FIAS Frankfurt Institute for Advanced Studies

Improvements of the ALICE HLT data transport framework for LHC Run 2

David Rohr *for the ALICE Collaboration*

Frankfurt Institute for Advanced Studies

CHEP 2016, San Francisco

10.10.2016

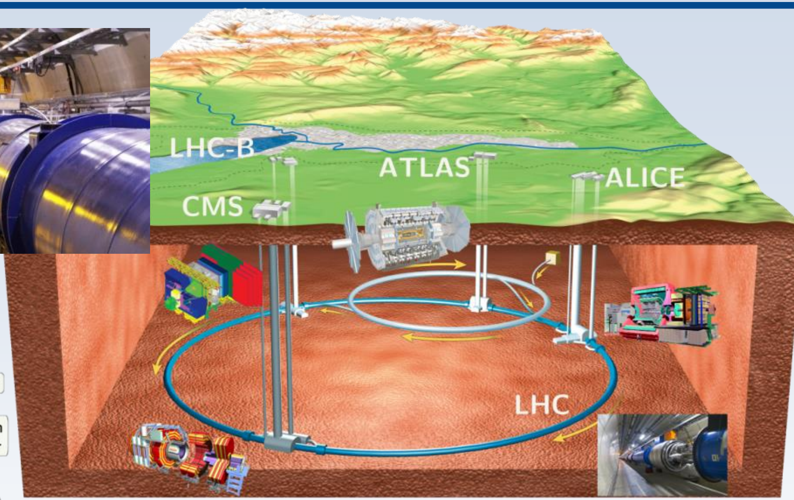
SPONSORED BY THE

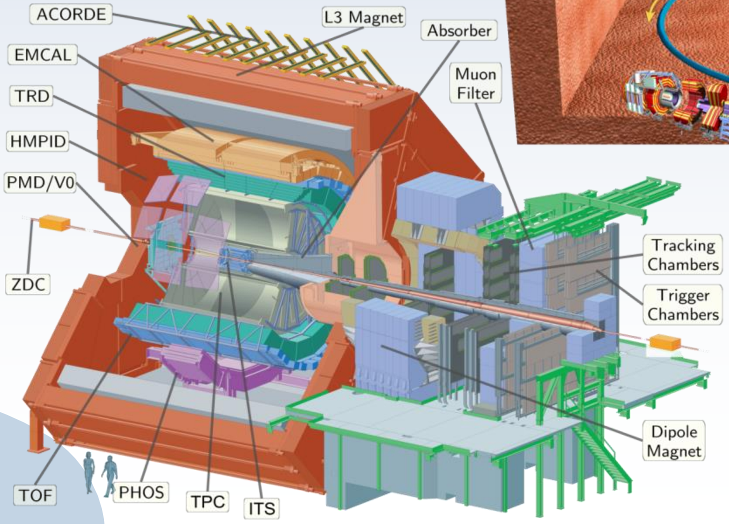
Federal Ministry of Education and Research

ALICE

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.



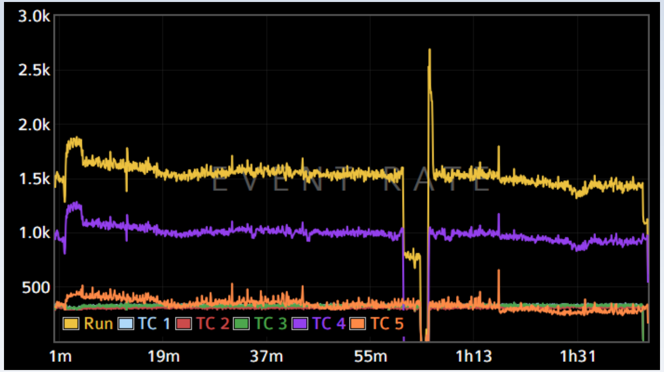


Challenges for framework

- **High Date Rate**
  - The HLT processes an incoming date rate of up to 50 GB/s. This data must be distributed in the cluster and processed in real-time with low latency.
- **High Event Rate**
  - Event rate does not depend on data rate, although it is related.
  - Fast detectors can send a very high event rate at low data rates.
  - The challenge is not the data size, but the merging of event fragments received on different links at high rate.
- **CPU load**
  - The data transport should use as little CPU resources as possible to leave the capacity for processing.
- **Startup and configuration**
  - The HLT needs to configure all the processes at start of run for the current run / trigger / detector configuration.
  - Startup should not take longer than for the detectors in order not to waste beam time.
- **New framework features for new task (online QA, online calibration).**
- **Differences to ALICE run 1:**
  - Higher event / data rate, e.g. faster TPC read out with new RCU2 readout card (twice the bandwidth).
  - Aim to run more processing and QA components for more detectors than before.

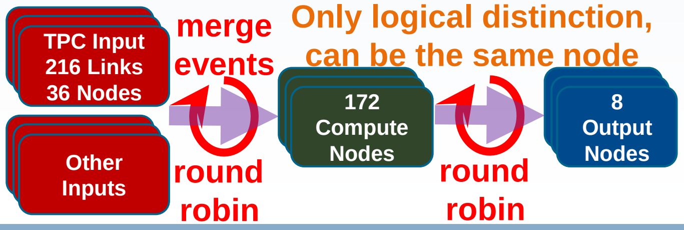
Estimate worst case TPC scenario

- **For compute performance stress test, we use data replay of Pb-Pb events from Run 245683.**
  - (Run was above design luminosity for a short time → biggest events)



- **In this way, we determine the maximum data / event rate.**
- **Worst case analysis: the TPC with RCU2 runs at 3.125 GHz**

→ Maximum possible data rate:
  - ~ 280 MB/s per link with max occupancy, or 50 GB/s in total.
    - Corresponds to **1.377 GB/s** per input node.
    - Translates to maximum output of **1.53 GB/s** per output node.
    - Infiniband IPoIB transfer above **2.4 GB/s**.
  - The total output data rate (compressed TPC clusters, ESD) of the entire HLT in this scenario is 10.7 GB/s.
    - Data output to DAQ has been tested up to **12 GB/s**.
- **Overall, from processing, network, and DDL perspectives, HLT can handle the maximum rate.**
- **Other detectors are a different story:**
  - With TPC readout of 500 Hz, other detectors might have few kHz.
  - Then, our bottleneck is the event merging of the many (small) events.
  - The problem is not the big TPC events.



Optimization steps

- **Event merger:**
  - Use hash-based lists for fast indexing
  - No single bottleneck exists in the merger:
    - Much time used for spinlocks and gettimeofday (for nanosleep), many context switches.
      - Often no accurate time needed, some delays are accepted to avoid context switches.
    - **One bottleneck were system calls to read / write for the named pipes.**
      - → **Named pipes are now replaced by shared memory based communication.**
    - We reduce the rate of PubSub messages to the merger, or merge messages (e.g. merge messages).

→ **Merger (on its own) can now operate with up to 6 kHz with 12 Inputs (maximum due to 12 DDLs per FEP).**

– **(12 inputs is the maximum we can have from our Read Out Receiver Card (C-RORC).)**

- **Highest expected rate for 2016 Data Taking is 2 kHz central barrel + ~1-2 kHz from fast detectors.**

Possible rates

- **Maximum event rates measured in data replay.**

- **Selection test scenarios (all detectors in):**
  - Single Publisher (ZDC) without Event Merger on FEP: > 10 kHz.
  - 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, **Without TPC**, Only ITS, EMCAL, V0, ZDC): **6 kHz** (**Limit: Event merger**)
  - PbPb (Run 245683, **local TPC Reco only**, no data transport): 2.5 kHz (**Limit: CPU / GPU**)

  - Before, the limit was **500 Hz instead of 950 Hz** and **3 kHz instead of 6 kHz**.

- **Real scenario with real event mix (not all detectors always in):**
  - **PbPb (Run 245683)** **950 Hz TPC, 3.75 kHz Total**
  - **pp (Run 239401)** **2.4 kHz TPC, 6 kHz Total**

- **Since beginning of 2016, there has not been a single run that failed because HLT could not keep up the rate.**

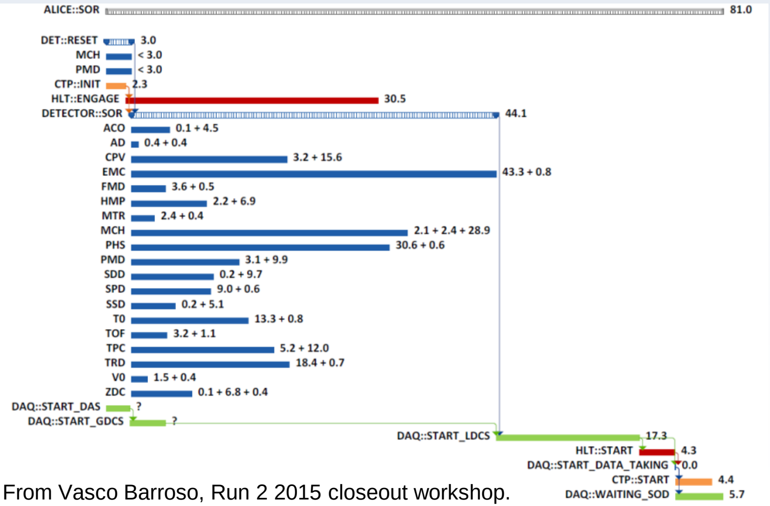
Configuration improvements

- **Run coordination asked us to improve the configuration to reduce ALICE startup time**
  - Main driver: MakeConfig python script, takes up to 210 seconds.
    - Read config input: 30 down to 1.5 seconds.
    - Create process list: 160 down to 13 seconds.
    - Write output: 20 down to 2 seconds (through **python-multiprocessing**).

→ Total now: **16.7 seconds**

- **Besides the MakeConfig script, other minor tasks have been improved, or hidden in the shadow of MakeConfig.**
- **Total configure time improvement: 215 seconds down to 18 seconds.**

Analysis of startup times before improvements.
HLT was in the shadow of detectors,
which improved in the meantime.



From Vasco Barroso, Run 2 2015 closeout workshop.

Engage and configure time

- **Another task was to reduce the engage time:**
  - There was much less margin than for configure. Via software improvements, we could reduce the engage time from 32s to 22s.

- **We can move some steps from the engage step to the configure step.**
  - This has a negative effect on the possible parallelization during startup.
    - → **Engage time goes down.**
    - → **Configure time grows.**
    - → Total time goes up slightly (+1 second for creation and distribution of GRP object.)
  - Engage **22.5** secs to **16.5** secs.
  - Configure **15.5** secs to **22.5** secs.
    - *(Different configure time than before due to slightly different setup.)*

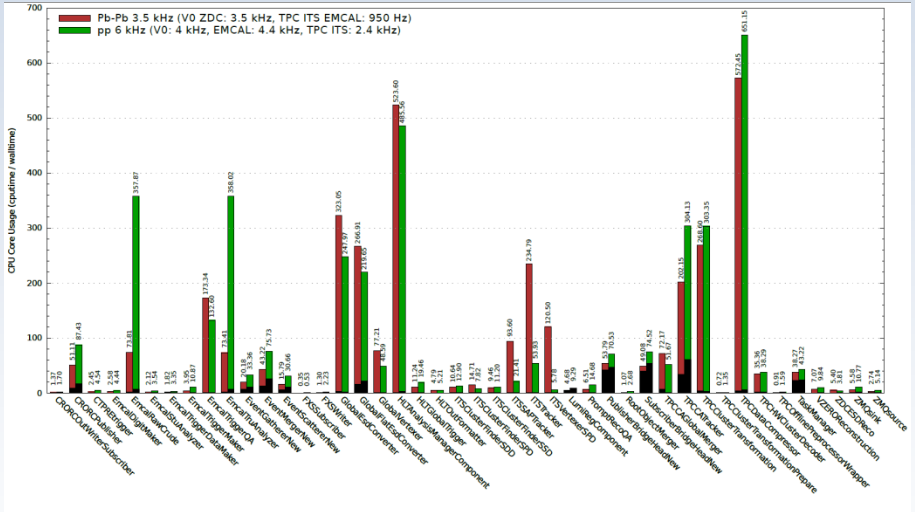
- **Both for configuration and for engage the HLT is now in the shadow of either DAQ or of multiple detectors. HLT never delays the start of a run.**

- **Also: all race conditions and problems with ECS interface fixed ensuring constant startup time – no startup failures (except for obvious regions – wrong B-field) any more this year.**

Total CPU load reduction:

- **Benchmark at high rate processing for maximum framework load**

- ***Rate:*** ***3 kHz*** ***→ 6 kHz***
- **Event Merger:** **240%** **→ 200%**
- **TaskManager:** **100%** **→ 30%**
- **RORCPublisher:** **12 * 75%** **→ 12 * 30%**
- **DataRelay:** **80%** **→ 0%**
- **EventScatterer:** **80%** **→ 60%**
- **Sum:** **1200%** **→ 650%**

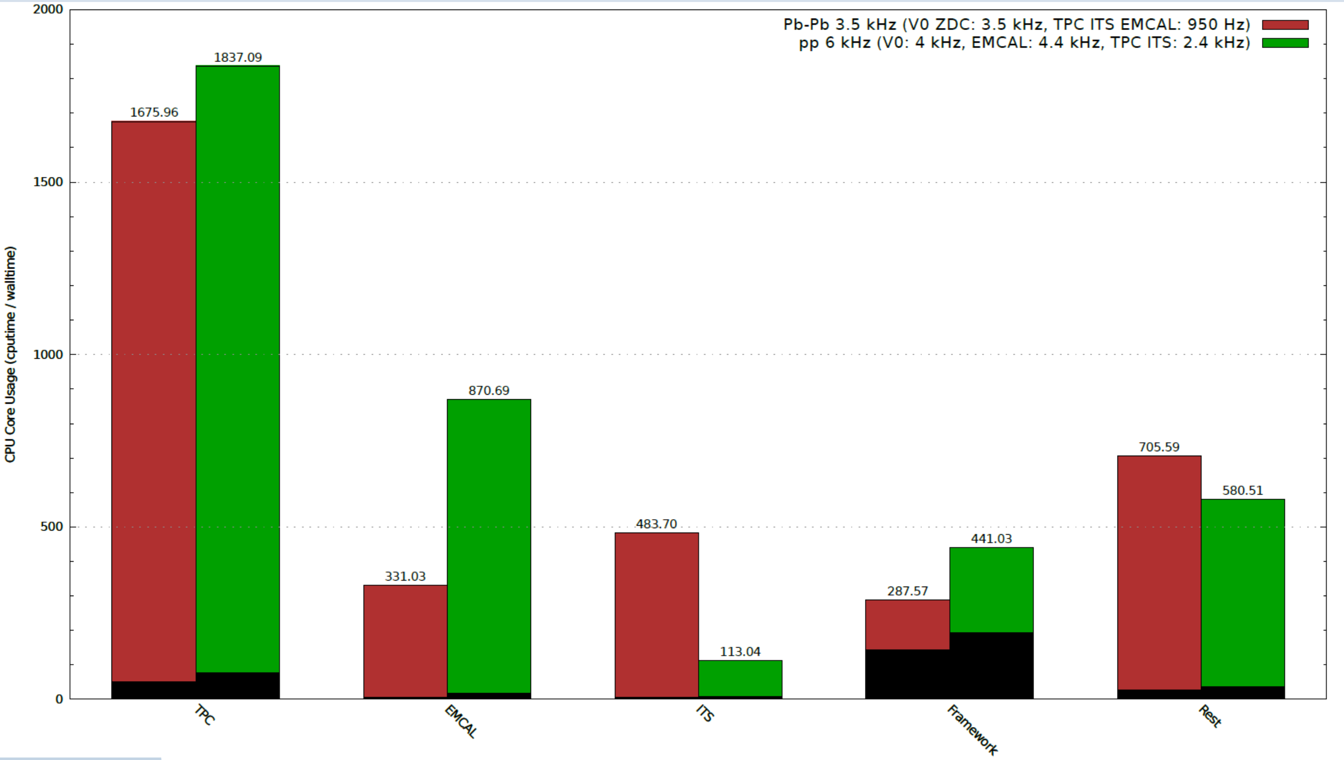


- **This frees up plenty of CPU resources on the FEP.**

- **Some individual components with very high compute load.**
  - Mostly the TPC components.

Processing Time Overview

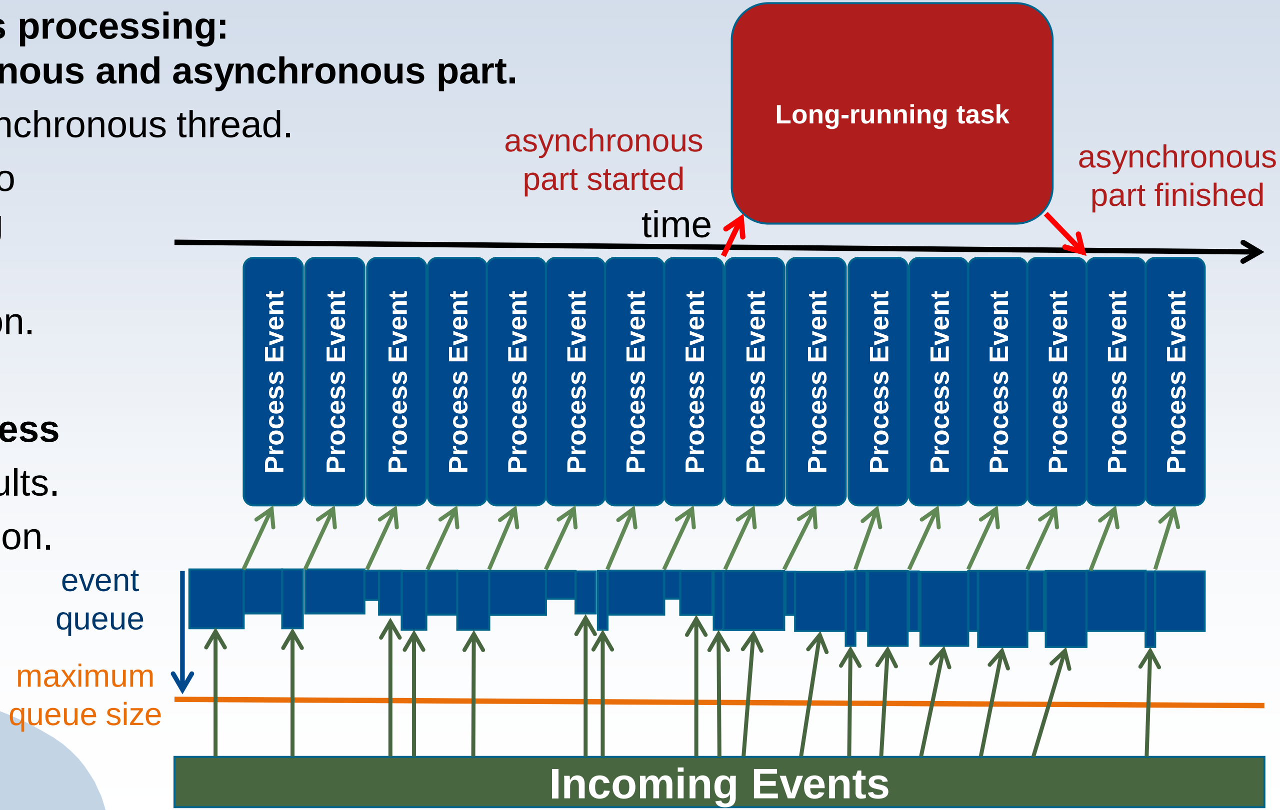
- Black bars show system load in kernel space.
- Framework has significant system load for data transport.
- TPC has some system load for DMA transfer to GPU.
- Overall, framework load is not dominant.



Asynchronous Side Tasks

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

- **Task runs in a different process**
  - Resilient to segmentation faults.
  - Cannot affect normal operation.



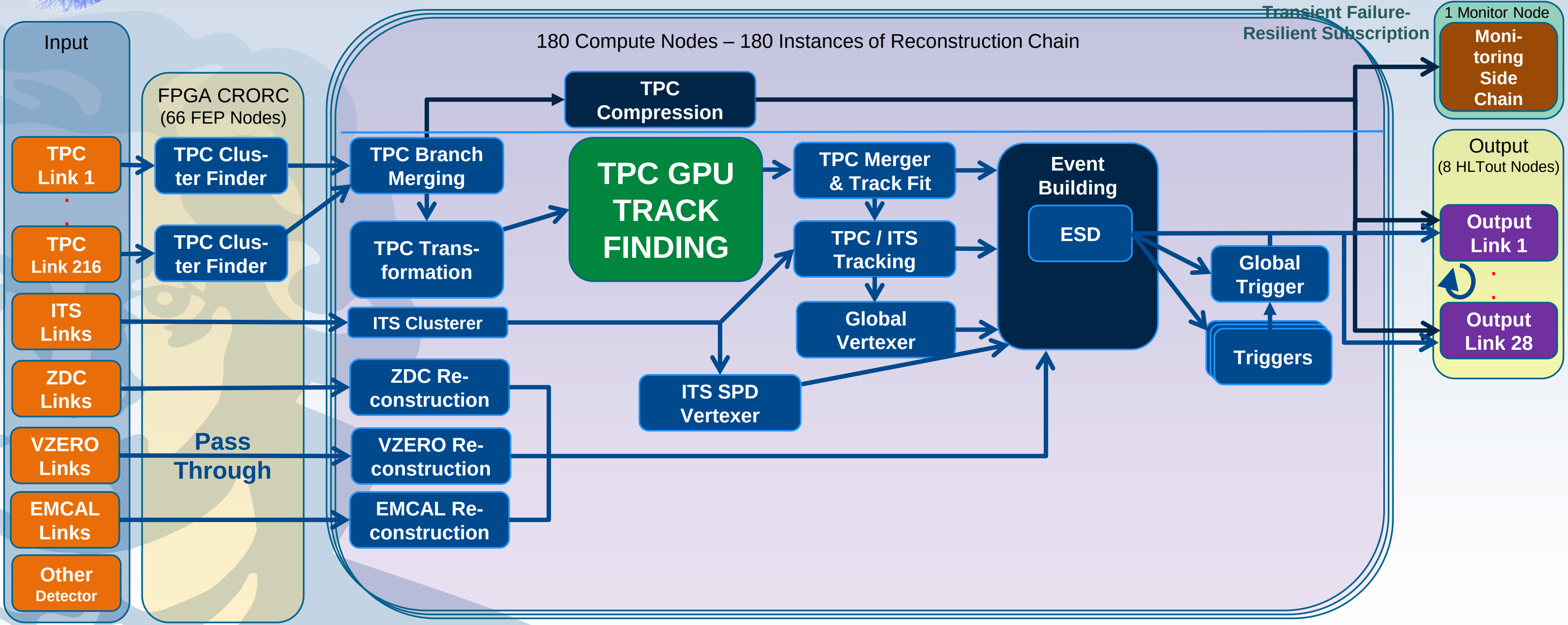
New Zero-MQ based message transport

- **Some features were not feasible with the original HLT data transport:**
  - HLT framework is a loop-free directed graph → no feedback loop.

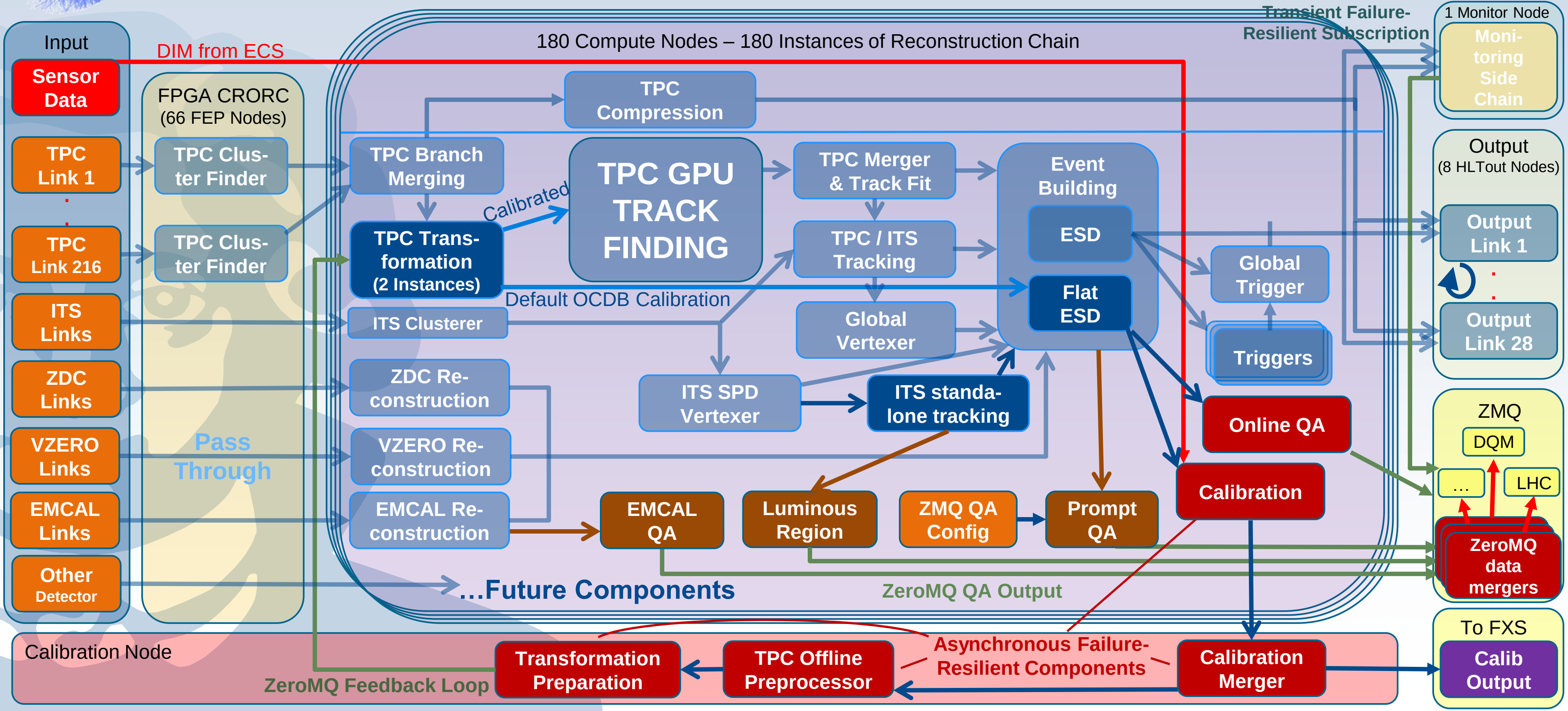
- **New ZeroMQ transport as additional transport mechanism**
  - Similar message based approach as in the HLT itself.
  - Works also as prototype implementation for O2.
  - Used in the HLT for online calibration feedback loop.
  - All new online QA components and the event display use this new approach.

- **Transparent inclusion in HLT configuration:**
  - ZMQ sources / sinks take messages from HLT framework and forward via ZeroMQ.

Overview of Run 1 HLT components



Overview of current HLT components



Summary

- **HLT framework throughput improved:**
  - Can cope with any data and event rate expected for run 2.
  - Can process TPC data at maximum link speed of 50 GB/s.
  - Event mergers with highest load of 12 links operate at up to 6 kHz.
  - Framework load reduced significantly, leaving more resources for reconstruction tasks.
- **HLT Startup time improved → never delays the start of run.**
- **Main improvement step:**
  - Improve inter-process communication via shared memory.
  - Redesign processing graph for better load-balancing.
  - Speed up python configuration scripts, use multi-processing in python.
- **New feature added:**
  - Feedback loop and asynchronous processes enable online calibration.
  - ZeroMQ transport added for calibration and for online QA.
  - Asynchronous processes protected against fatal errors like segmentation violations.