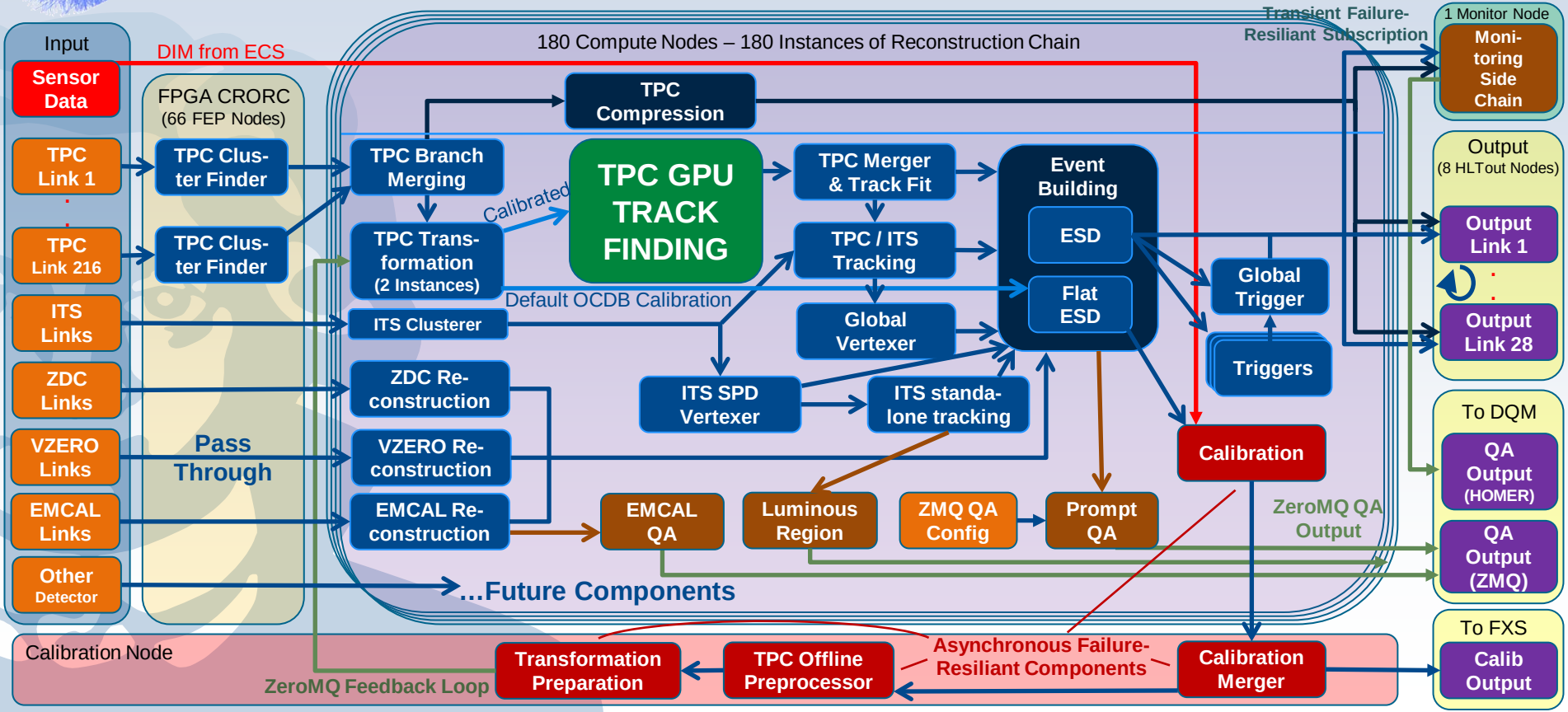
Approach for distributed calibration

- HLT runs 180 compute nodes, possibly multiple instances of a process per node.
- We need to run many AnalysisTasks in parallel for calibration (as in offline).
- Calibration objects must be merged afterwards:

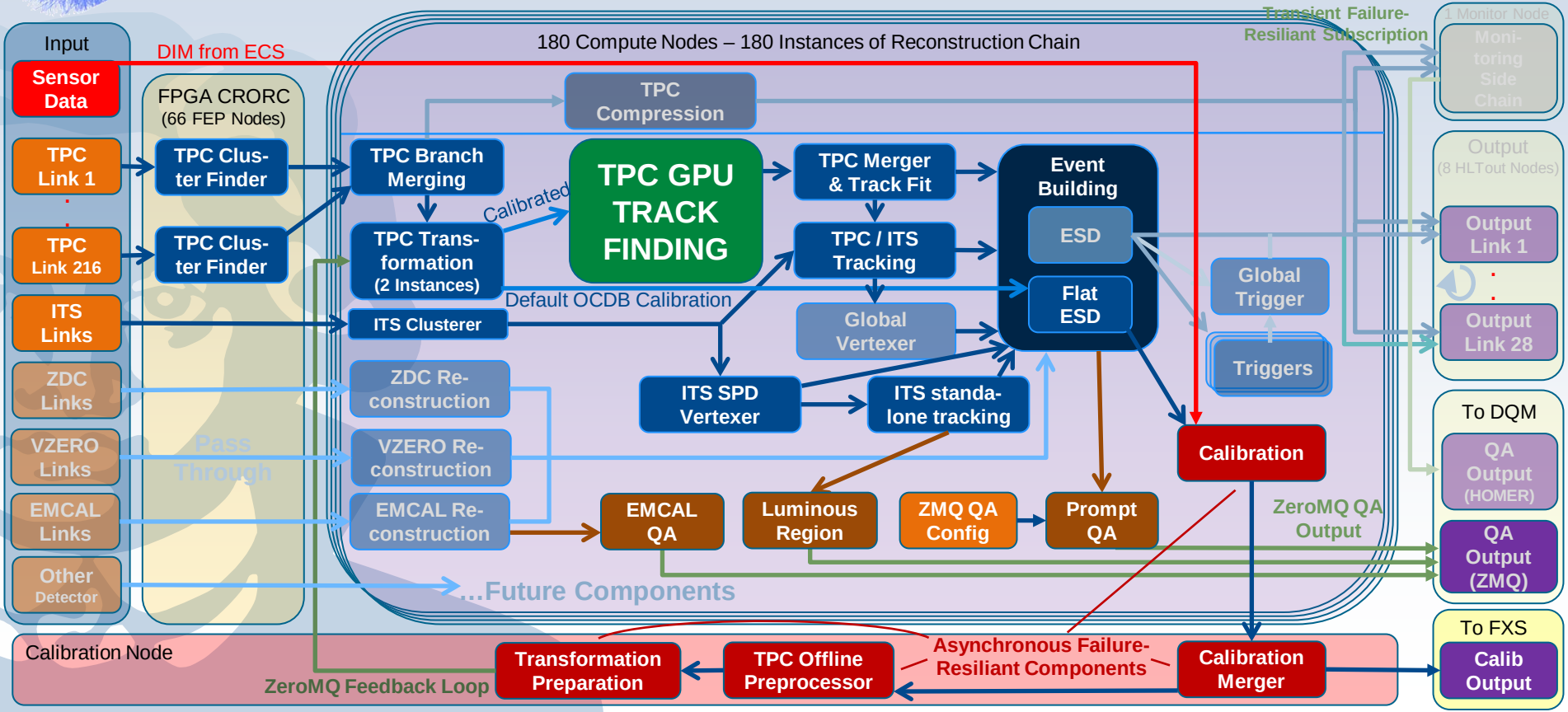
- Current approach:
  - (At least) one AnalysisTask per HLT compute node.
    - Ships output to a central calibration node once in a while (configurable, possibly data-driven).
  - The central component merges all the AnalysisContainer.
  - Then it runs the OfflinePreprocessor to create the OCDB Object.
    - This object is then shipped to FXS.
    - This object is passed back into the chain to apply the calibration to the cluster transformation (feedback-loop).

- Feedback-loop data transport is implemented via ZeroMQ.

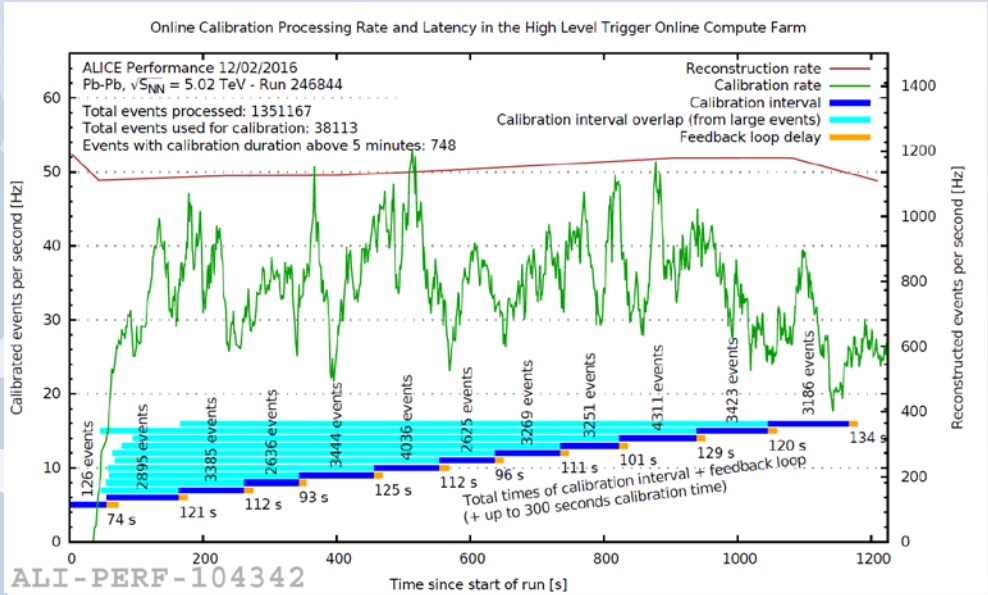
Overview of current HLT components



Overview of HLT calibration components.

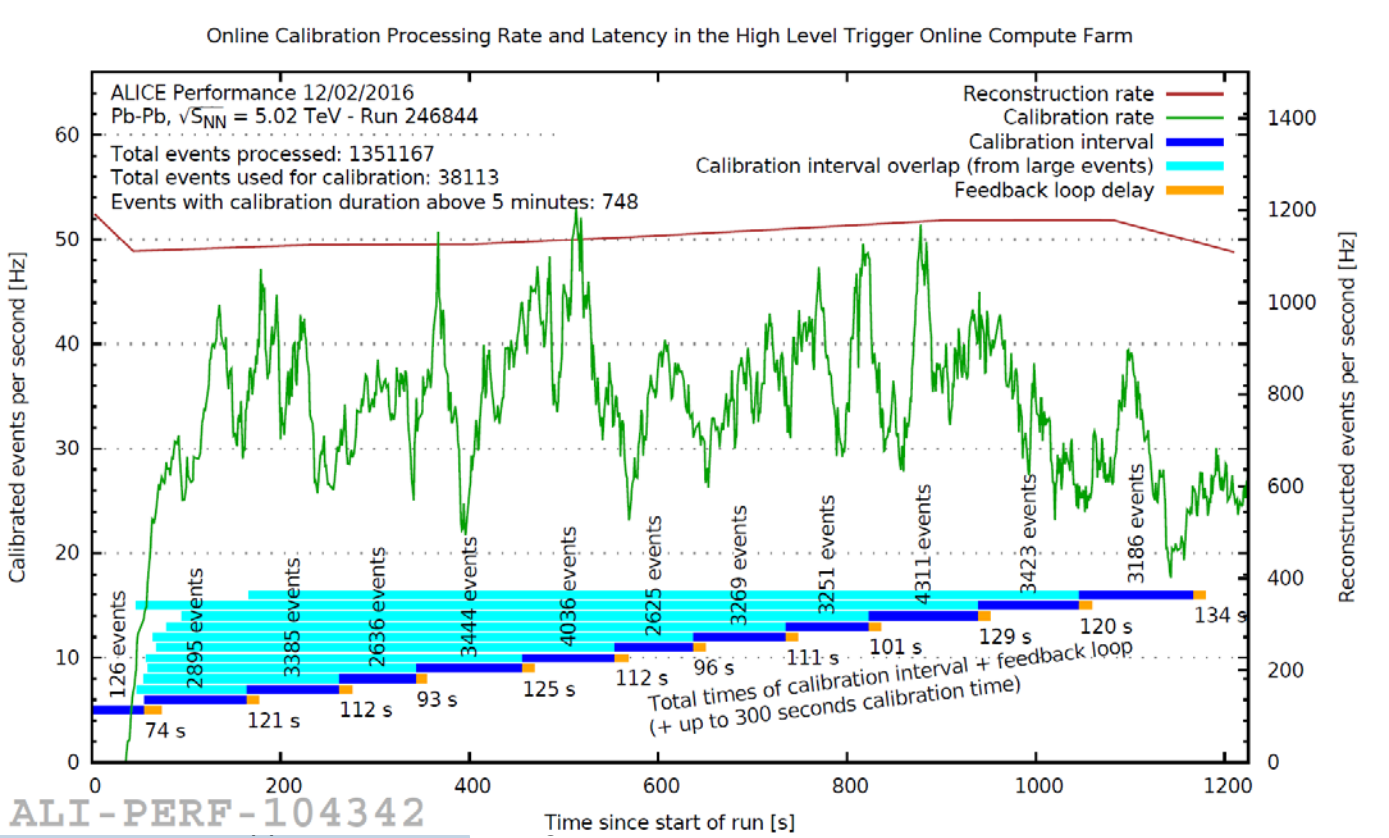


Online Calibration Results



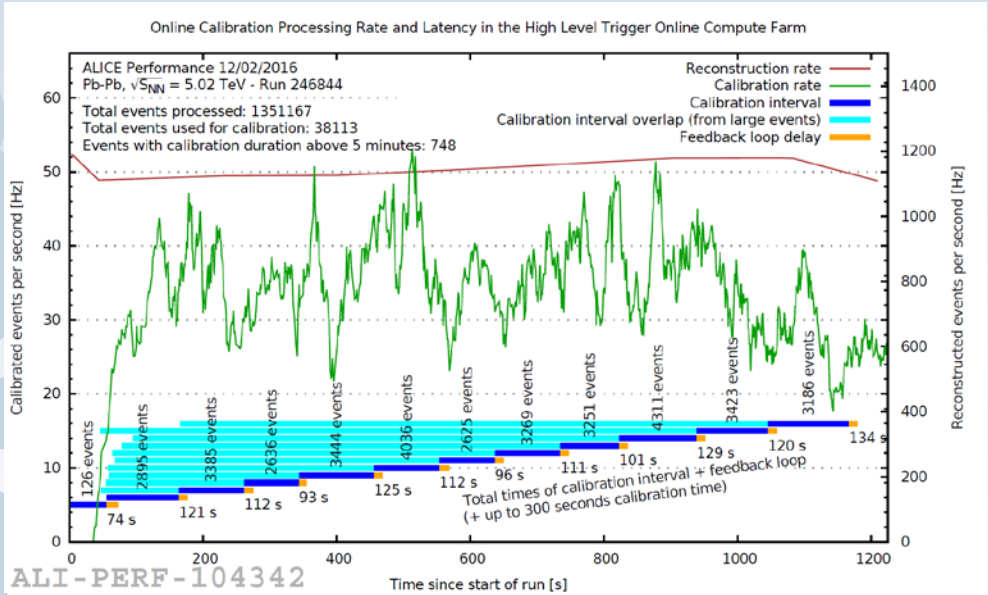
- The calibration rate shows the total number of events that finished calibration per second aggregated over all HLT nodes. The rate is 0 at the beginning because at t = 0 the calibration for the first events is already running but no events are finished yet.
- The dark blue bar shows the duration of one calibration interval and the number above states the number of events that finished calibration during that calibration interval. The yellow bar is the additional delay after the calibration interval ends up until the new calibration is applied in HLT tracking.

Online Calibration Results



- The calibr … l HLT nodes. The rate is … ents are finished yet.
- The dark … events that finished ca … nterval ends up until the n …

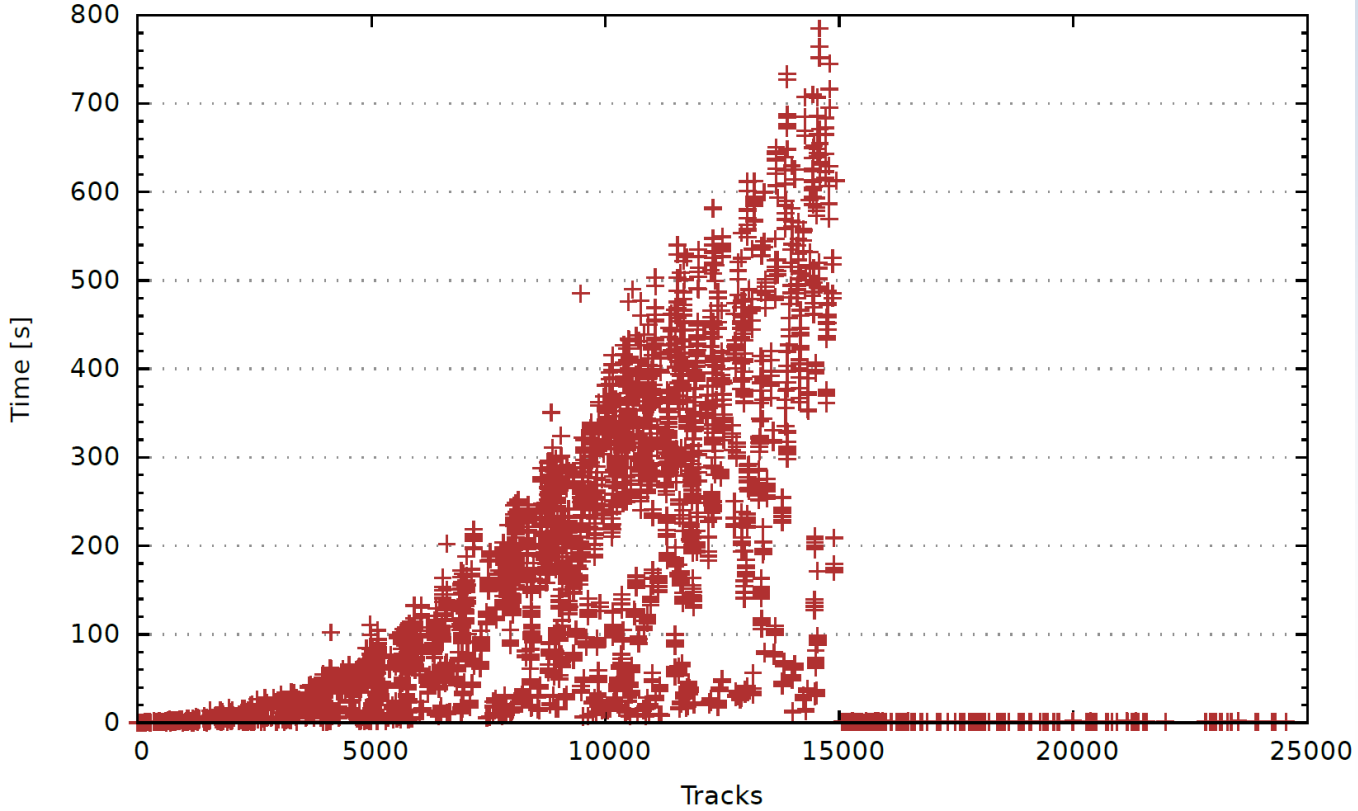
Online Calibration Results



- Due to the long-running calibration task, the calibration of an event that finishes during one interval can start before the beginning of that interval. The light blue bar shows when the first event was recorded that ends in the calibration interval. The processing of an event can take up to 15 minutes, but only around 2% of the events take more than 5 minutes.
- Overall, the plot shows that the HLT can run the calibration tasks on events at an average of 31 Hz and it can apply a calibration that contains 98% of the processed events to the reconstruction with a delay of less than 7 minutes. (up to 5 minutes calibration time + ca. 2 minutes calibration interval and feedback loop delay)

Processing Time

- Large events (many tracks) need much more compute time.
- By selecting only smaller events, we can process more tracks per time.



Online Calibration Results

- **We did a first successful test of online calibration in December 2015.**
- **The test was done during Pb-Pb data-taking under real conditions.**
  - 5 CPU cores per node (on 120 nodes) used for calibration.
  - In average, 31 events per second processed in calibration component.
    - New optimized configuration achieves 81 events per second.

- **There are still some software issues / bugs / inefficiencies to be solved in order to make this run in production, but we have seen that in principle we can run calibration tasks in a high load situation during Pb-Pb data taking online and apply the calibration results fast enough.**
  - Calibration objects passed to cluster transformation in calibration interval via feedback loop:
    - Delay: ~140 seconds (20 seconds for feedback loop and preparation of transformation map).
    - ~ 3000 events per interval
    - 98% of events finish calibration within 5 minutes
      - → Delay until application of transformation map: < 7.3 minutes.
  - With precalculated transformation map, the calibration can be switched instantly.

Online calibration problems

- Performance is a bit critical:
  - We were able to process sufficient events / time, but at the cost of high CPU load.
  - Selecting on number of tracks will help here → **OK (improvement possible).**
  - Should be possible to speedup the calibration task *(Investigation ongoing by Mikolaj and Sergey).*
- Memory leaks:
  - There was a large memory leak in the merging of calibration objects → **fixed**.
    - A similar leak was also preventing Alex from working on dE/dx calibration → Now also fixed.
    - This could be one of the next tests.
- Some bugs that could lead to crashes in the Flat-ESD → fixed.
- Problems in calibration output:
  - Calibration QA Histograms look weird for some runs.
  - This depends on the run: Some look OK, some look bad.
  - We can reproduce this with data-replay, so it should be possible to fix this offline.

Progress on fixing calibration output

- One idea was that it could be OCDB related:
  - HLT uses HCDB (slightly modified copy of more or less recent OCDB).
  - → Test with OCDB Snapshot from cpass0 of the correct run:
    - Feature added to HLT to use OCDB snapshots.
    - Unfortunately, there is absolutely no difference between HCDB and OCDB snapshop in terms of calibration.
- Next step: reproduce this in HLT-emulation in AliRoot (recraw-local.C macro).
  - Not yet possible due to a problem that prevents running on HLT-Mode-C data.
  - This problem is currently being fixed.
  - Investigation ongoing.

Summary

- Flat-ESD implemented and deployed in HLT – Running stable and fast.
- Feedback loop implemented and tested.
- Asynchronous AnalalysisTasks can now run in HLT (using AliVEvent → FlatESD).
- Online calibration tested during data taking in physics run under real conditions:
  - Result looks rather good.
  - Some performance (speed) deficiency, this is under controlled.
  - All memory leaks / crashes fixed.
  - We still have to understand why the calibration output for some runs is wrong.