LHCb HLT2: Real-time alignment, calibration, and software quality-assurance

Miroslav Saur
(TU Dortmund)

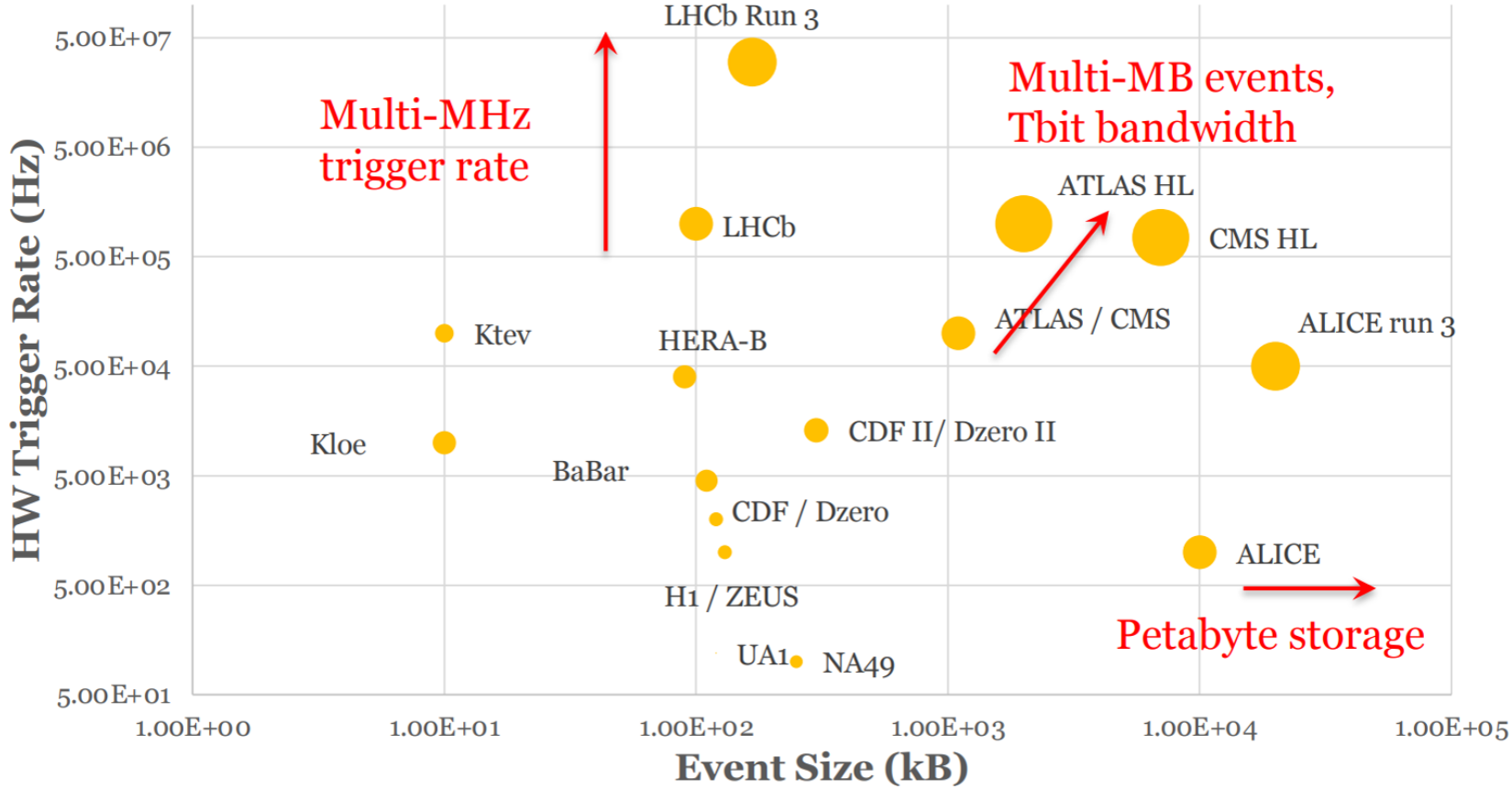
on behalf of the LHCb-RTA project

ICHEP 2022

2022/07/09

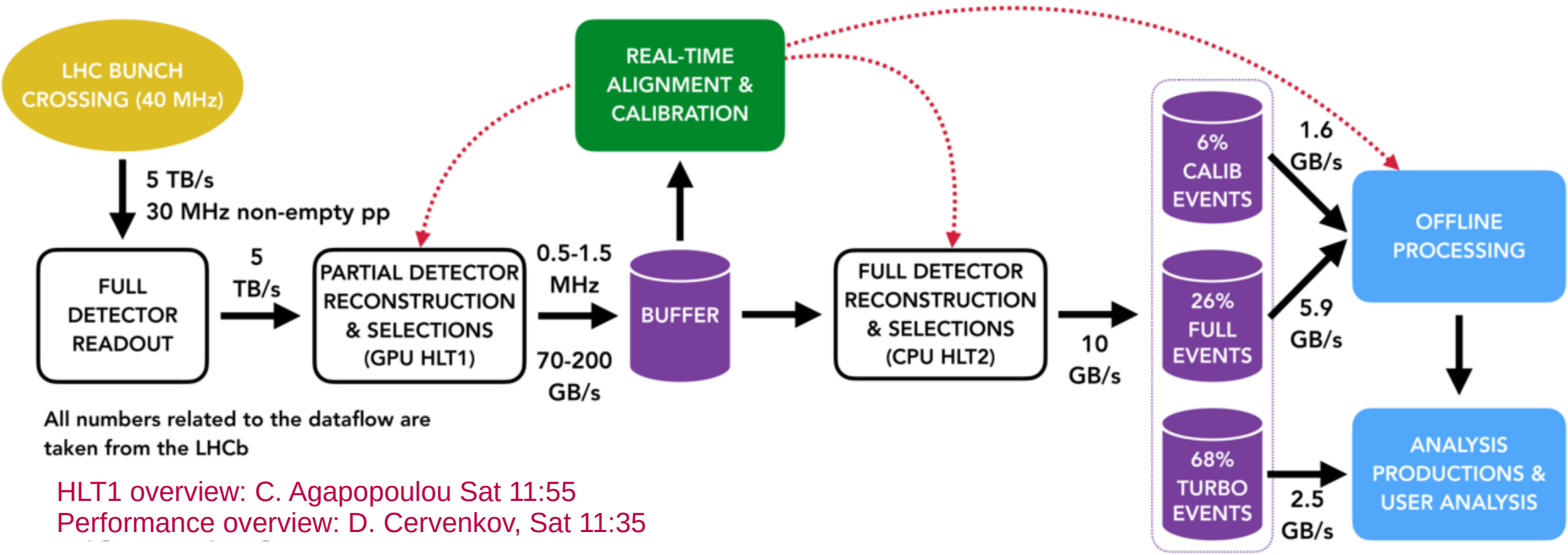
Trigger strategies

- Luminosity of 2×10^{33} cm^{-2}s^{-1}, $\sqrt{s}$ = 13.6 TeV, visible collisions per bunch $\mu \sim 5$
  - Almost all events will produce heavy quarks (*b*, *c*)
- Hard constrains: Bandwidth [GB/s] ≈ Accept Rate [kHz] × Event size [kB]



LHCb trigger design

- LHCb is using a software-only trigger for Run 3
- Full offline-quality reconstruction in real time at the trigger level

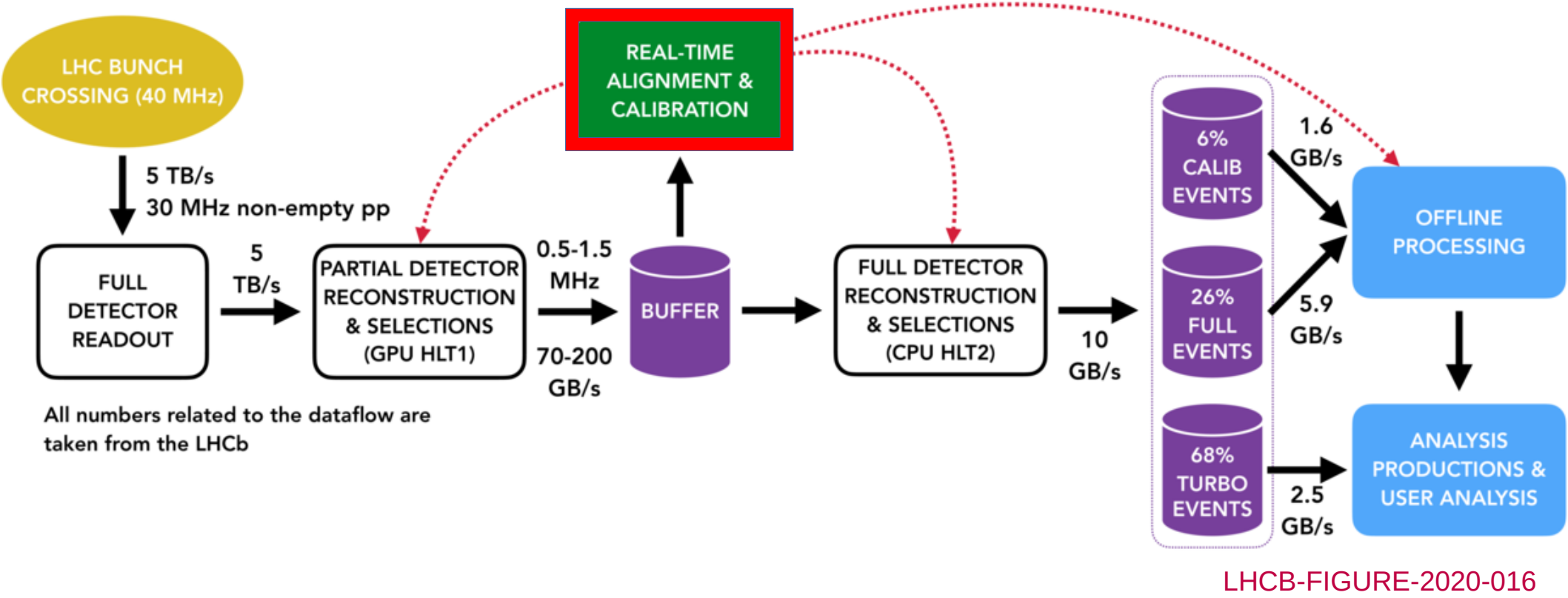


HLT1 overview: C. Agapopoulou Sat 11:55
Performance overview: D. Cervenkov, Sat 11:35

LHCB-FIGURE-2020-016

- Modern computing approach for HLT2 (CPU):
  - Multi-threading, task-based scheduling of algorithms, vectorization, ...

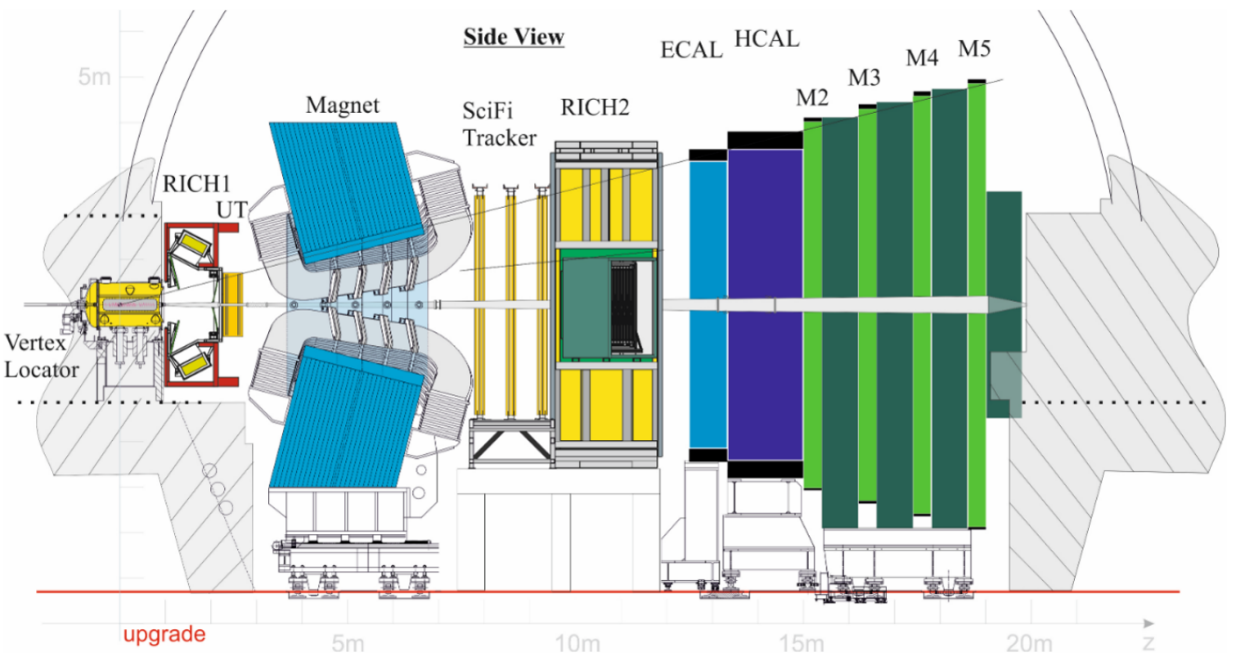
LHCb trigger design – Alignment and calibration



LHCB-FIGURE-2020-016

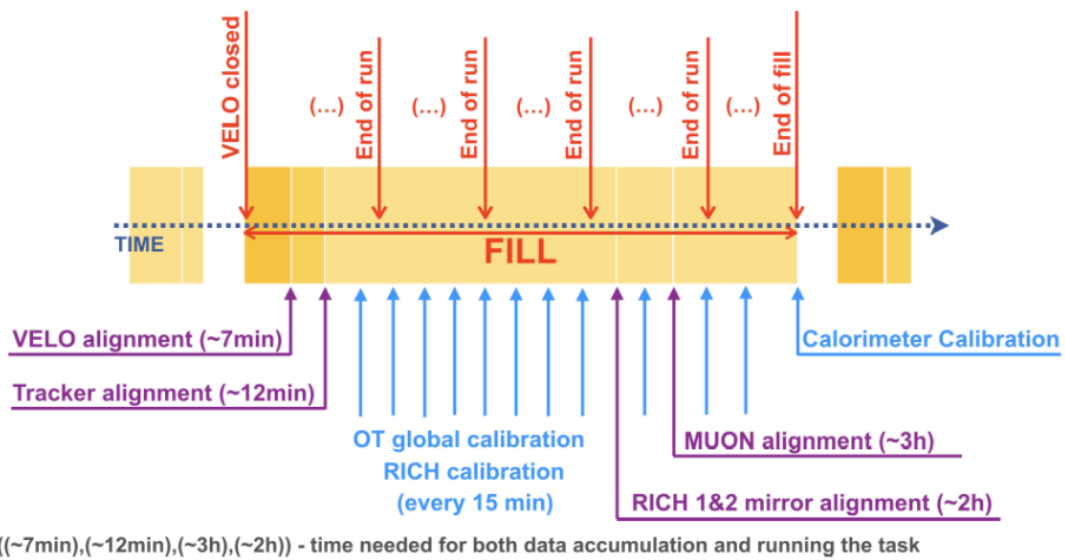
Alignment and calibration

- Fully aligned and calibrated data needed before running HLT2
- Online alignment and calibration pioneered in Run 2, crucial in Run 3
- Buffer capacity of O(10 PB), corresponds to O(days) of running, situated between HLT1 and HLT2
- LHCb distinguish two processes:
  - Alignment: VELO, RICH mirrors, UT, SciFi, Muon
  - Calibration: RICH, ECAL, HCAL



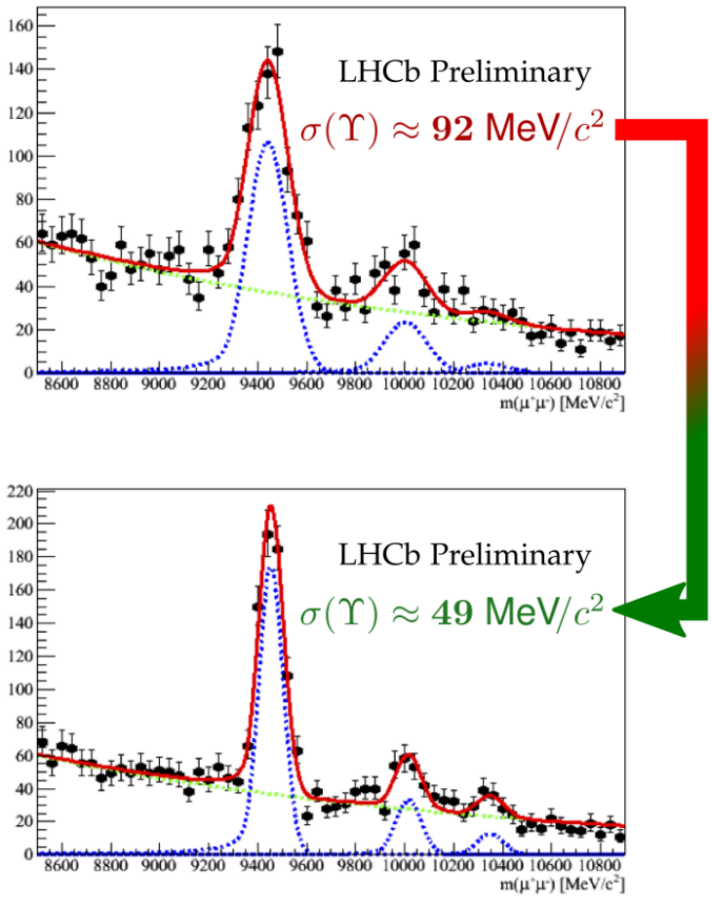
Alignment and calibration

- Fully aligned and calibrated data needed before running HLT2
- Online alignment and calibration pioneered in Run 2, crucial in Run 3
- Buffer capacity of O(10 PB) situated between HLT1 and HLT2
- LHCb distinguish two processes:
  - Alignment: VELO, RICH mirrors, UT, SciFi, Muon
  - Calibration: RICH, ECAL, HCAL



Run 2 timing

Run 2 results:

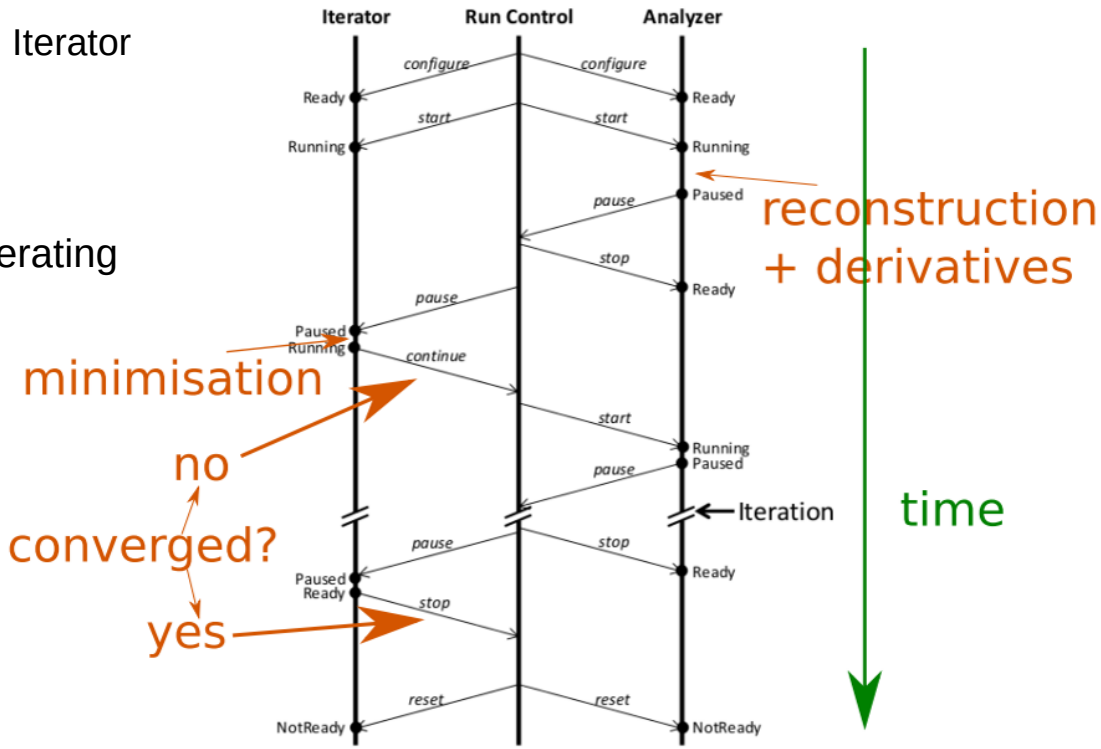


J. Phys.: Conf. Ser. 664 082010

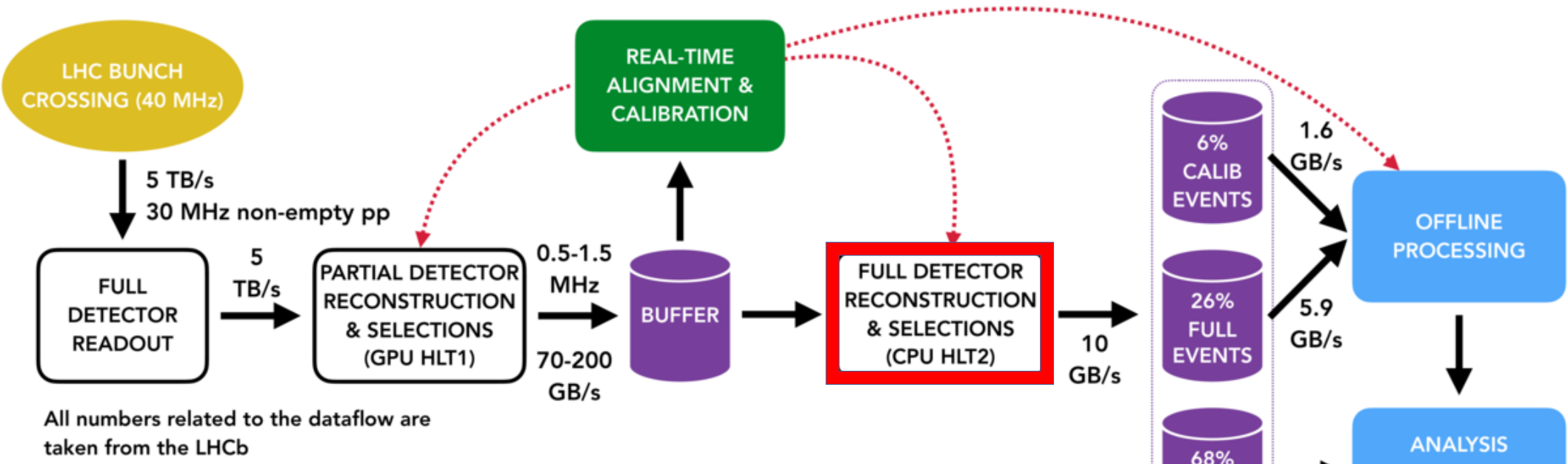
Alignment and calibration

- Alignment and calibration procedure is to be run multi-threaded on roughly 160 nodes
- Based on a dedicated set of HLT1 events (special HLT1 selection lines for Alignment)
- Implemented as finite-state machine steered by Run Control (fully integrated into ECS)

- Alignment procedure is based on Analyzer and Iterator
- Analyzer:
  - Run reconstruction
  - Calculating alignment constants and iterating
- Iterator:
  - Collect derivatives/histograms
  - Obtain new constants
  - Convergence check



LHCb trigger design - HLT2



All numbers related to the dataflow are taken from the LHCb

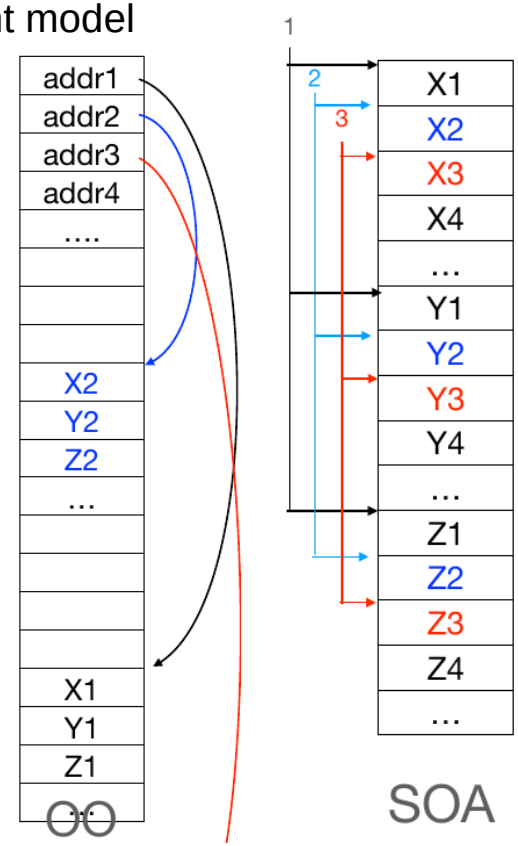
LHCB-FIGURE-2020-016

SoA event model

- Important part of the Upgrade is to switch between v1 (OO) and v2 (SoA) event model
  - OO: standard object oriented programming approach
  - SoA: structure of arrays

- Very different memory behaviour of different models:
  - OO: many small memory allocations, random jumps, copying objects
  - SoA: only reads what will be used, easily vectorisable

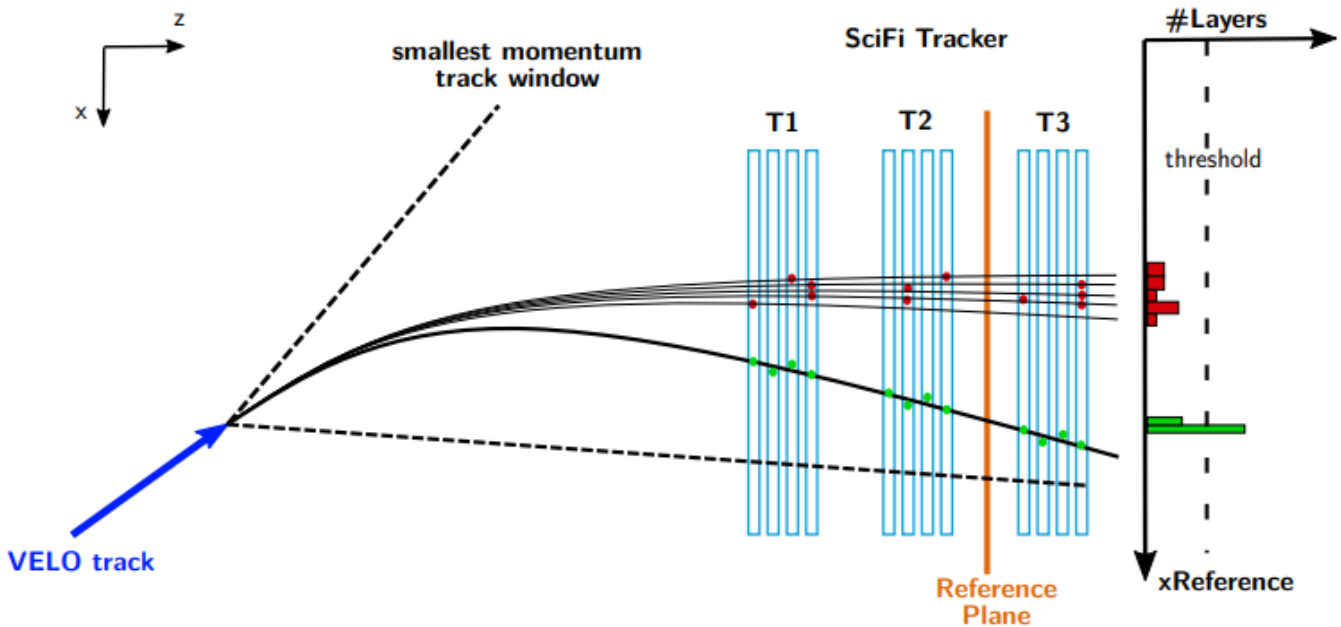
- SoA works well with the SIMD (single instruction multiple data) approach
- However, SoA-based computing differs significantly in how data are accessed
  - Only a slices of SoA-collection are accessed, no objects (in OO terms)



- <u>SoA event model is crucial to fully utilise vectorisation and achieve the required HLT2 throughput</u>

Vectorisation in HLT2

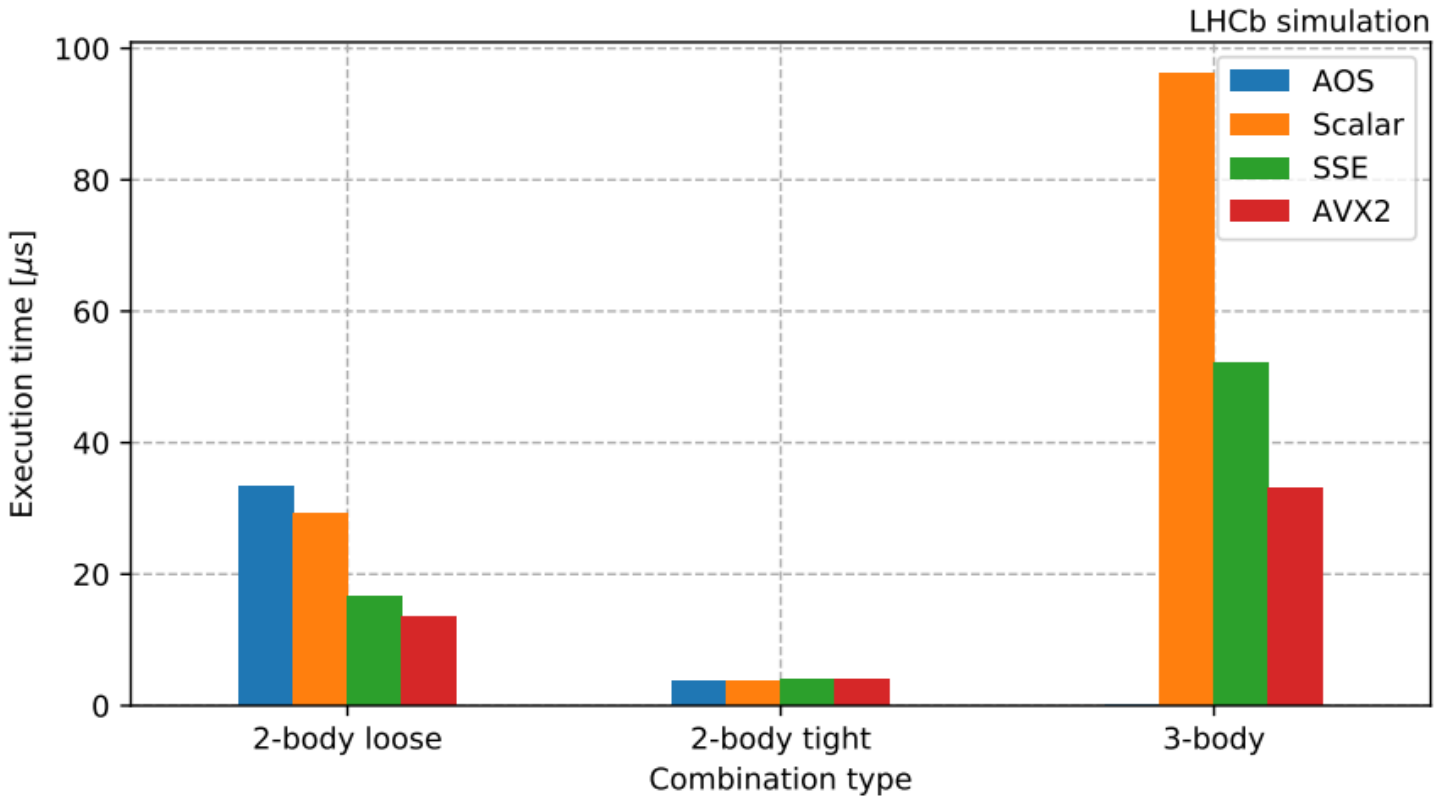
- Hits and any information are stored in SoA layout
  - Ideal place where to apply Single Instruction Multiple Data (SIMD) approach
- Example from forward tracking in HLT2
  - Several thresholds can be scanned in parallel
  - 8 single precision floats/integers in parallel (AVX2)
  - Increases event throughput of Forward Tracking by 60%



A. Günther CTD 2022

Vectorisation in HLT2

- Vectorisation directly useful also for the selection algorithms
- Comparison of speed between AoS and SoA data models using a scalar instruction set
- Also two SoA-based vector instruction sets, SSE and AVX2, are compared
- Gain in speed up to 65 %



LHCB-FIGURE-2020-018

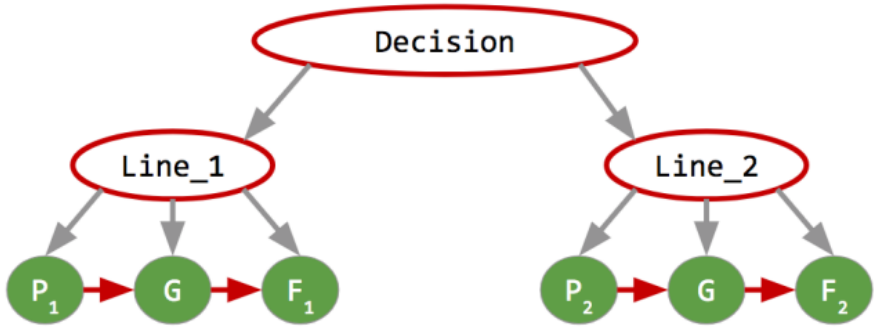
Throughput oriented selection

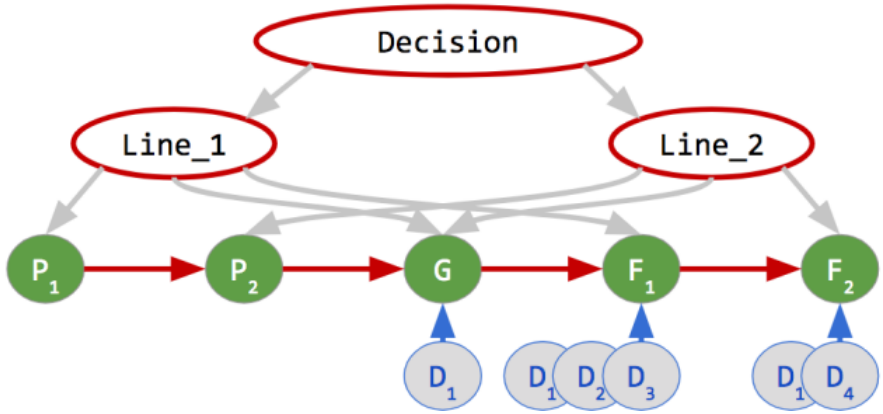
- In Run2, reconstruction was about 70% and selection 30% of time spent
- Throughput Oriented (ThOr) functors (function objects) are designed to be agnostic to input / output type and to be flexible on what they operate on.
- Functors are composable allowing a simple chaining of basic functors
  - e.g. X @ POSITION @ VERTEX ⇒ Particle.vertex().position().x()
- Simultaneously developed for old and new event model
  - Significant gain when using SoA model
- Using functor cache instead of JIT compilation
  - Functors that are defined in python during build ⇒ compiled into a cache in
- Compile time to be used natively in the application.

Compile	Basic compiler (s)	Just in time compilation (s)
Single functor	**11**	**13**
5 different functors	**20**	**13**
Typical HLT2 selection	**70**	**24**
Compilation rerun	**70**	**0**

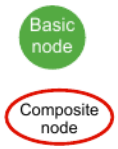
Event scheduler

- Dealing with O(1000) of dedicated selection lines
- Multi-threading friendly algorithms needed
- Automatic handling of data and control flow
  - Data flow: Configurable properties with user defined input/output
  - Control flow: what to run and where to stop
- Handle the data flow with specific logical types
  - Order the basic nodes with specific control flow
- Configured during initialization
- Automatically resolve data dependencies by matching input / output
- Static graph with ordered nodes (respecting data constrains)





- Basic node: one algorithm with data dependencies
- Composite node: logic operation (AND, OR, NOT)

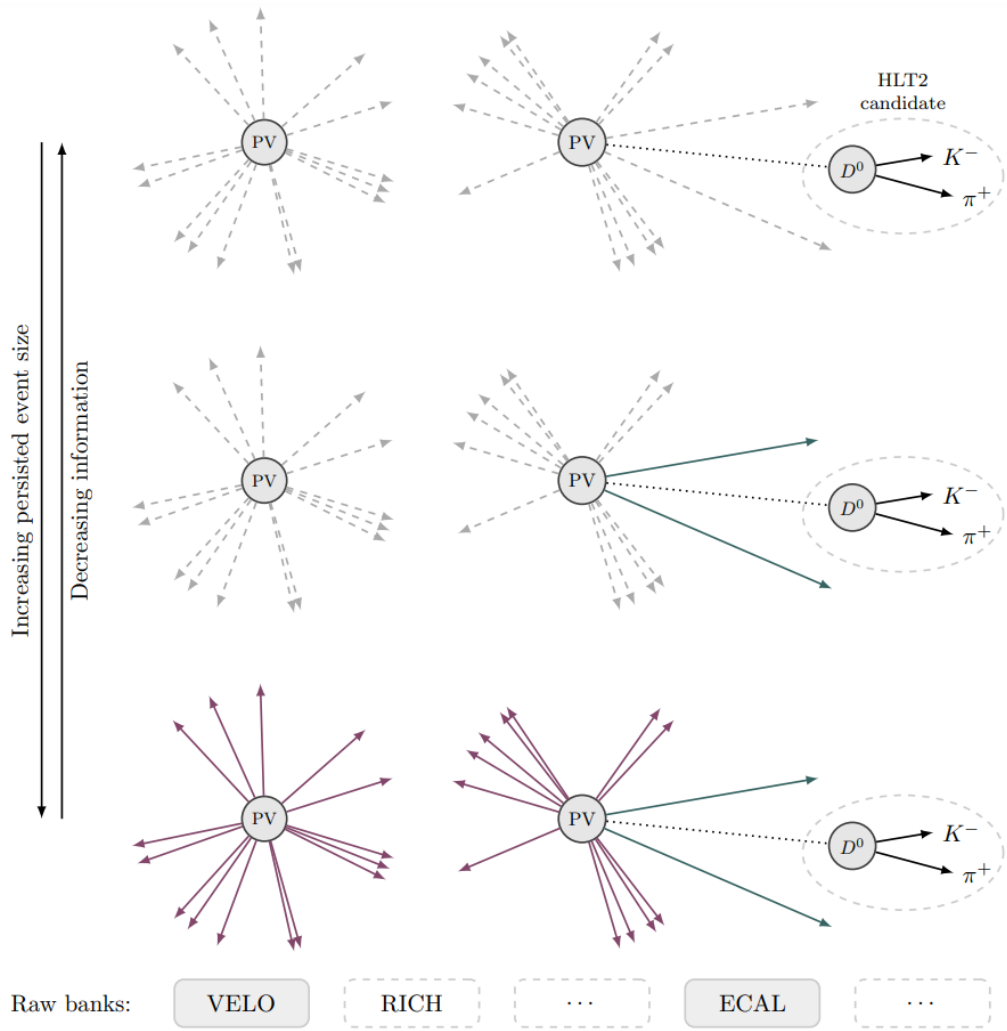


J. Phys.: Conf. Ser. 1525 012052

Turbo model

- Given the hard limit on bandwidth and expected event rate, event size is the only free parameter
- Instead of saving of full event, only information needed for a physics analysis can be stored
- Extensively used during the Run II
  - Around 30 % of the trigger rate is Turbo – almost all Charm physics
  - But only about 10 % of the bandwidth!
  - Approximately 2/3 lines keep raw detector information (Turbo SP)
- Significant reduction of data size ⇒ more events at same bandwidth
- <u>Baseline approach for Run 3</u>

Persistence method	Average event size [kB]
Turbo	**O(10)**
Turbo++/SP	**O(10-100)**
Raw event	**O(100)**

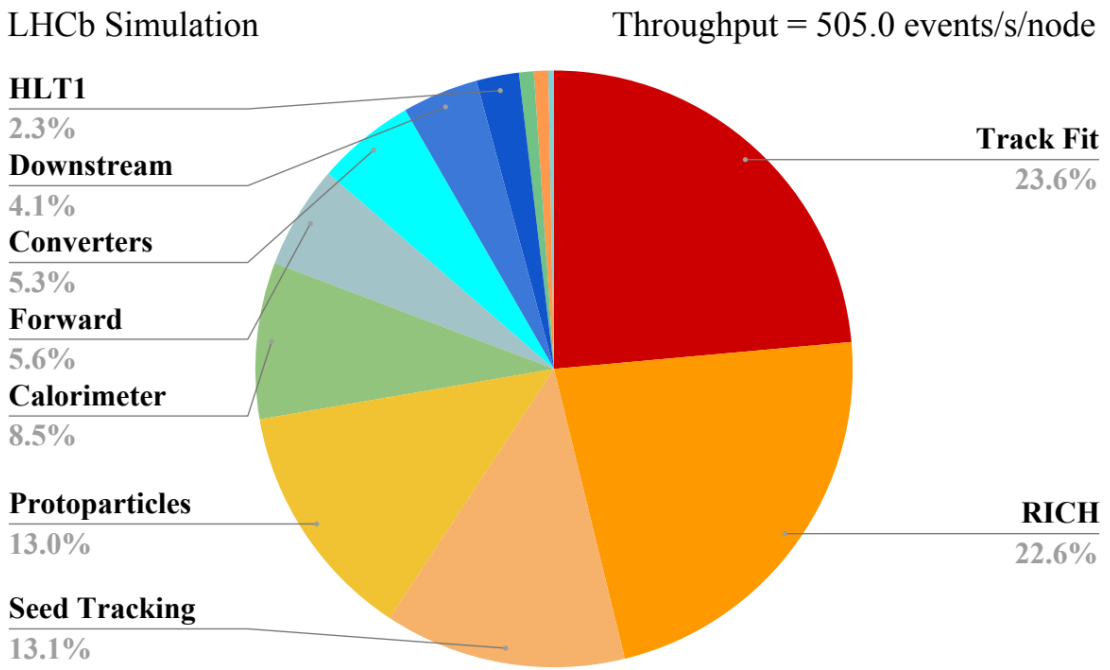


JINST 14 (2019) 04, P04006

HLT2 throughput

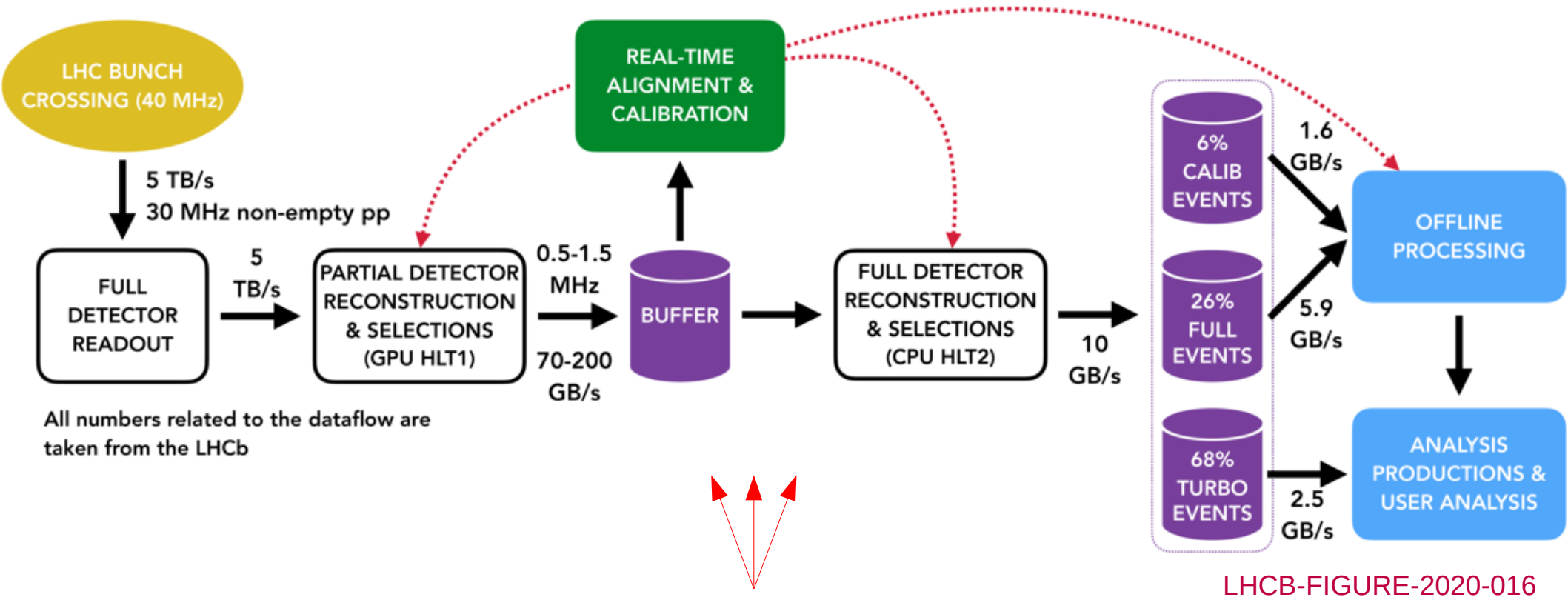
- Full HLT2 throughput
  - Physics selection not included

- Optimised sequence:
  - Removing the redundancy in the reconstruction of Long Tracks
  - Using a partially parametrised Kalman Filter (material scattering)
  - Additional improvements in matching between tracks and ECAL clusters.



LHCB-FIGURE-2022-005

LHCb trigger design - software QA



LHCB-FIGURE-2020-016

Software QA, testing and training

Software quality-assurance

- A dedicated working package with the goal is to improve and maintain the quality assurance of code
  - This includes also work on relevant documentation
- The LHCb software stack is a highly modular system based on Gaudi with code being hosted on GitLab
  - Large base of developers, concurrent development of parts of code base.
  - Submission of new code ⇒ Merge Request (MR).
- Changing one part may affect on rest of stack, such an impact may often be hidden

- As any large project, LHCb has an internal review policy for any contribution
  - Any MR must be reviewed before merging

- System depends on consisting of maintainers (experts) and shifters (junior members)
  - This assures that each line of code is fully reviewed by a relevant experts and senior LHCb software expert (maintainer)
  - Shifter helps with checking requirements of each MR and evaluating tests
  - Each contribution should be written as accessible to shifters who then can learn more about the LHCb code base – ideal place to learn both about LHCb software and computing

Testing infrastructure at LHCb

- Testing infrastructure has two main parts:

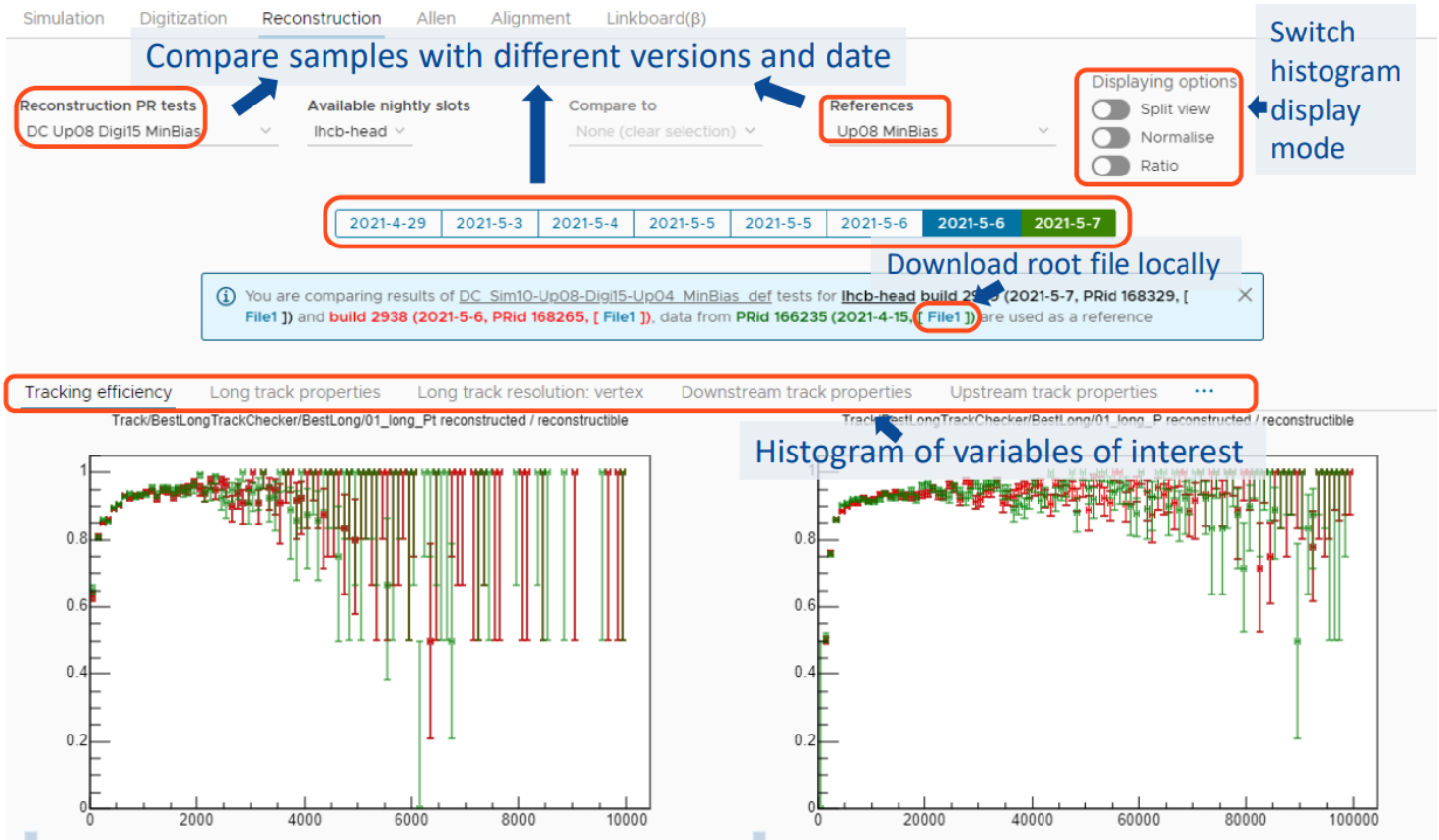
- The LHCb nightly build system
  - Compile & Test & Compare: Built? Ran? Finalized? (code error or not)
  - O(300) cores, jobs managed by Jenkins
  - Can be run directly from GitLab using web-hooks for any MR

- The LHCb Performance Regression (LHCbPR)
  - Utilities same infrastructure as nightlies
  - Focus on physics variables (momentum, tracking efficiency, vertex…)
  - Configured by python scripts
  - The LHCbPR front-end (browser-based)
  - Quickly check and comparison of test results (histograms)

Modernisation of LHCb CI system: M. Szymanski:Computing track, S at. 16:00

Reconstruction dashboard

- Additionally a high-level physics properties can be obtained from nightlies and compared between each other and / or references
- Results are then visualised and accessible via LHCbPR front-end
- Used extensively to compare results between various HLT2 settings and evaluate their impact



Software training

- Extremely important for any code development is not to only gather experts but also to pass the knowledge
- LHCb organized 25 dedicated upgrade software hackathons during the last 6 years

- Development of the new framework and training of a new contributors to all relevant aspects
  - Modern computing methods in general, heterogeneous (GPU) programming, FPGAs, …
- These skill are necessary for any modern HEP experiment, but (often) not taught at universities or even recognized in hiring / promotion
- Community-wide effort needed to train and keep those who decided focus also on computing aspects

Summary

Summary

- The LHCb experiment was upgraded during the last few years resulting into a new experiment at LHC
- Flexible software trigger is the nominal strategy for Run 3

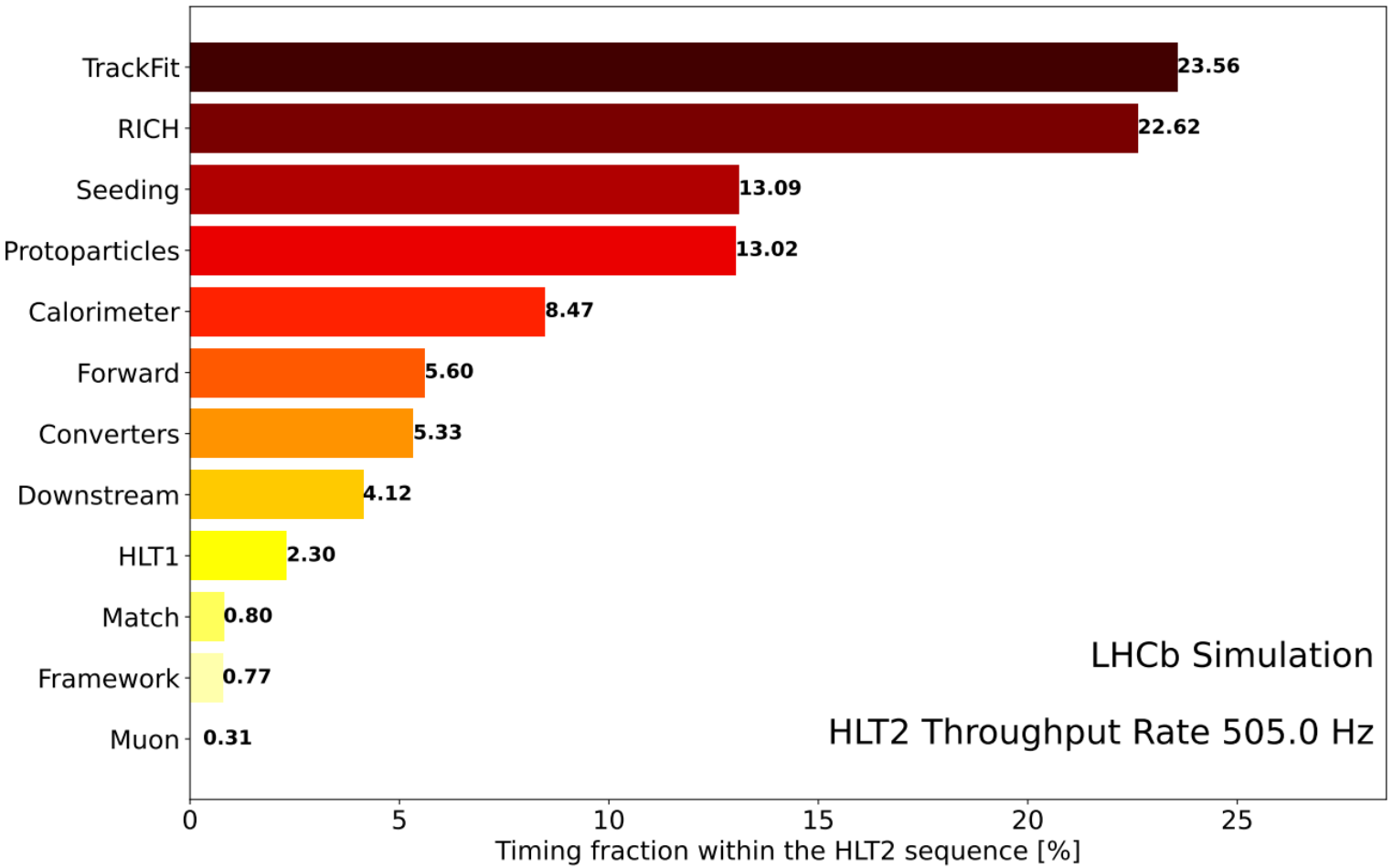
- HLT2 is based on modern computing methods: SoA, vectorisation, multi-threading
- Turbo model is a baseline for Run 3 allowing to record higher statistics at same bandwidth

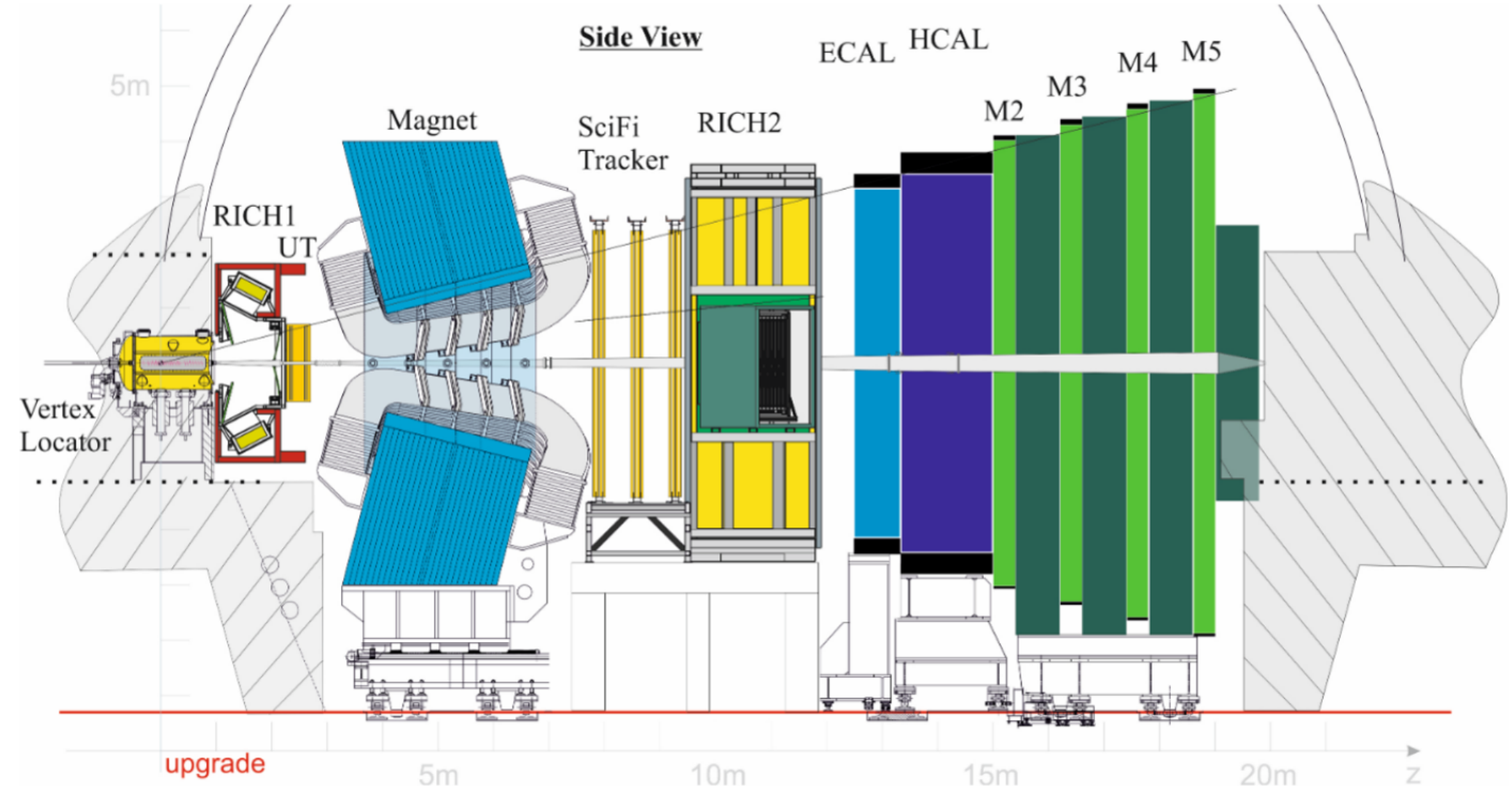
- Declared goal of running HLT2 at 500 Hz was successfully achieved

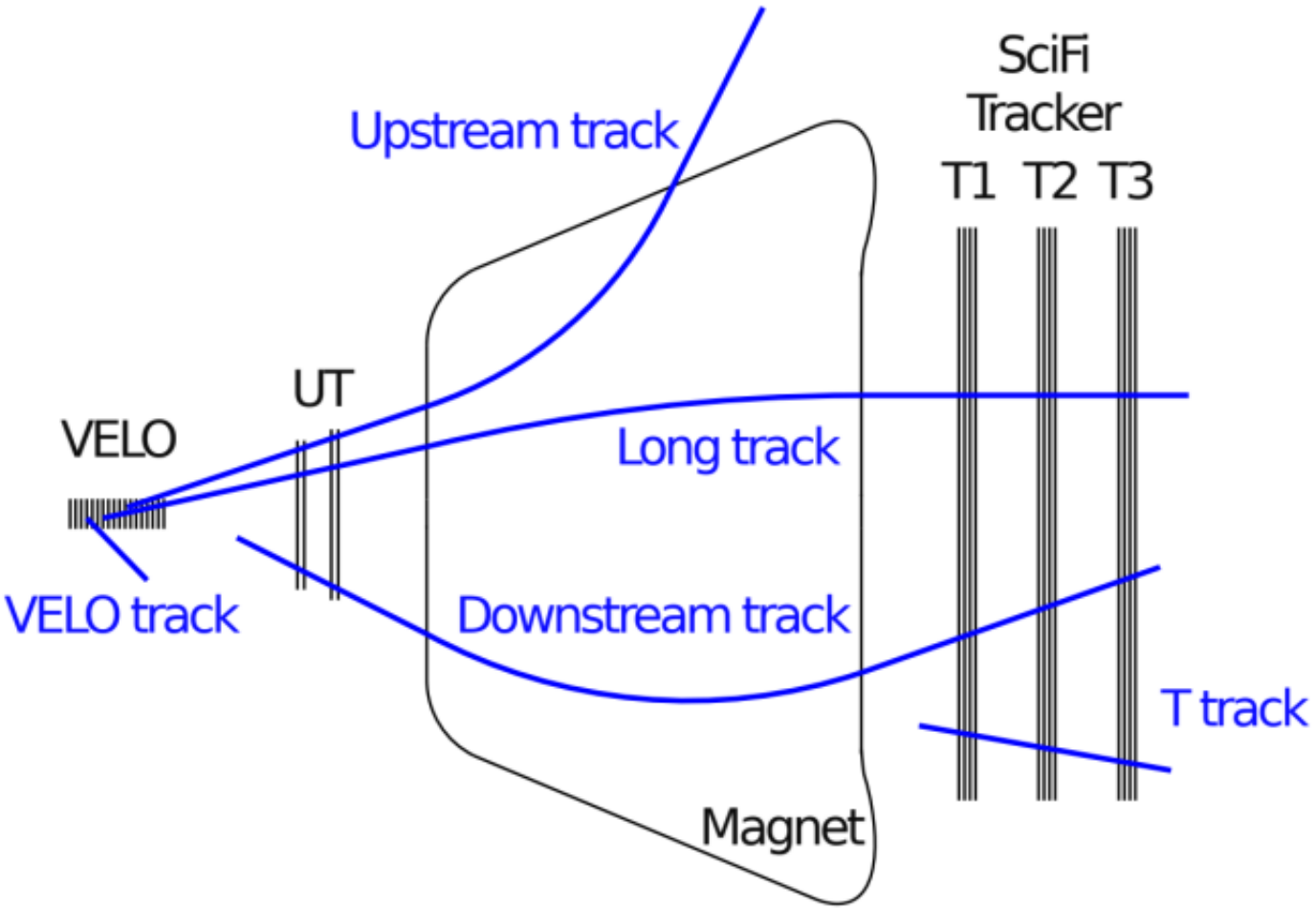
- Extended QA and testing system is well established part of the development cycle

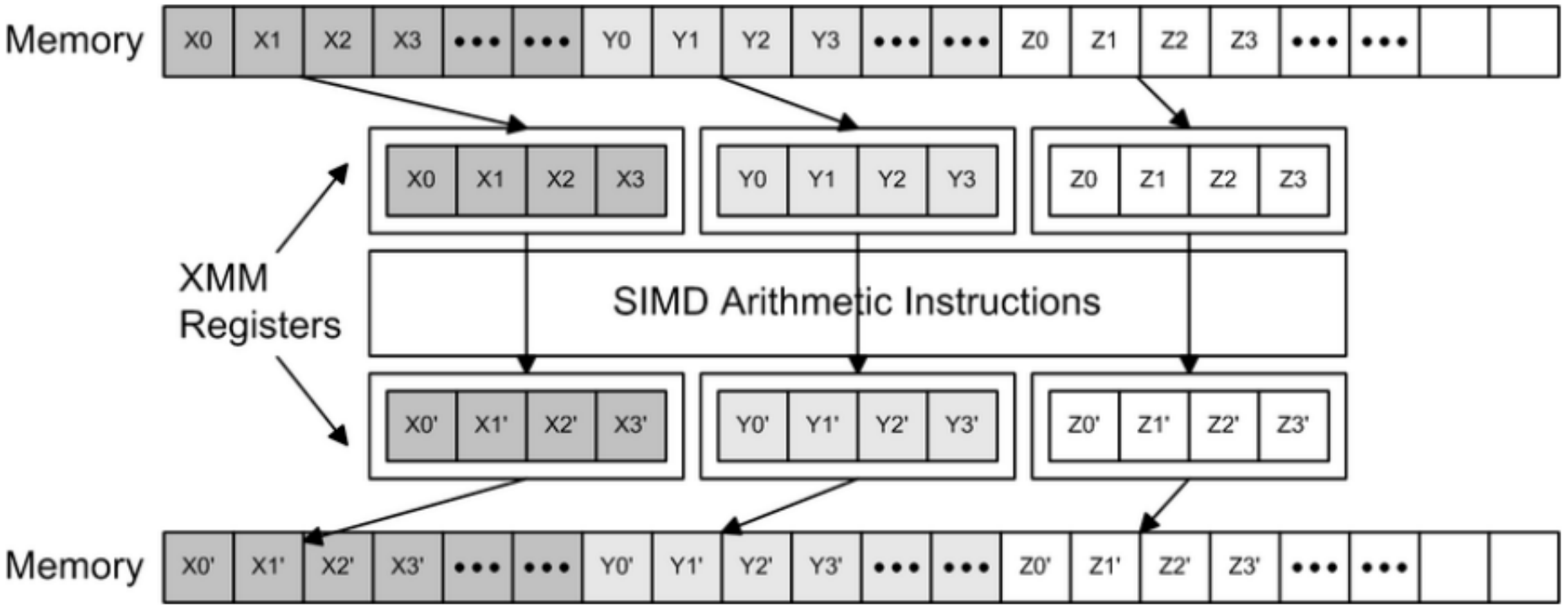
- HLT2 is fully working and ready for Run 3 data

Spare slides


TrackFit
RICH
Seeding
Protoparticles
Calorimeter
Forward
Converters
Downstream
HLT1
Match
Framework
Muon
23.56
22.62
13.09
13.02
8.47
5.60
5.33
4.12
2.30
0.80
0.77
0.31
LHCb Simulation
HLT2 Throughput Rate 505.0 Hz
0
5
10
15
20
25
Timing fraction within the HLT2 sequence [%]

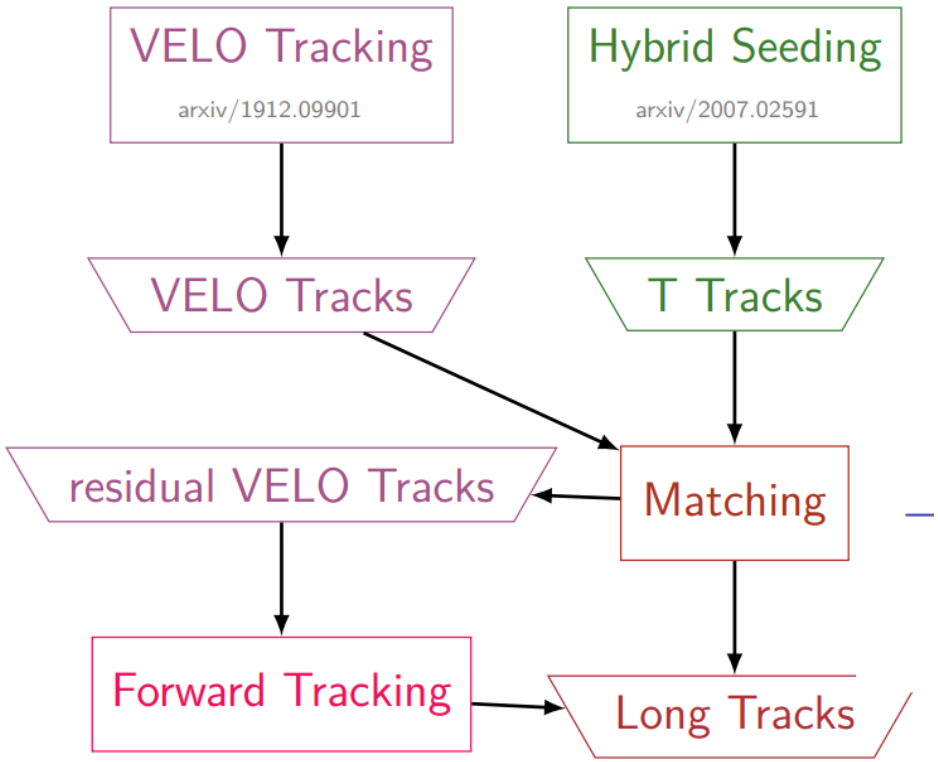

Side View
ECAL
HCAL
M2
M3
M4
M5
5m
Magnet
SciFi
Tracker
RICH2
RICH1
UT
Vertex
Locator
upgrade
5m
10m
15m
20m
z


SciFi
Tracker
T1 T2 T3
Upstream track
UT
VELO
Long track
VELO track
Downstream track
T track
Magnet


Memory
X0 X1 X2 X3 ••• ••• Y0 Y1 Y2 Y3 ••• ••• Z0 Z1 Z2 Z3 ••• ••• ••• •••
X0 X1 X2 X3
Y0 Y1 Y2 Y3
Z0 Z1 Z2 Z3
XMM
Registers
SIMD Arithmetic Instructions
X0' X1' X2' X3'
Y0' Y1' Y2' Y3'
Z0' Z1' Z2' Z3'
Memory
X0' X1' X2' X3' ••• ••• Y0' Y1' Y2' Y3' ••• ••• Z0' Z1' Z2' Z3' ••• ••• ••• •••

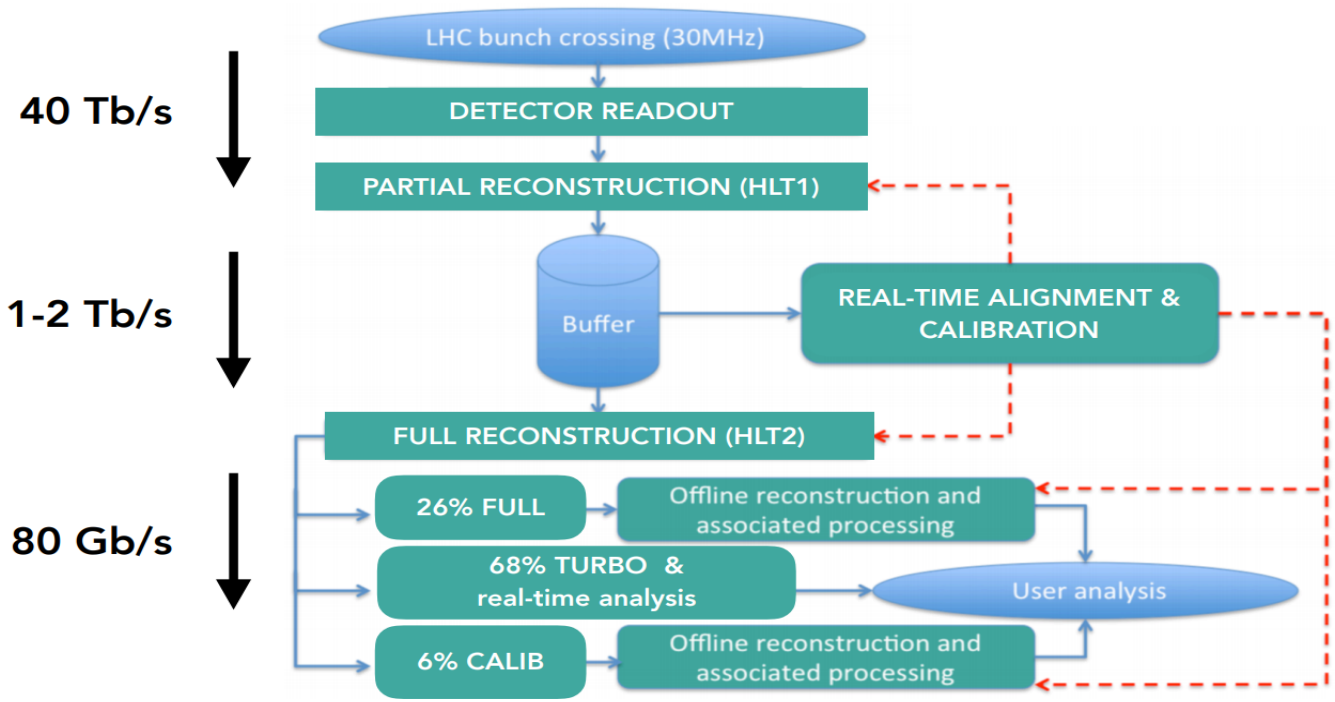
HLT2 Forward tracking

1) Define hit search window
2) Treat magnet as optical lens to simplify track and hit projection
3) Hough-like transform: project all hits in window to reference
4) Plane and count number of SciFi layers in histogram
5) Scan histogram, collect hits from bins above threshold
   - found set of SciFi hits extending VELO track
6) Clean-up hit set and fit using 3rd order polynomial
7) Estimate q/p from fit result



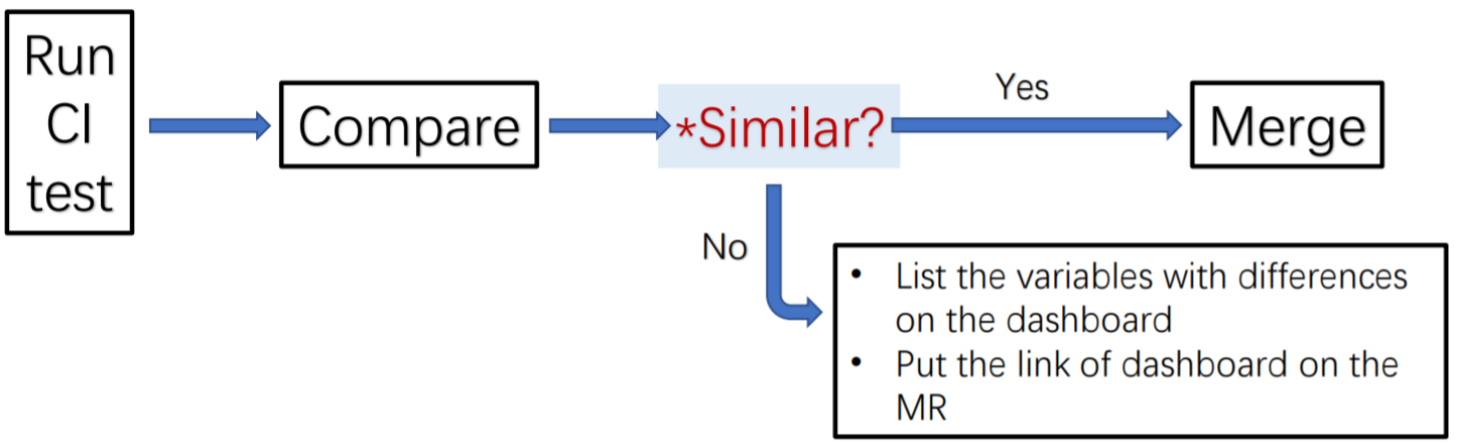
LHCb Upgrade I

- Real-Time Analysis – efficient decision about data in the full online mode
- Keeping only a signal and suppress any unnecessary information about event
- Continuous readout, full software trigger at visible collision rate of 30 MHz



Automatic testing of new MR

- Already existing system is still being updated with a new features
- Aim is to provide evaluation of physics performance (above the number of processed HLT1/2 events) of any new MR to any RTA related projects
- Test is based and extends current implementation of ci test on gitlab



- Difference between two histograms will be computed as the sum of the differences in an absolute value per bin
- System is currently under development
- Automatic testing allows us to test larger amount of possible changes than manual check – however all the resulting discrepancies must be evaluated by shifters and developers