LHCb HLT software and GPU algorithm

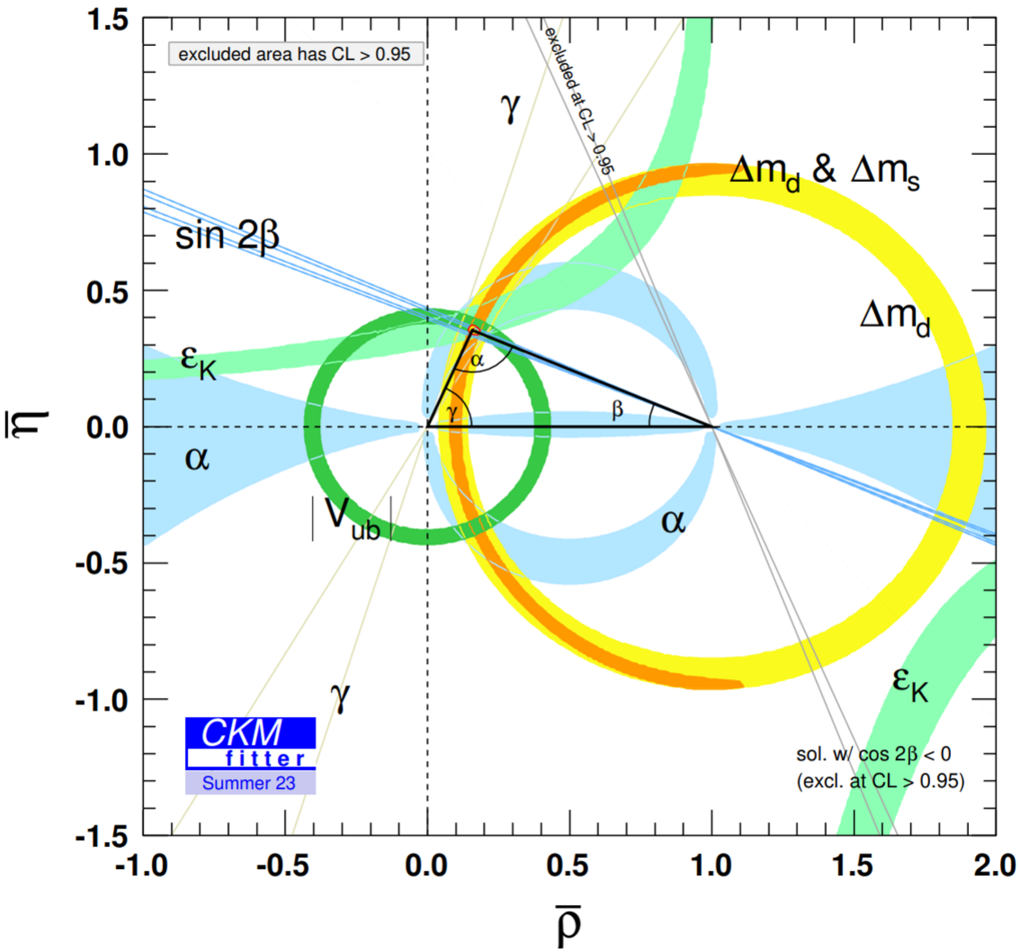
Miroslav Saur on behalf of the LHCb collaboration
(Lanzhou University)

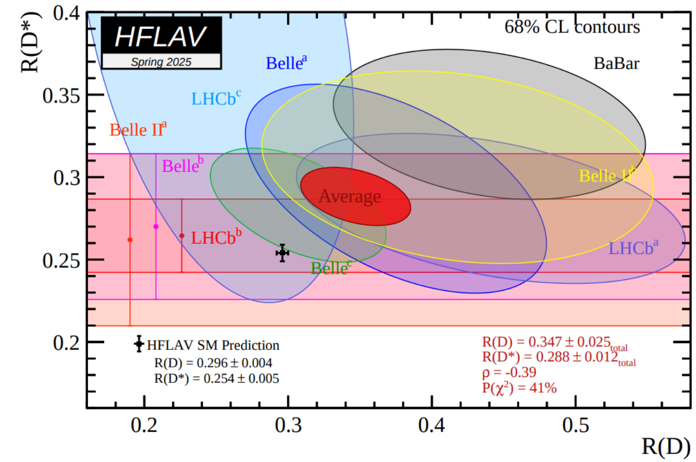
The 2025 International Workshop on the High Energy Circular Electron Positron Collider
Guangzhou, China

2025/11/09

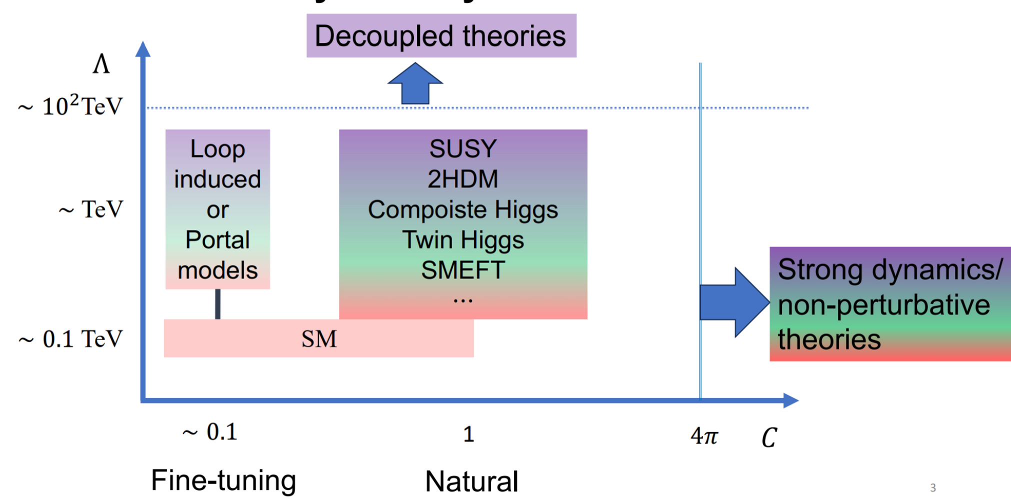
Necessity of recording more data

- Many physic observable in high energy physics are still limited by available statistics ⇒ more data required





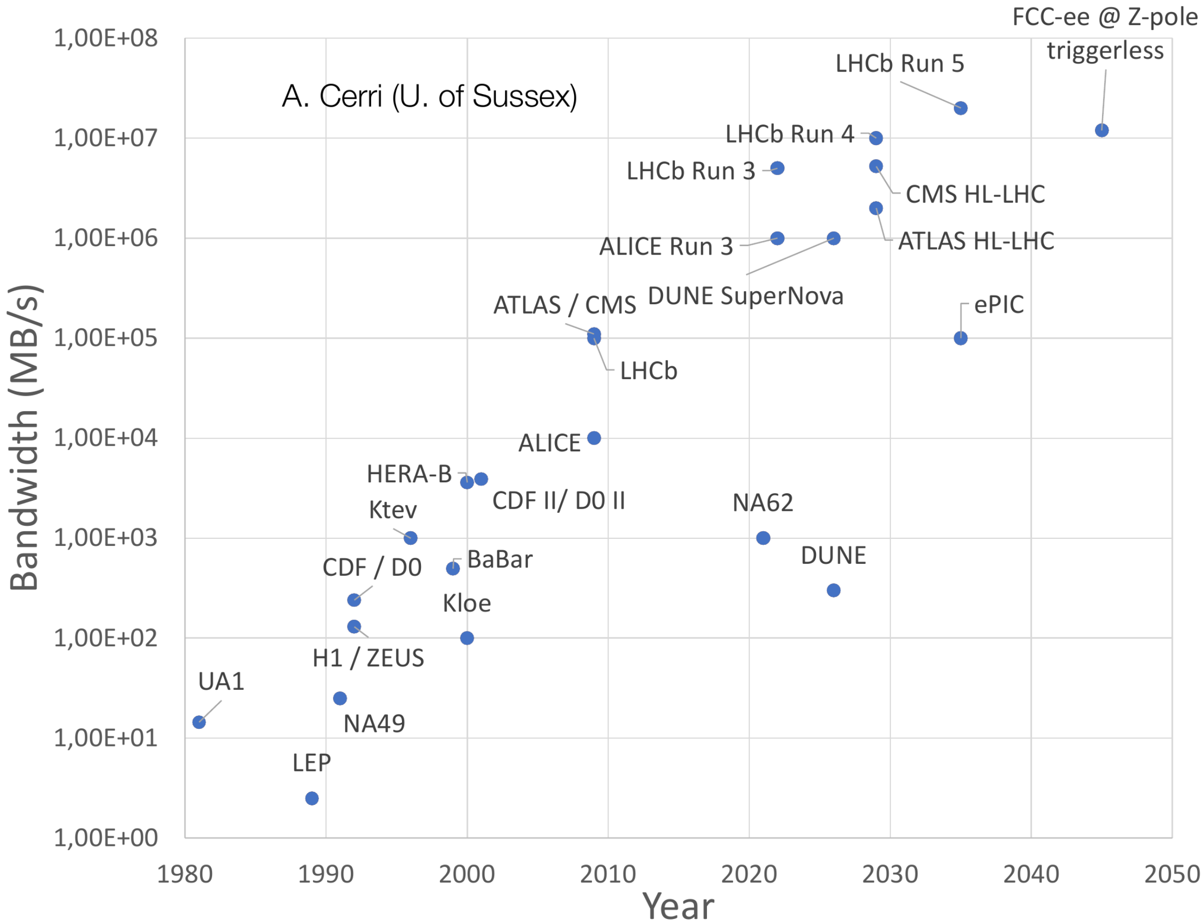
New Physics beyond Standard Model



Yandong Liu Bijing Normal University

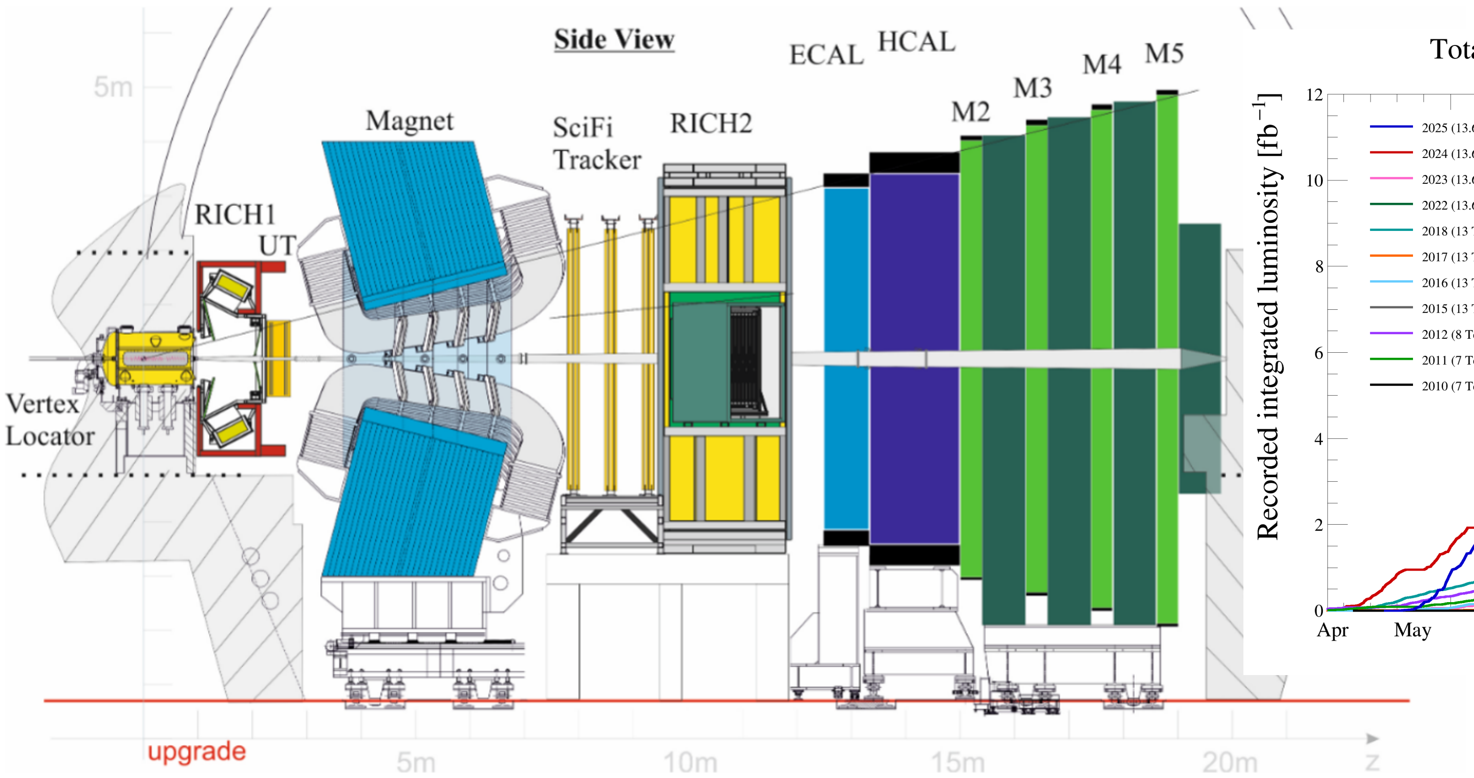
HEP trigger strategies

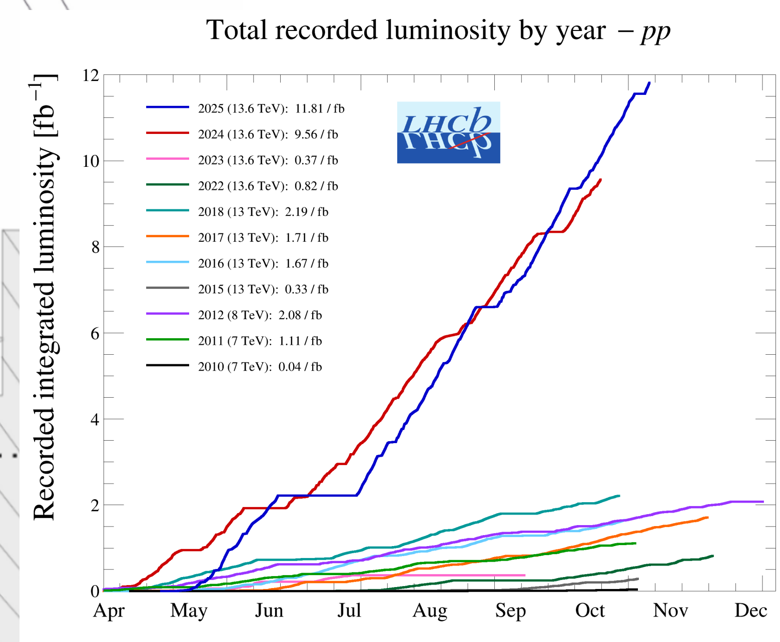
- Many physic observable in high energy physics are still limited by available statistics ⇒ more data required
  - Almost every pp collision is interesting for LHCb as is contains a heavy quark (*b*, *c*)
  - Resulting into the highest online bandwidth numbers ⇒ new computing model needed



LHCb experiment in Run 3

- LHCb conditions in Run 3: luminosity of 2×10^{33} $cm^{-2}s^{-1}$, $\sqrt{s}$ = 13.6 TeV, visible collisions per bunch $\mu \sim 5$
- New tracker detectors, upgraded electronics, fully software trigger, ...
- LHCb Upgrade I: A new general-purpose forward-region detector at LHC

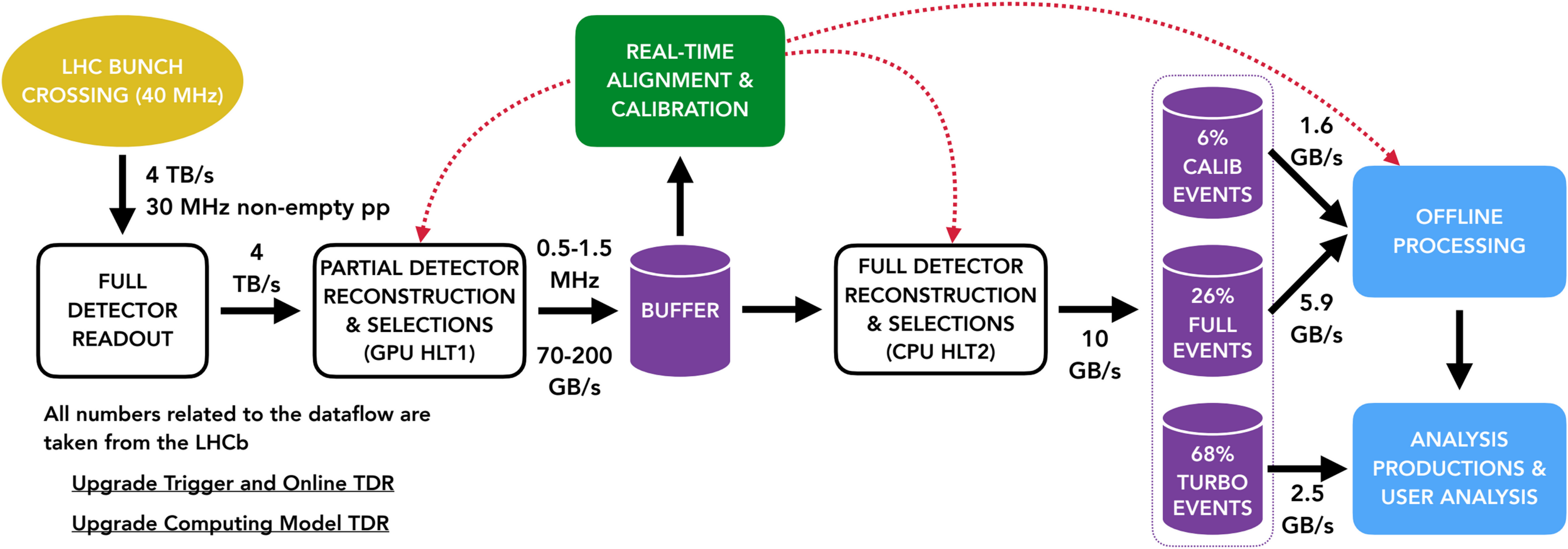




- More details about hardware and physics part of LHCb Upgrade I and II in talk by Xuhao Yuan, Flavour session 07/11 14:25

LHCb trigger design

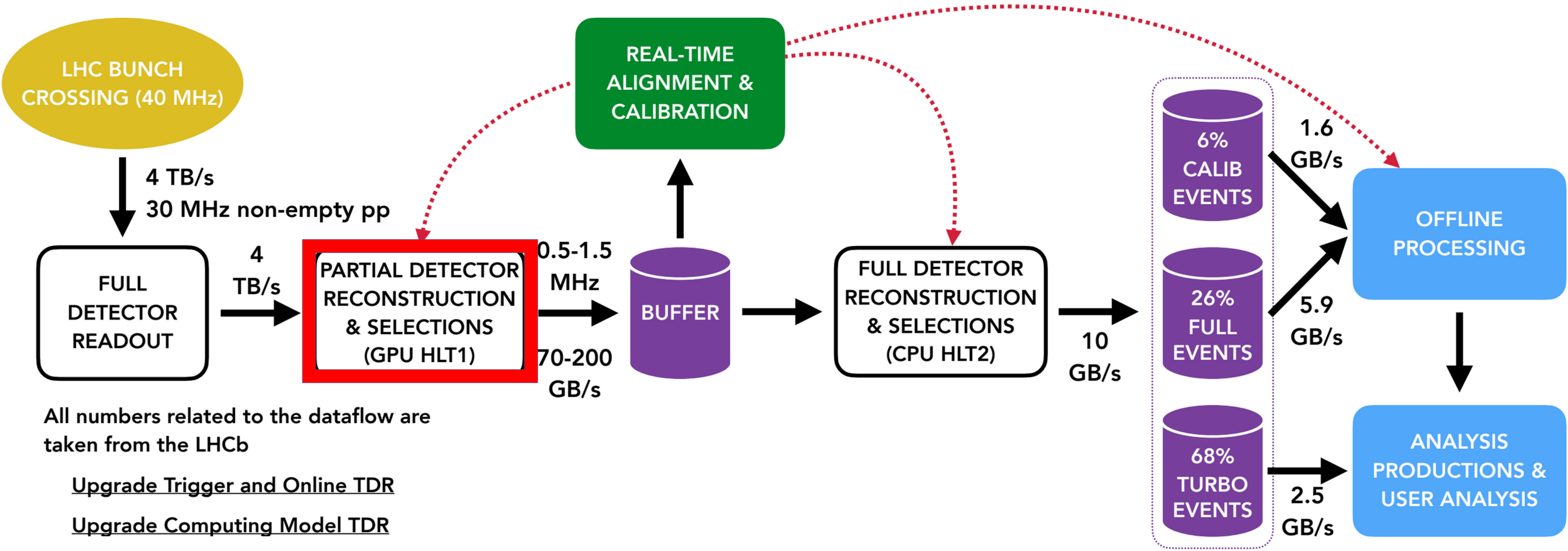
- Real-Time Analysis paradigm: full offline-quality reconstruction and selection achieved at the trigger level
- Online and offline processing are using same code base



- Buffer of O(40 PB) between HLT1 and HLT2 allows asynchronous processing
  - Corresponds to around 10 days of LHC production without running HLT2
  - Ouf-of-fill processing, more stable usage of HLT2 farm

LHCb-FIGURE-2020-016

LHCb trigger design: HLT1



All numbers related to the dataflow are taken from the LHCb

Upgrade Trigger and Online TDR

Upgrade Computing Model TDR

LHCb-FIGURE-2020-016

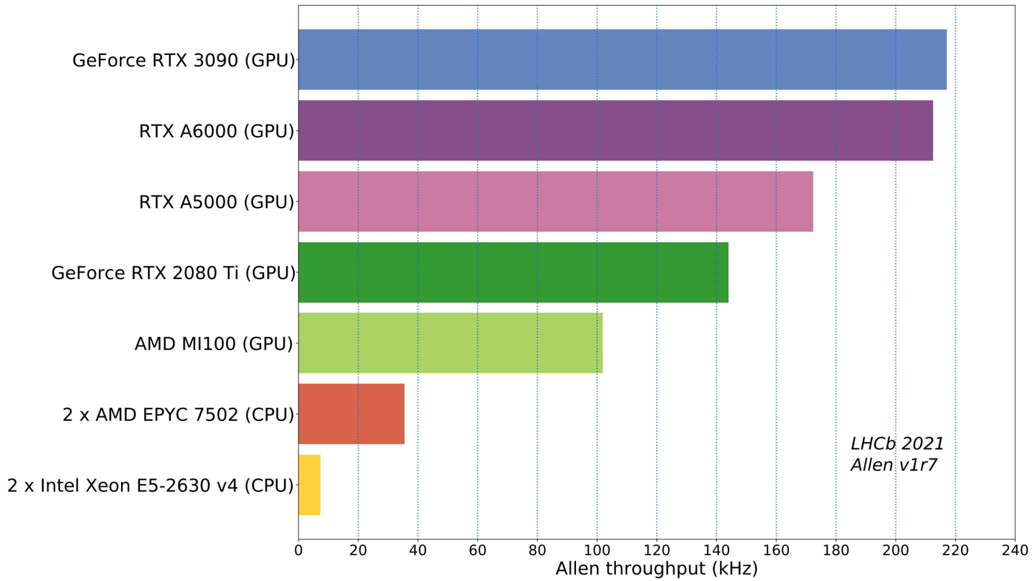
- More details about online and DAQ in talk by Guoming Liua, TDAQ session 09/11 11:15

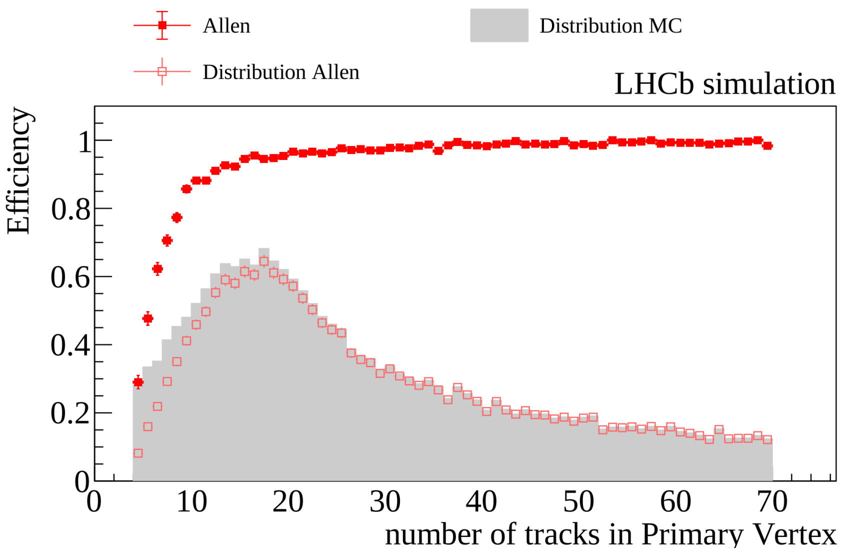
HLT1: Why GPUs?

- LHCb had been pursuing GPU reconstruction algorithms since 2012, and by 2014 most of work on the vertex detector reconstruction algorithm and associated infrastructure done
- However porting single algorithms to GPUs was not going to work: no single algorithm was expensive enough to make an “off loading” model cost-effective, and no model for multi-event processing on a GPU at that time
  - Off loading is instead used at ALICE and CMS for specific tasks
- At the end of 2017 it was decided to try to put the entire HLT1 on GPUs and develop dedicated multi-event scheduling
- During the same time, development of baseline HLT1-CPU was still ongoing but cost, especially networking, started to be a significant value
- GPUs are matching well with LHCb online architecture – using GPUs could significantly reduce networking cost
- Proved to be a successful decision leading to full implementation of HLT1 functionality on GPUs
  - The very first fully GPU-based trigger at HEP

HLT1: Overview

- The goal of HLT1 is to process of the LHCb raw data at 30 MHz and reduce rate by a factor of 30 (to 1 MHz)
- HLT1 is implemented in the form of Allen project [Comput. Soft. Big Science 4, 7 (2020)], using RTX A5000
  - Cross-architecture support: x86, CUDA/CUDACLANG (NVIDIA GPUs), HIP (AMD GPUs)
- Partial event reconstruction: vertexing, tracking, muon PID, simplified CALO information
- Rough selection based on O(50) trigger lines covering LHCb physics program
  - High/low pT muons, NN-based one-/two-track selection, detached lines, ...
- Required performance obtainable using O(200) GPUs, currently around 500 RTX A5000 installed

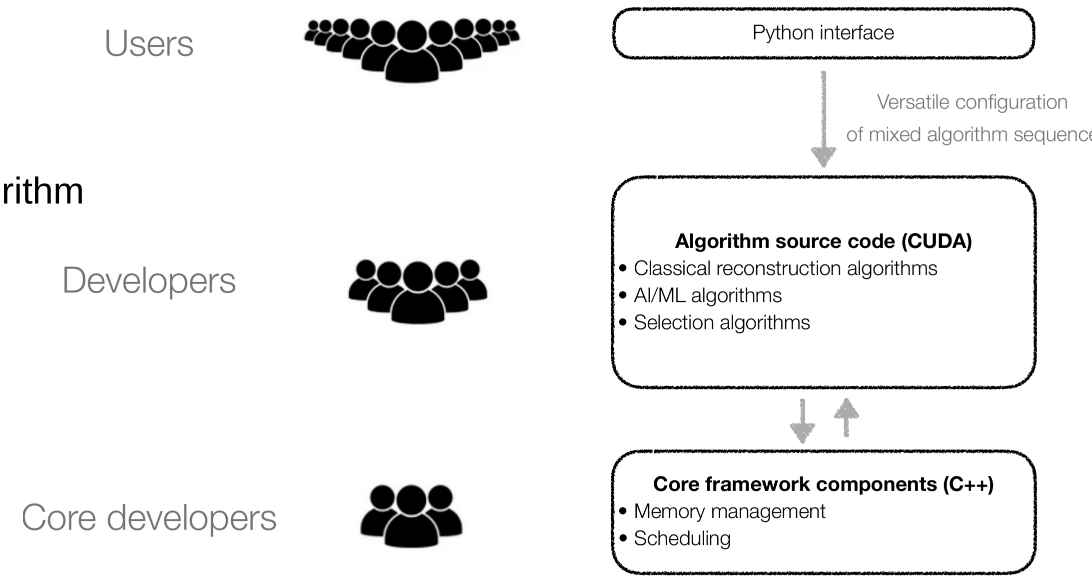




LHCb-FIGURE-2020-014

HLT1: Design approach

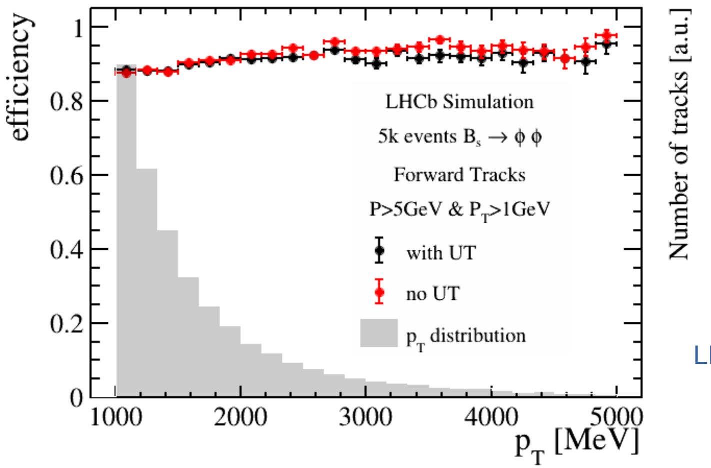
- Do as much as possible with data directly on GPU
  - Minimise copies to/from the GPU, I/O is one of the main bottlenecks
  - Allocate only necessary amount of memory
- Parallelise on multiple levels
- Maximise GPU algorithm performance
  - Design software framework to optimise algorithm throughput performance
  - Interleave ML/AI and classical algorithms
  - Single precision only
- Split high-level configuration (python) from core algorithms (CUDA/C++)
- Execute on multiple compute architectures
  - NVIDA/CUDA as the baseline
- Simple event model



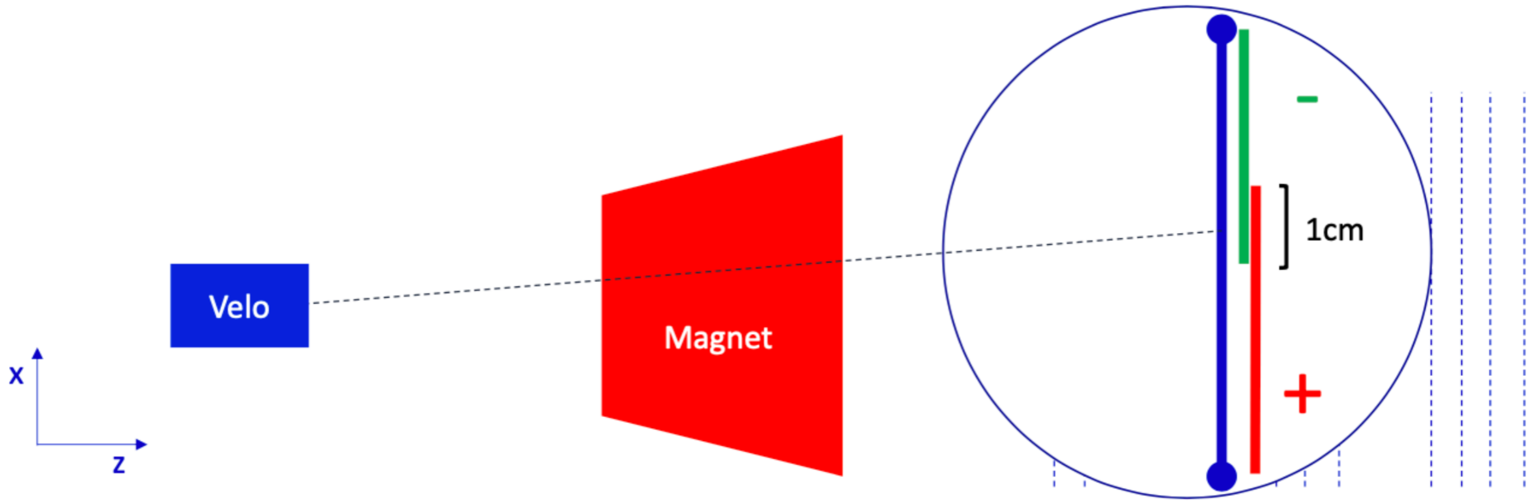
- Due to unavailable UT in 2022 and 2023 two independent tracking approaches were used in HLT1 together

1) Forward tracking without UT

- VELO track is extrapolated as a straight line
- Two search windows in SciFi (based on charge)
- Assumed: p > 5 GeV and p_T > 1 GeV



LHCb-FIGURE-2023-007

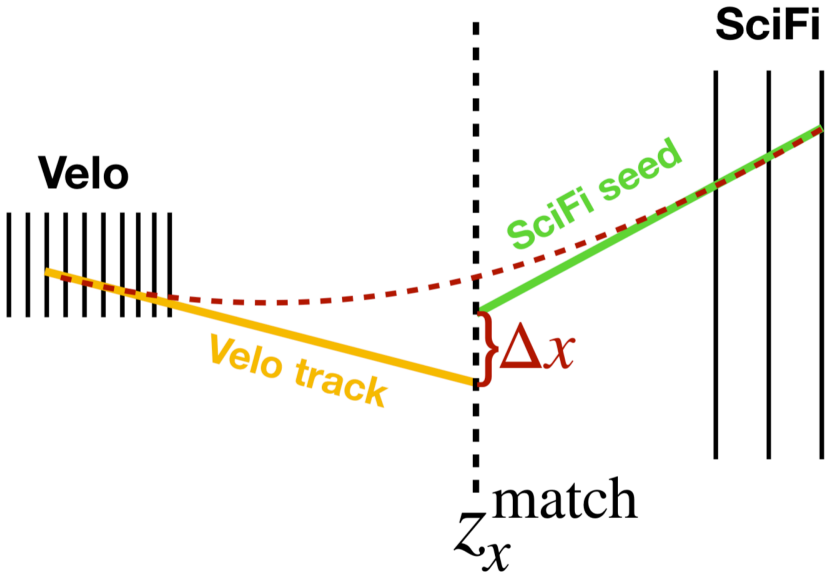


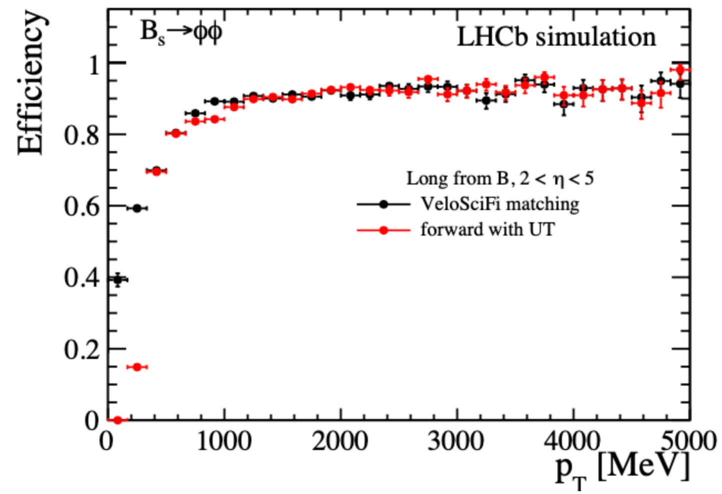
HLT1: Tracking overview

- Due to unavailable UT in 2022 and 2023 two independent tracking approaches were used in HLT1 together

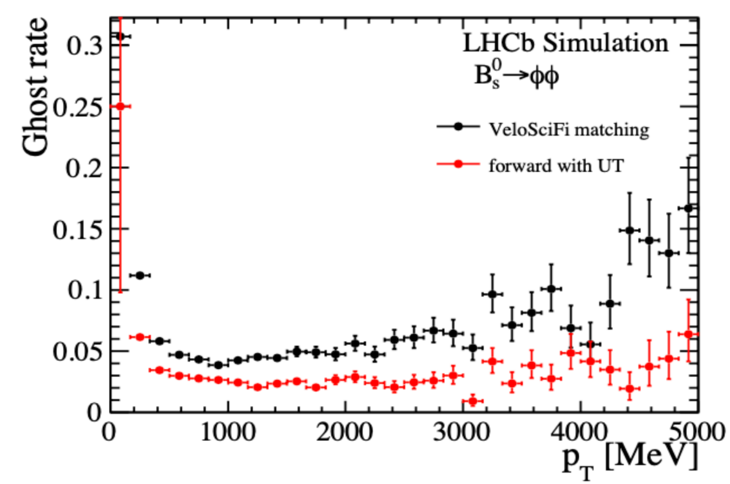
2) Seeding and Matching

- Standalone reconstruction of SciFi tracks
- Matching SciFi tracks to VELO seeds
- Efficient for a low momentum tracks



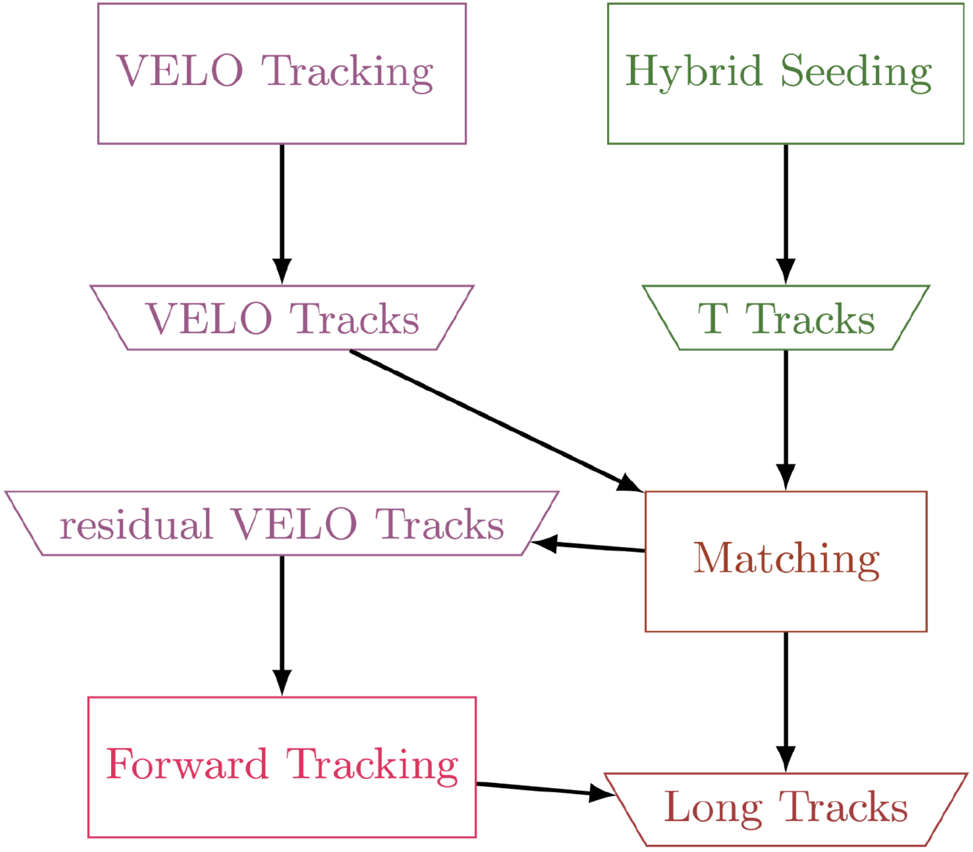


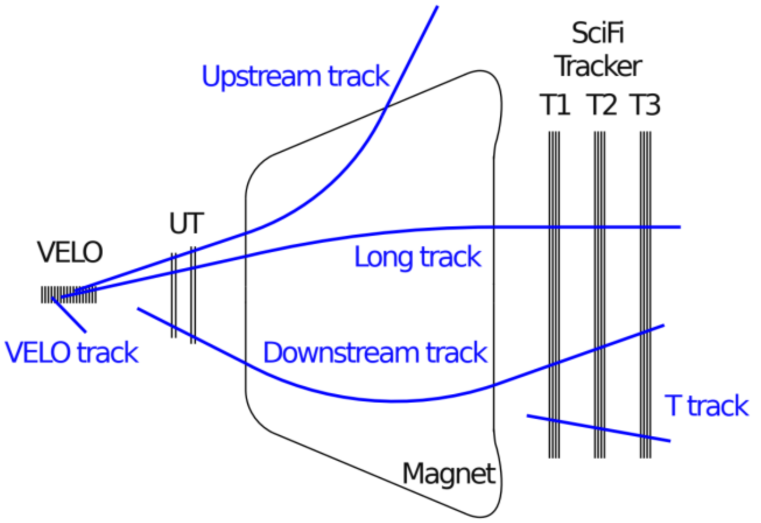
LHCb-FIGURE-2022-010



HLT1: Nominal HLT1 sequence

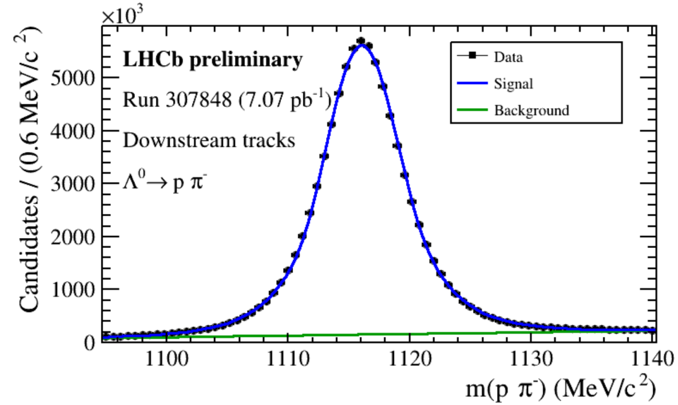
- Nominal HLT1 tracking sequence combines both forward tracking and Seeding and Matching
- Originally developed for HLT2, lately found efficient also for HLT1





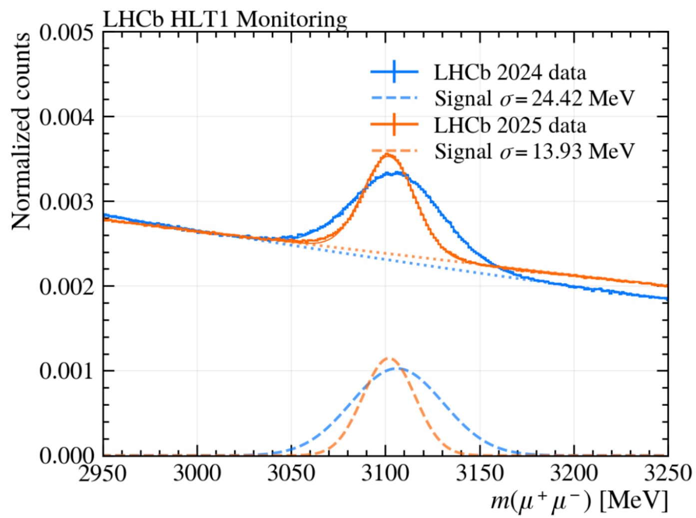
HLT1: developments beyond TDR plans

- HLT1 practically limited only by available computing power and limitations of sub-detectors hardware
  - Trigger and data rate not anymore a bottleneck for data-taking



LHCb-FIGURE-2024-035

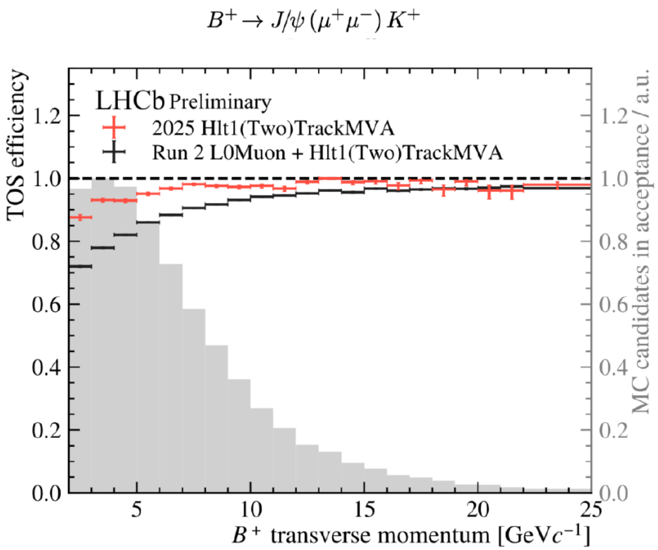
- Downstream tracking at HLT1
  - Long-lived particles decaying outside of VELO
  - Dedicated algorithms for downstream tracking directly in HLT1
  - Possible thanks to 3rd GPU card per node installed
  - Extending physics coverage for flavour physics and BSM searches

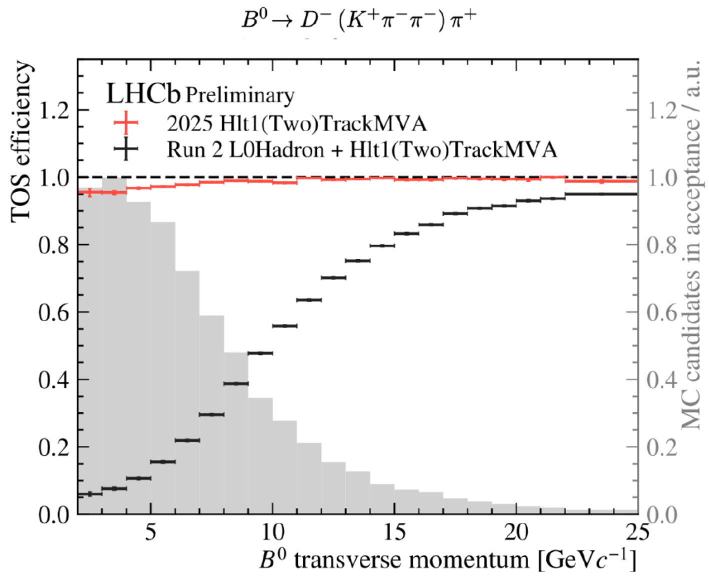


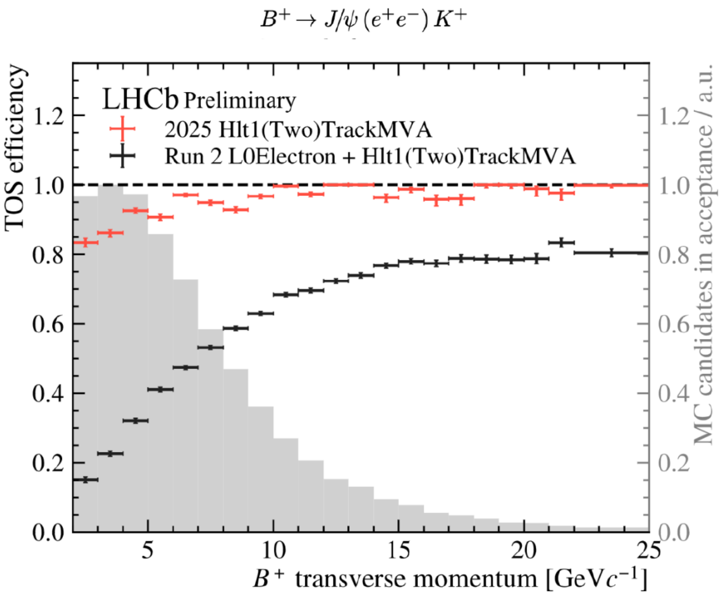
- Kalman filter in HLT1
  - Originally only parametrised Kalman filter applied for VELO tracks
  - Kalman filter with parameterised scattering errors and transport through the magnetic field
  - Deployed from the beginning of 2025
  - Better resolution in HLT1 allows higher selection precisions and decrease of output bandwidth

HLT1: performance in 2025

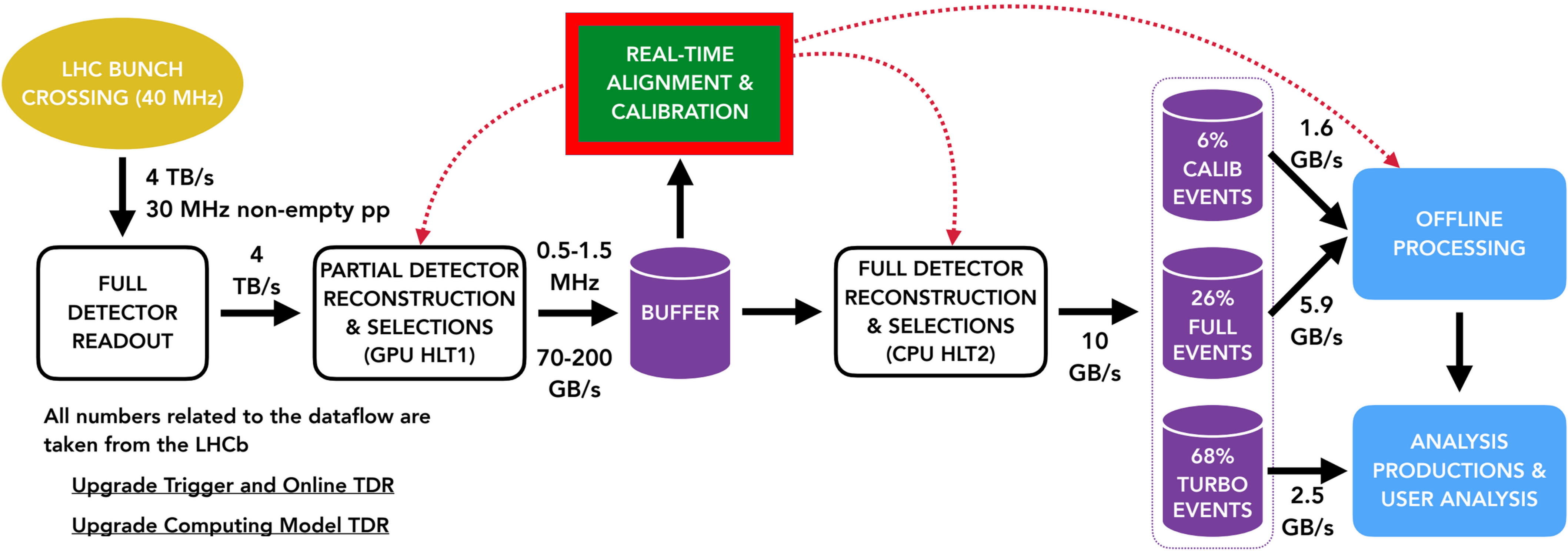
- Efficiency of HLT1 reconstruction and selection on selected b-physics channels
- Clear success of the selected approach







LHCb trigger design: Alignment and calibration



All numbers related to the dataflow are taken from the LHCb

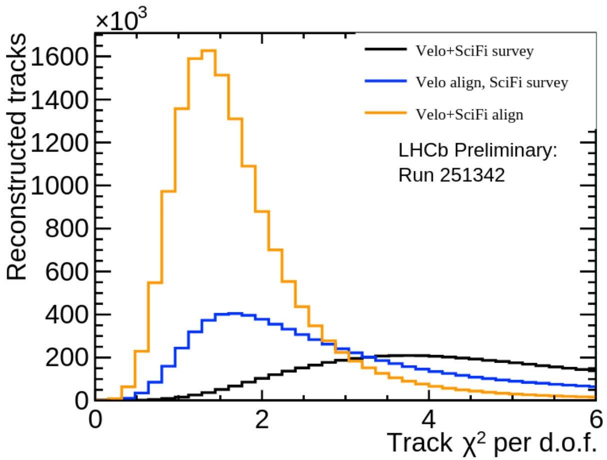
Upgrade Trigger and Online TDR

Upgrade Computing Model TDR

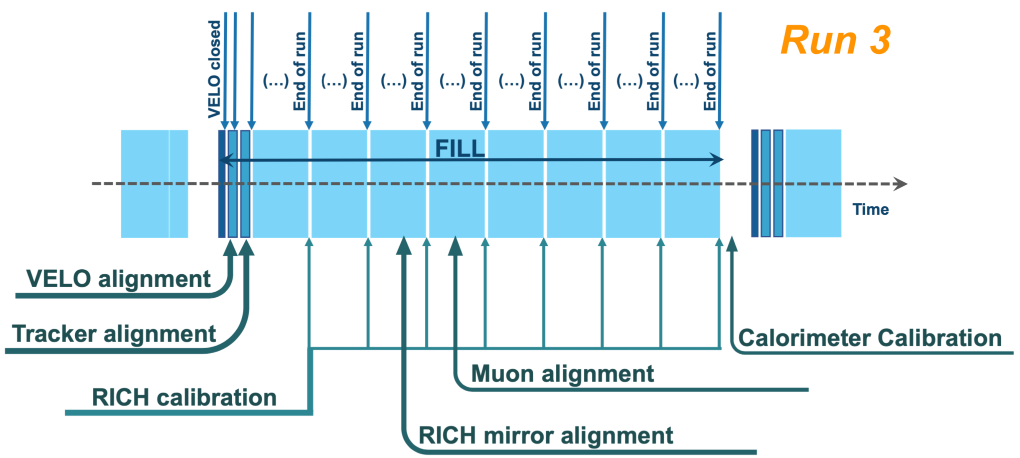
LHCb-FIGURE-2020-016

Alignment and calibration

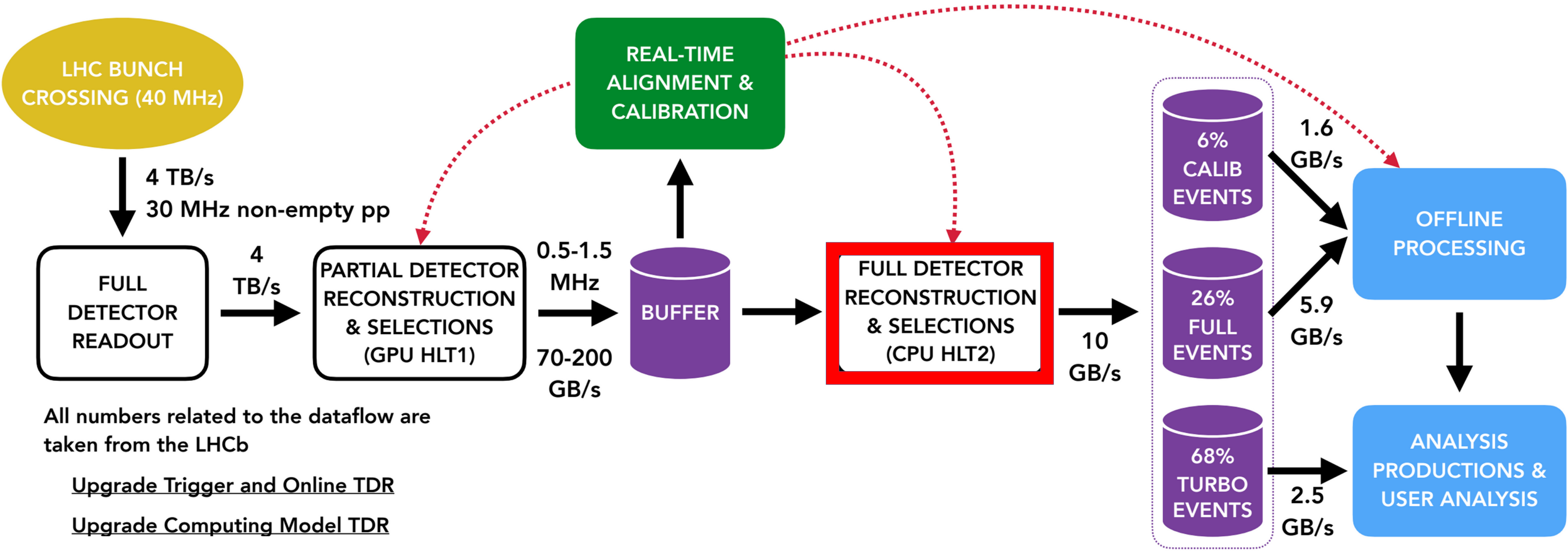
- Well aligned and calibrated data needed before running full reconstruction and selection in HLT2
  - Online alignment and calibration pioneered in Run 2, baseline in Run 3
  - Procedure fully automated by 2025
- Data collection for alignment based on dedicated HLT1 selection:
  - JpsiToMupMum, D0ToKpi, Z0toMupMum, prompt pi0, ...
- Alignment parameters are also propagated to HLT1
- LHCb distinguish two processes:
  - Alignment (VELO, RICH mirrors, UT, SciFi, Muon) and Calibration (RICH, ECAL, HCAL (also offline part))



LHCb-FIGURE-2022-018



LHCb trigger design: HLT2



All numbers related to the dataflow are taken from the LHCb

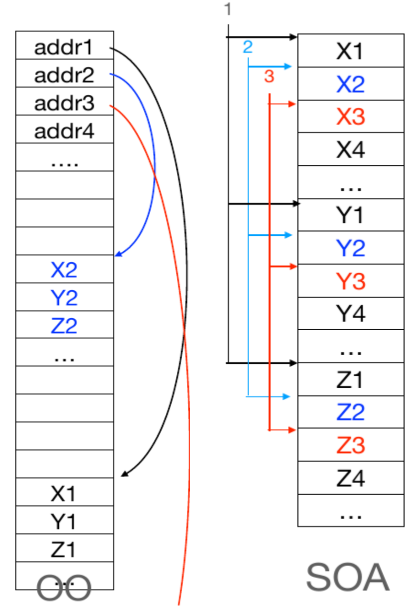
Upgrade Trigger and Online TDR

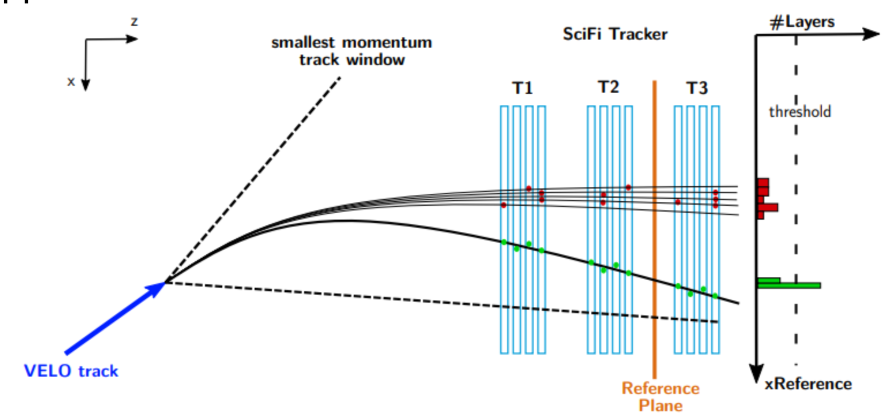
Upgrade Computing Model TDR

LHCb-FIGURE-2020-016

HLT2: Vectorisation and SoA data structure

- Underlying data structure directly influences possible routes of the code development
- For a proper vectorisation it is beneficial to switch from OO to SoA representation
  - OO: Object oriented 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
- However, SoA-based computing differs significantly in how data are accessed
  - Only a slices of SoA-collection are accessed, no objects (in OO terms)
- SoA relates well to SIMD (single instruction multiple data) approach
- Used in Forward tracking at HLT2:
  - Several thresholds can be scanned in parallel
  - 8 single precision floats/integers in parallel (AVX2)
  - Throughput of Forward Tracking increased up to 60%
- Ongoing studies of SoA structure usage in event model

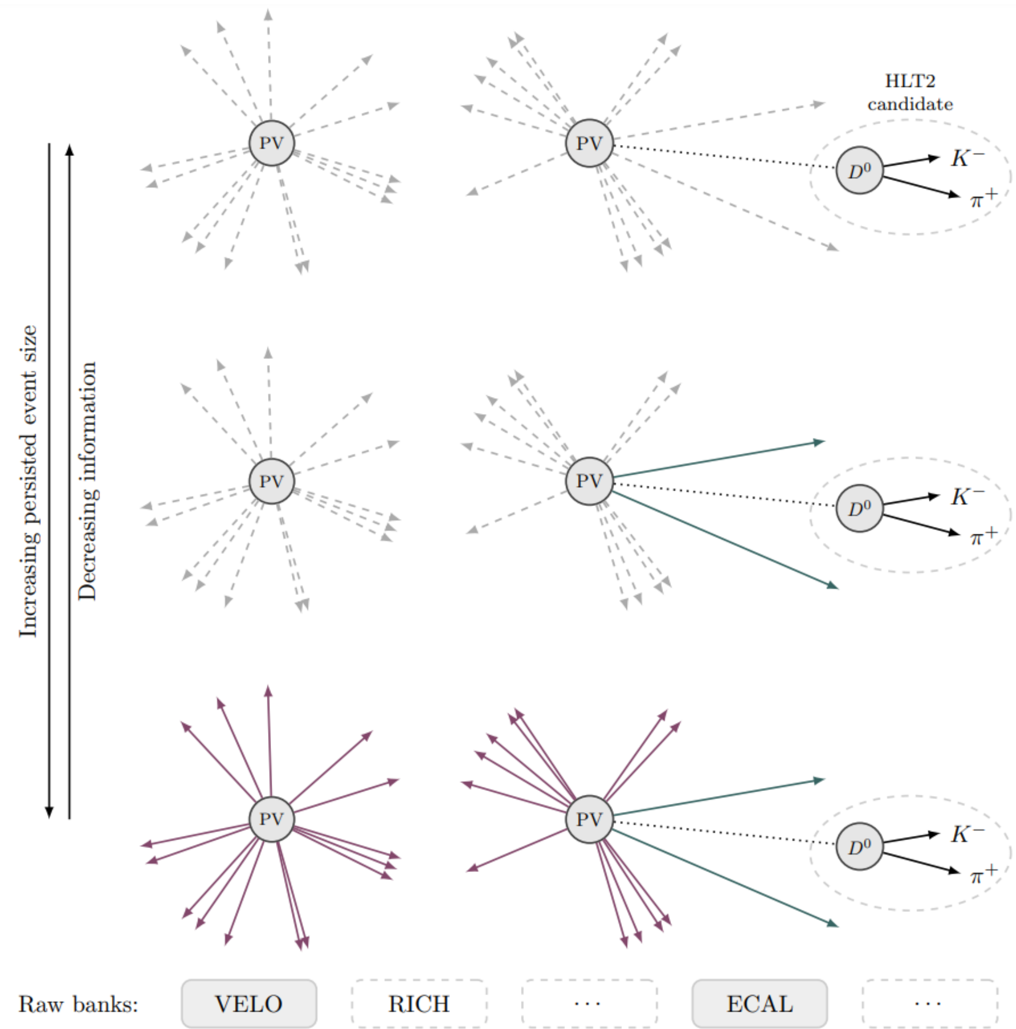




HLT2: Turbo model

- Bandwidth [GB/s]
  ≈ Accept Rate [kHz] ╳ Event size [kB]
- Instead of saving of full event, only information needed for a physics analysis can be stored
- Flexible persistence settings
  - Turbo: Save only information related to signal candidate
  - Selective persistency: Save additional objects
  - Full persistency
  - Option to save raw banks as well
- Baseline of Run 3 computing model
  - <u>70% of rate while only 20% of bandwidth</u> (to tape)
- Allows analysis directly on HLT2 output
  - Reducing demand on offline resources

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



JINST 14 (2019) 04, P04006

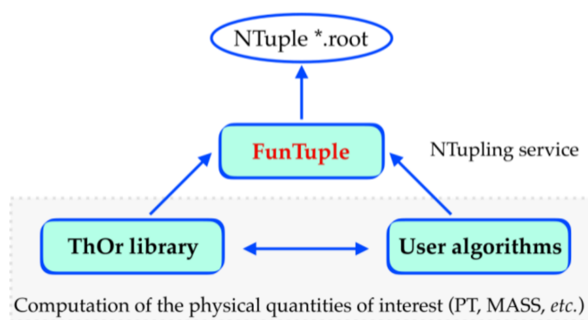
HLT2: selection framework for online and offline

- HLT2 timing: reconstruction + PID 60%, selection 30%, other functionalities ~ 10%
- Selection lines written in python using Throughput Oriented (ThOr) functors then translated into C++ code
- Functors are composable allowing a simple chaining of basic functors
  - e.g. X @ POSITION @ VERTEX ⇒ Particle.vertex().position().x()
- Using functor cache instead of just-in-time (JIT) compilation
  - Functors that are defined in python during build ⇒ compiled into a cache
  - Recompiling cache for each offline job would be too demanding
  - Compiled cache is available on CVMFS for offline computing usage, mainly used for MC production
- Code for HLT2 lines and offline tupling is using same framework and same syntax / logic (ThOr functors)

```python
@configurable
def make_mass_constrained_jpsi2mumu(name='MassConstrJpsi2MuMuMaker_{hash}',
                                   jpsi_maker=make_dimuon_base,
                                   nominal_mass=masses['J/psi(1S)'],
                                   pid_mu=0,
                                   probnn_mu=None,
                                   pt_mu=0.5 * GeV,
                                   admass=250. * MeV,
                                   adoca_chi2=20,
                                   vchi2=16):
    """Make the Jpsi, starting from dimuons"""

    # get the dimuons with basic cuts (only vertexing)
    # note that the make_dimuon_base combiner uses vertexChi2/ndof < 25,
    # which is looser than the vertexChi2 < 16 required here
    dimuons = jpsi_maker(
        pt=pt_mu, pid_mu=pid_mu, probnn_mu=probnn_mu, adoca_chi2=adoca_chi2)

    code = F.require_all(
        in_range(nominal_mass - admass, F.MASS, nominal_mass + admass),
        F.CHI2 < vchi2)
    return ParticleFilter(dimuons, F.FILTER(code))
```




```python
"""
Tuple observables relatex to Jpsi -> mu+ mu-
"""
# FunTuple make fields to tuple
fields_jpsi = {
    'Jpsi': "[J/psi(1S) -> mu+ mu-]CC",
    'mup' : "[J/psi(1S) -> ^mu+ mu-]CC",
    'mum' : "[J/psi(1S) -> mu+ ^mu-]CC"
}

# FunTuple make collection of functors for Jpsi
variables_jpsi = FunctorCollection({
    'THOR_P': F.P
    'THOR_PT': F.PT,
    'THOR_mup_PT': F.CHILD(1, F.PT),
    'THOR_mum_PT': F.CHILD(2, F.PT)
})
```

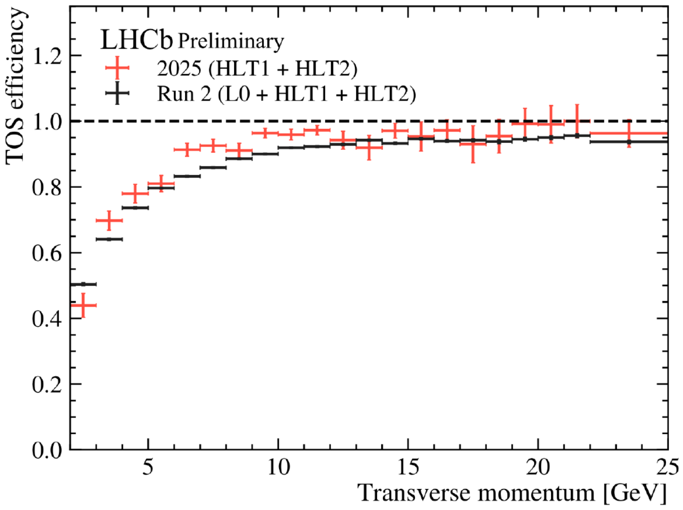
particles

variables

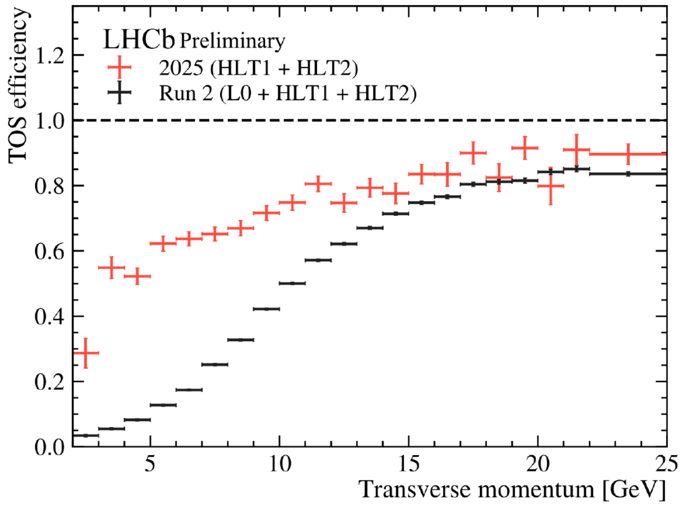
HLT2: performance in 2025

- Efficiency of HLT1 reconstruction and selection on selected b-physics channels
- Stable performance for dimuon modes, significant gain for hadronic and electron modes

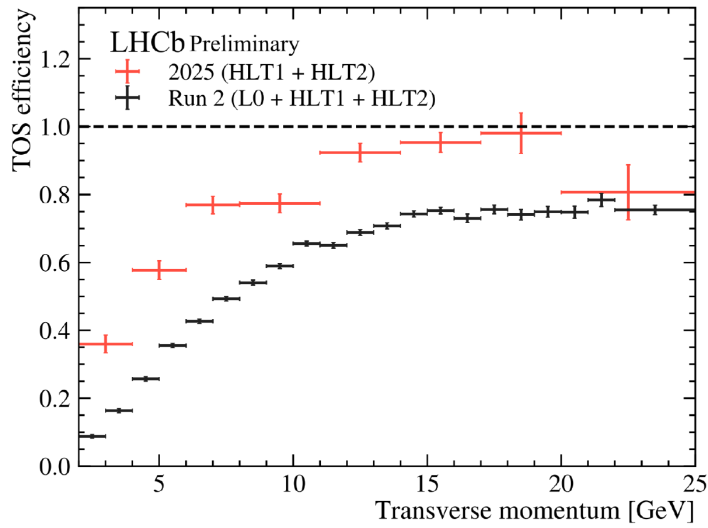
$B^+ \to J/\psi\,(\mu^+\mu^-)\,K^+$



$B^0 \to D^-\,(K^+\pi^-\pi^-)\,\pi^+$



$B^+ \to J/\psi\,(e^+e^-)\,K^+$

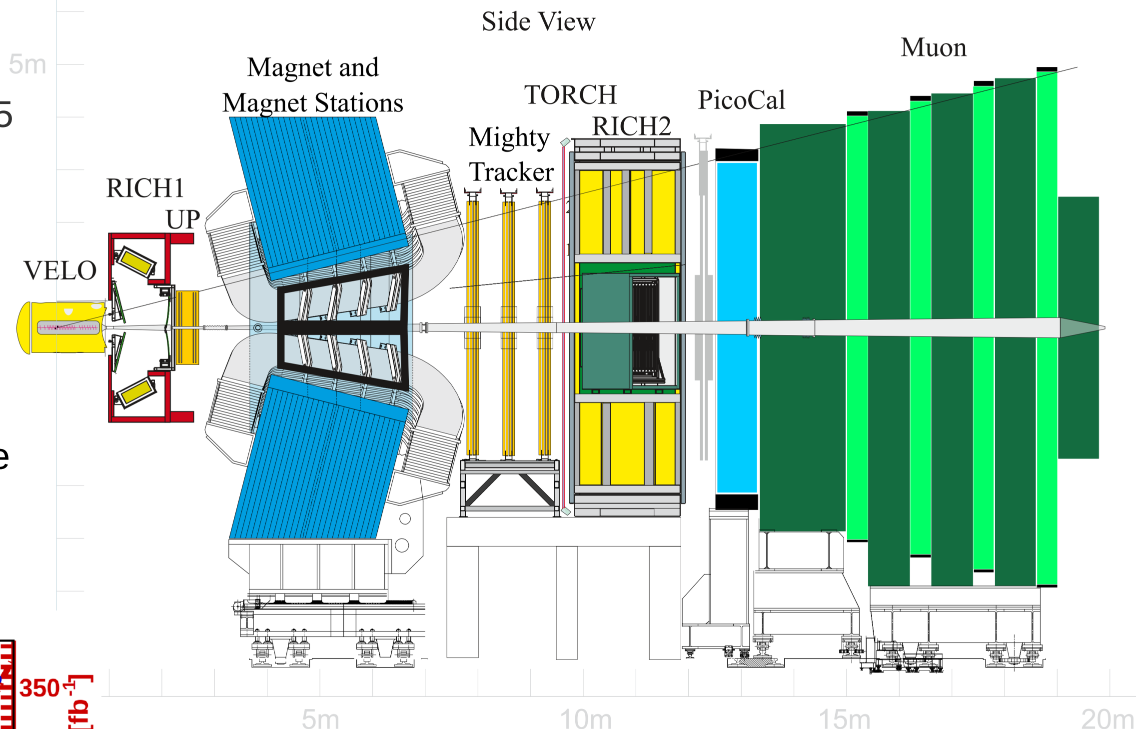


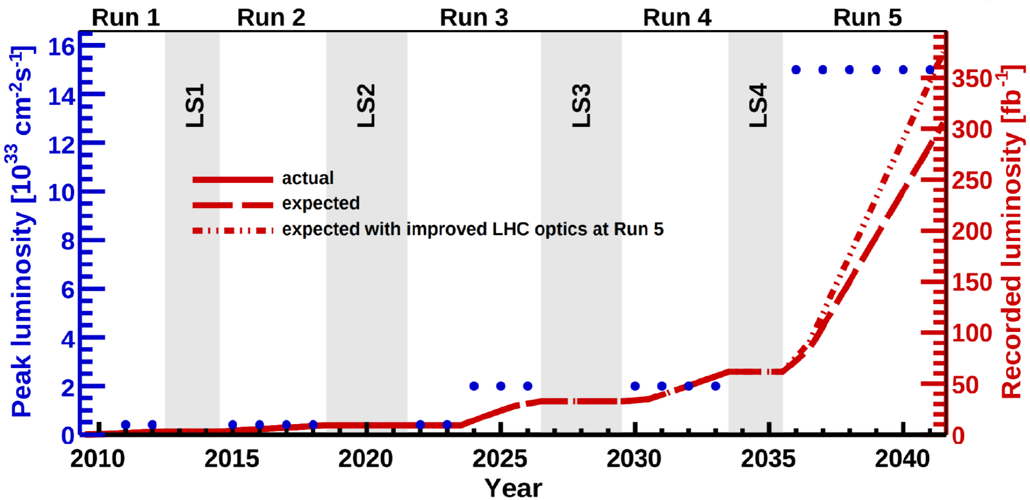
LHCb Upgrade 2
and
Computing advancements

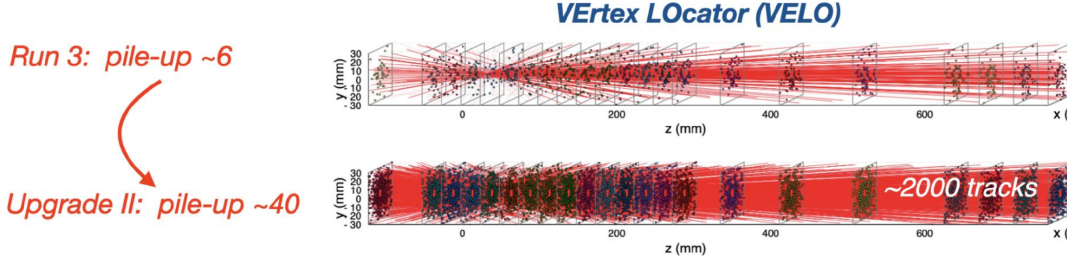
![LHCb logo]

LHCb Upgrade II

- Proposed for Run 5 (2036-2041)
  - Plans to record ~ 300fb^{-1} by end of Run 5
  - Luminosity: 1.5x10^{34} cm^{-2}s^{-1}; pileup: 40
  - New and improved detectors
  - Higher granularity
  - Add timing measurement (10 to 50 ps)
- Keeping same performance as LHCb Upgrade I under harsher conditions



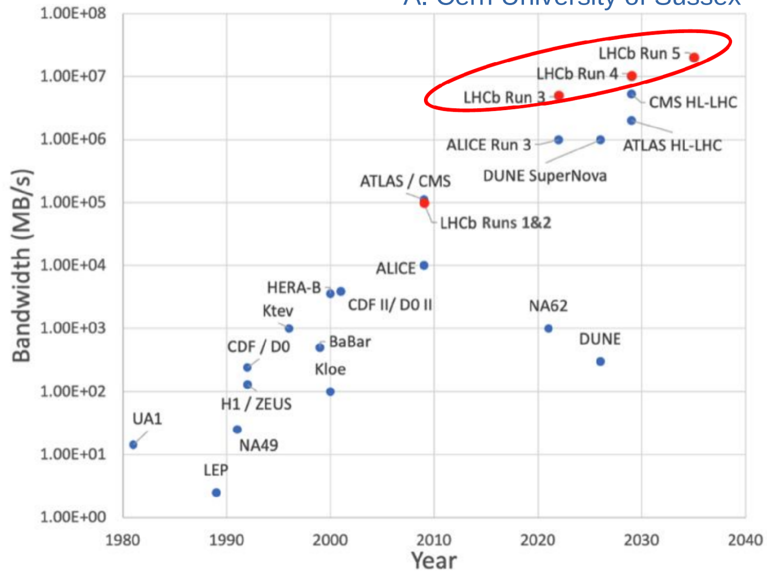




LHCb Upgrade II Trigger system

- Expected data rate is ~ 25 TB/s
- **The highest expected value at HEP**
- HLT2 and Alignment steps have to run on GPUs
  - Keeping CPU implementation would be too expensive
- LHCb Upgrade I HLT1 shown that reconstruction and basic selection works well on GPUs
- HLT2 is more complex than HLT1
  - Full reconstruction of all detectors
  - Full PID evaluation
  - Selection framework with more than 3000 selection lines
    - Majority of line will be based on their own MVA selection
    - The selection part of the trigger is not easily vectorizable!
  - Output persistency and possibly streaming
- High performance while selection framework has to be accessible to all physicists ⇒ domain-specific language required

A. Cerri University of Sussex



HLT1

- O(100) algorithms
- O(10) developers
- 125k LOC
- On GPU

HLT2:

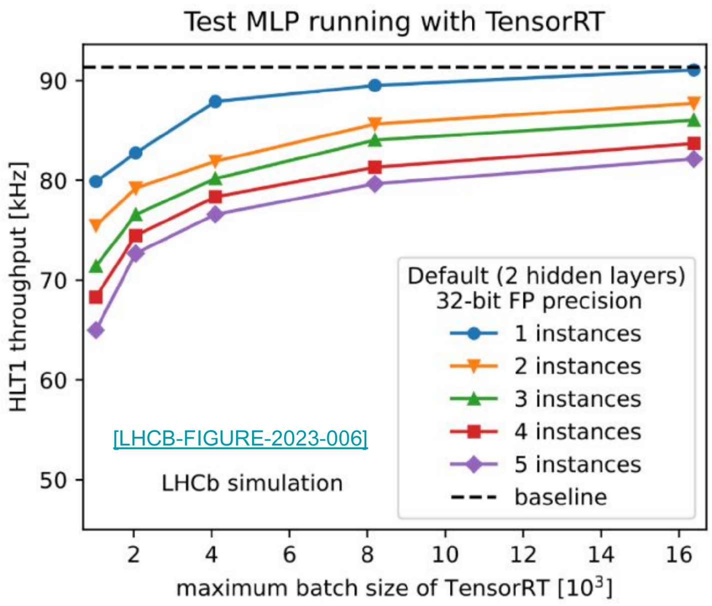
- O(2000) algorithms
- O(100) developers
- 2M LOC
- On CPU

Future computing architectures

B. Couturier

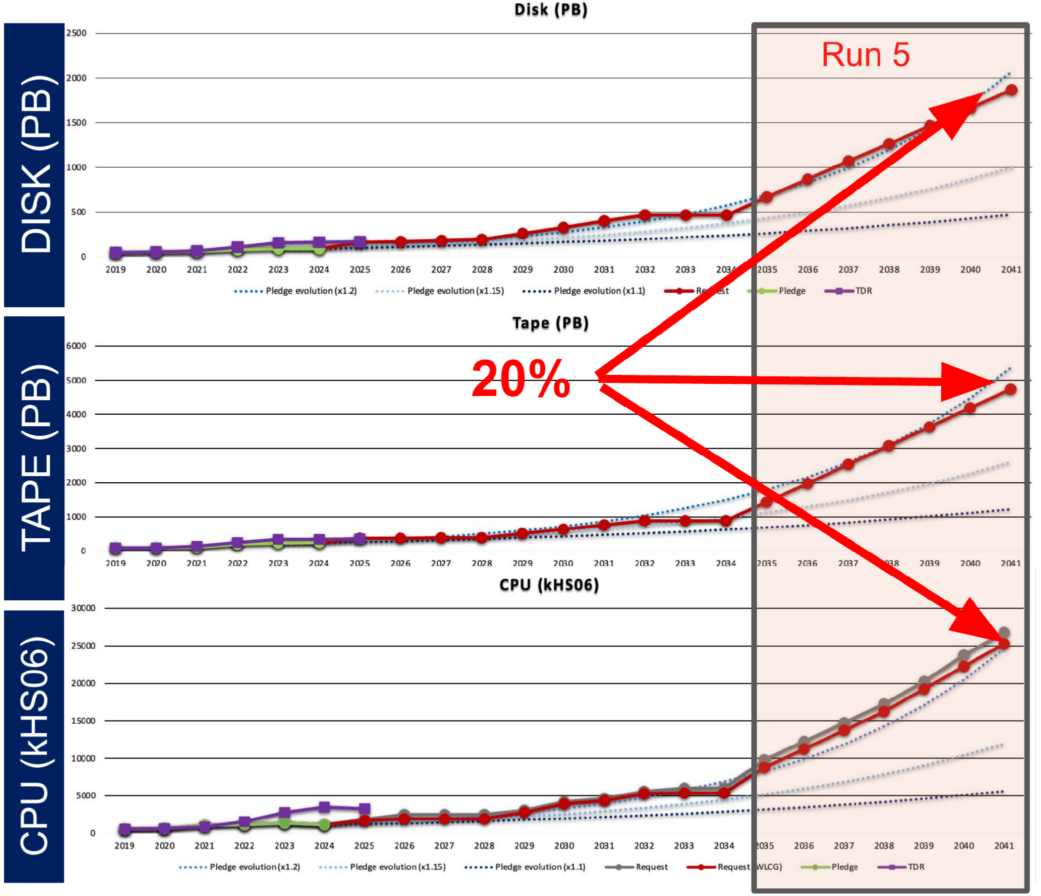
- Current AI/LLMs boom is pushing computing architectures out of HEP needs
  - Single precision versus floating point operations
  - Partially can be compensated by computing methods
- Various non-x86 architectures vastly outperforming in dedicated tasks
  - FPGAs offer a good ratio of computing power but flexibility is limited
  - Investigate new architectures as TPUs, NPUs, IPUs, … ?
  - Even more heterogeneous trigger system?
- ML/AI algorithms are evolving fast, already integral part of online computing with only extended presence
  - Simple deployment and fast inference crucial for a large-scale deployments

Area	Traditional HPC	LLM-Age HPC
Workload	Simulations	AI training/inference
Hardware	CPUs, GPUs	GPUs with reduced precision, NPUs?, AI fabrics
Scale	Large clusters	Exascale, hyperscale AI
Users	Researchers, scientists	AI labs, cloud, enterprise



Offline computing requirements

- LHCb Upgrade II is not challenge only for online/trigger computing
- Significant requirements for offline computing
- Naive extrapolation by factor 5 to the end of Run 5
  - Disk: 1.5 EB; Tape: 4.8 EB
- In term of CPUs:
  - CPU: 25 MHS06 ~2.5M cores
- Compare with the ATLAS baseline:
  - Storage: 8+ EB; CPU: 80 MHS06 ~ 8M cores
- Jobs by analysts are not listed
  - Significant challenges for the whole analysis infrastructure
- Offline resources are not scaling up as necessary
  - Do not forget about disk space for Simulation
- Think about what (signal/background) and how (format) is being saved and how often and why to be accessed (offline reprocessing)



Final remarks

- LHCb Upgrade I brought HEP 1st full-software trigger to reality
  - Highly flexible and efficient systems well working during Run 3
  - Flexibility allowed successful data taking even when some hardware systems were not ready

- Unification of online and offline software allowed broader base of developers and simpler maintenance

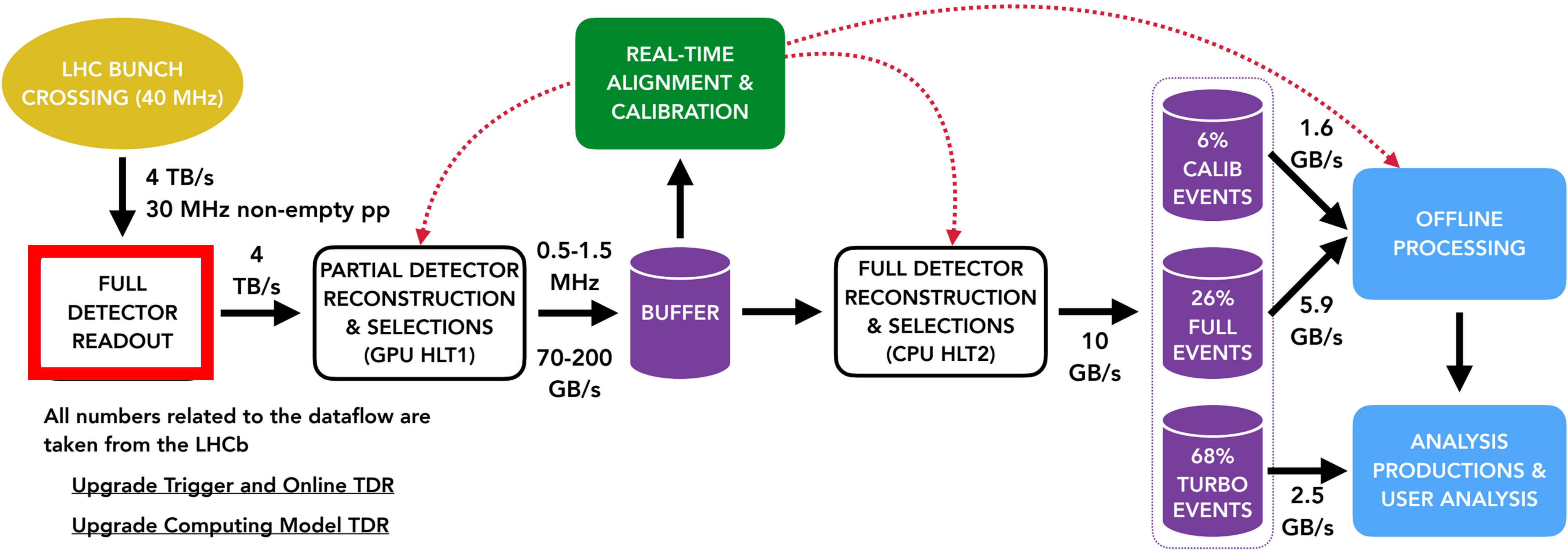
- Offline computing resources are becoming a limitation, computing model is critical part of any experiment
  - What steps should be performed online and offline?
  - Keeping raw events on tape while only pre-defined processing campaigns on disk?

- LHCb Upgrade II presents the most ambitious target for trigger and online computing in envisioned HEP future
  - The collaboration has to remain at the edge of computing to deliver outlined goals
  - Many challenges and opportunities at the same time

Thank you for the attention

Spare slides

LHCb trigger design: DAQ and Online



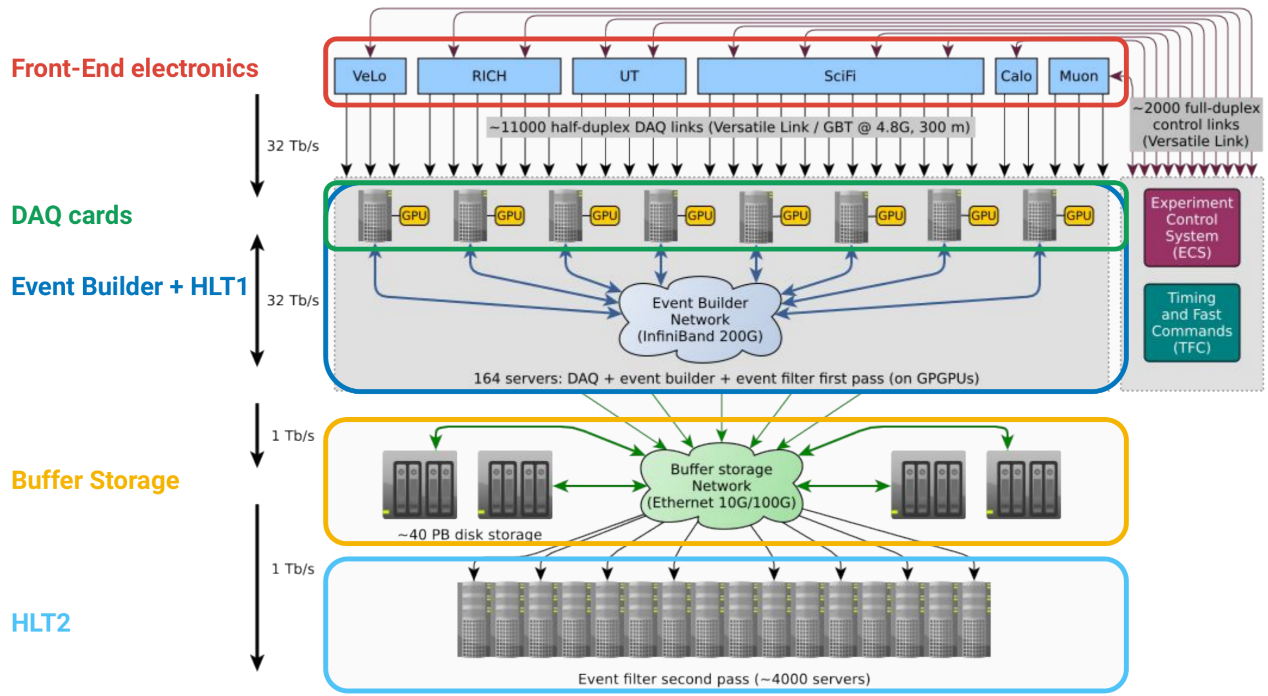
All numbers related to the dataflow are taken from the LHCb

Upgrade Trigger and Online TDR

Upgrade Computing Model TDR

LHCb-FIGURE-2020-016

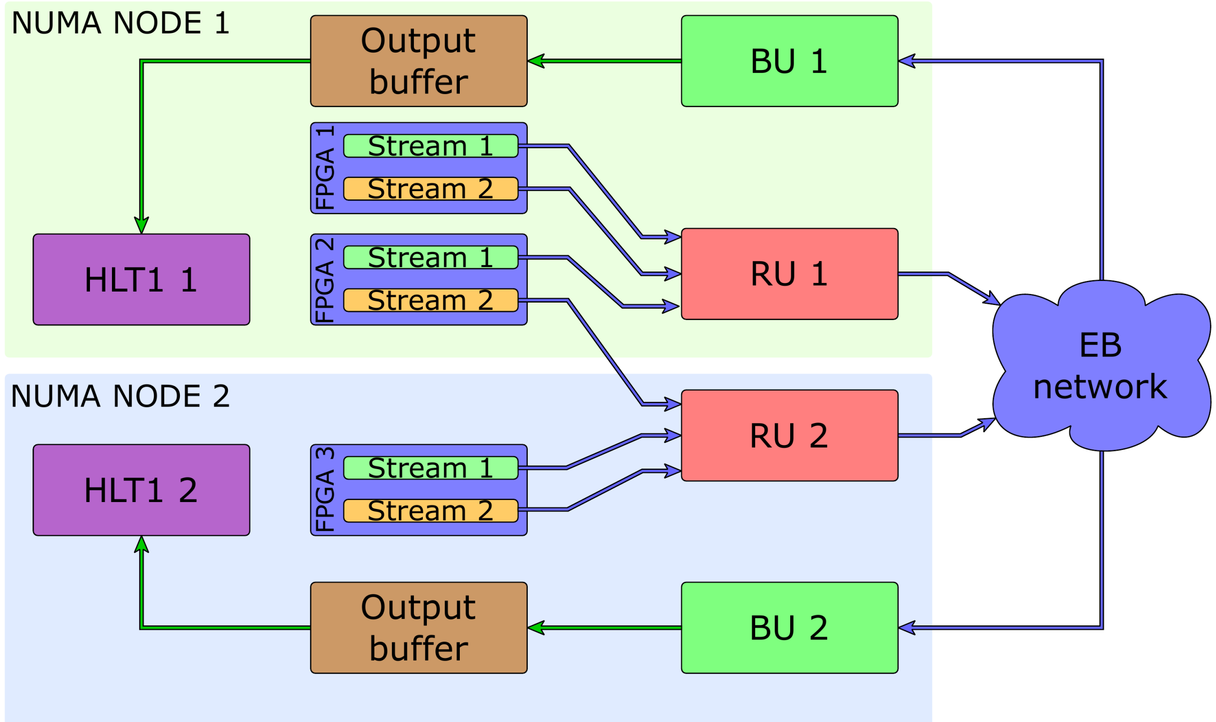
Online architecture: overview



F. Pisani, CHEP2023

- Full detector readout performed by O(500) custom PCIe40 FPGAs
- Event Builder (EB) farm consists of 173 servers with 3 free PCIe slots per server
- HLT1 stage implemented directly within the EB farm as HLT1 approach is inherently parallelizable
  - 3 GPUs (NVIDIA RTX A5000) per server -> around 500 GPUs installed in total
- Allows a lighter network post EB

Online architecture: data flow



- Custom-made modular architecture built in C++
- Readout unit (RU): reads the data from DAQ card and send it to the EB network
- Builder unit (BU): reads the data from the EB network and writes the built data into the HLT1 input buffer
- The scheduling synchronization is achieved by using an in-band data barrier
- Buffer-isolated critical section to minimise slowdowns and deadtime

Custom FPGA: PCIe40

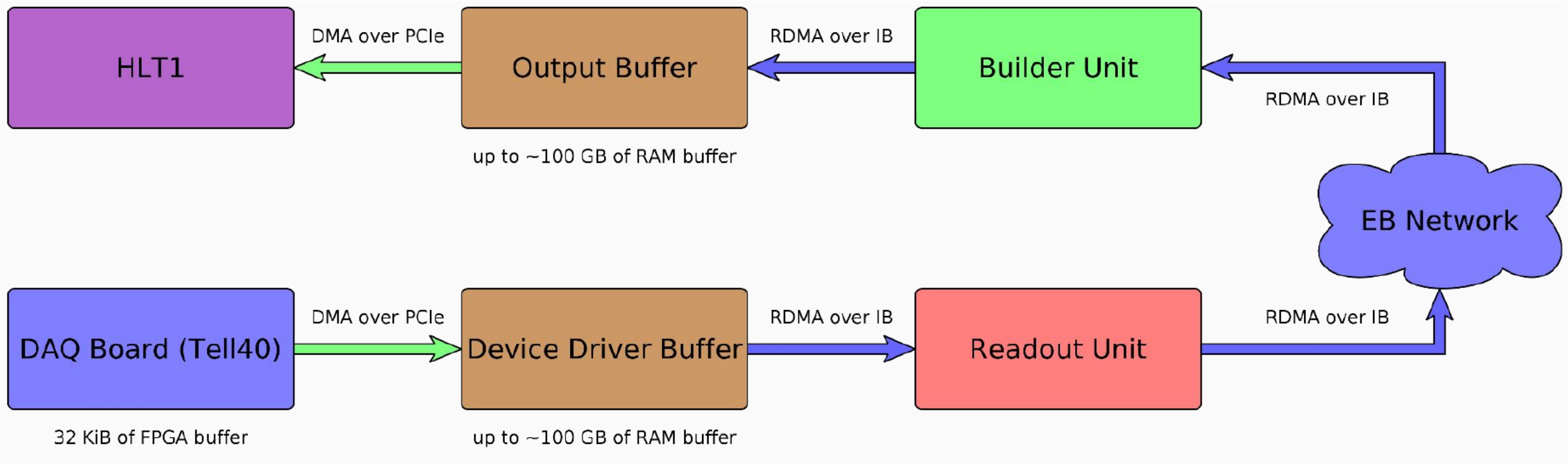
- Custom built card based on Intel Arria10
- 48x10G capable transceiver on 8xMPO for up to 48 full-duplex Versatile links
- 2 dedicated 10G SFP+ for timing distribution
- 2x8 Gen3 PCIe
- One card can serve in different roles based on FW:
  - Readout Supervisor (SODIN)
  - Interface board (SOL40)
  - DAQ card (TELL40)

Data output per sub-detector

- Multiple Event Packet (MEP): HLT1 input format combining several events
  - 1 MEP contains 30 000 events

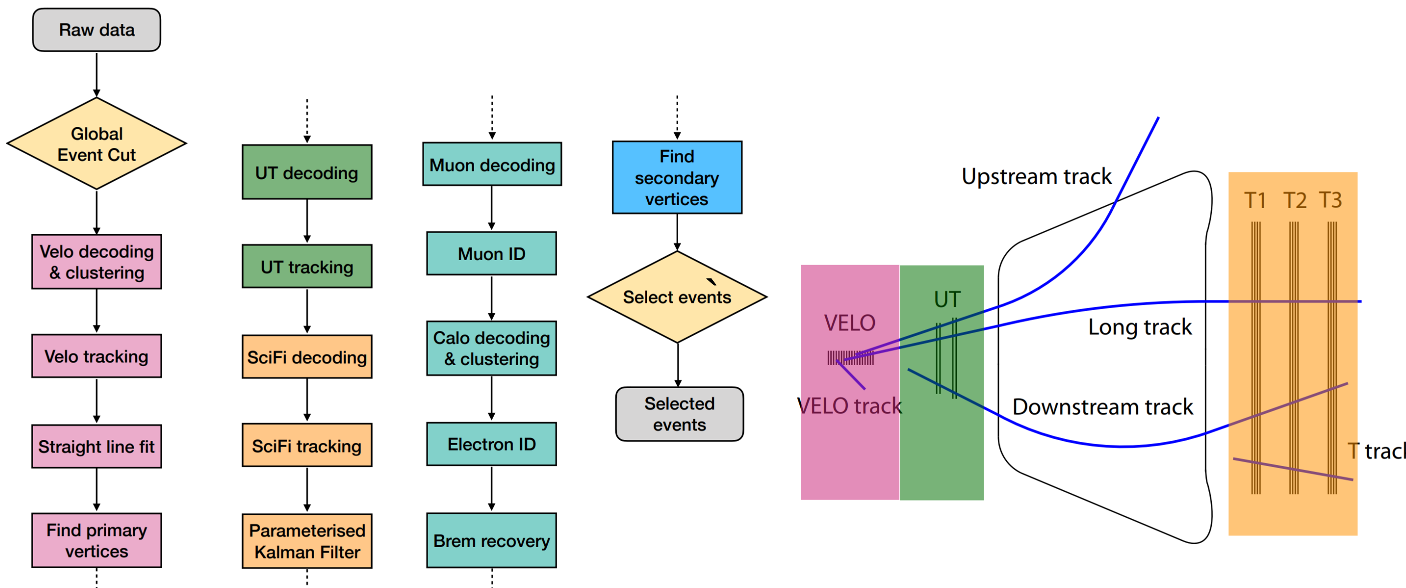
Sub-detector	fragment size [B]	#tell40 streams	event size [B]	event fraction	MEP size [GB]	MFP size [MB]	RU send size [MB]
Velo	156	104	16250	0.13	0.49	4.69	14.06
UT	100	200	20000	0.16	0.60	3.00	9.00
SCIFI	100	288	28800	0.23	0.86	3.00	9.00
Rich 1	166	132	22000	0.18	0.66	5.00	15.00
Rich 2	166	72	12000	0.10	0.36	5.00	15.00
Calo	156	104	16250	0.13	0.49	4.69	14.06
Muon	156	56	8750	0.07	0.26	4.69	14.06
Total	1000	956	124050	1	3.72	30.06	90.19

Online: latency and server flow



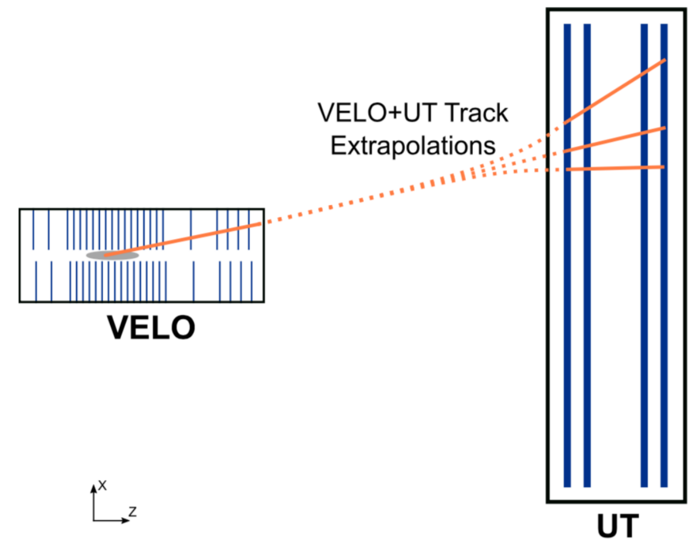
HLT1: overview

- The goal of HLT1 is to process of the LHCb raw data at 30 MHz and reduce rate by a factor of 30 (to 1 MHz)
- HLT1 is implemented in the form of Allen project [Comput. Soft. Big Science 4, 7 (2020)], using RTX A5000
  - Cross-architecture support: x86, CUDA/CUDACLANG (NVIDIA GPUs), HIP (AMD GPUs)
- Partial event reconstruction: vertexing, tracking, muon PID, simplified CALO information
- Rough selection based on O(50) trigger lines covering LHCb physics program
  - High/low pT muons, NN-based one-/two-track selection, detached lines, ...
- Required performance obtainable using O(200) GPUs, 346 RTX A5000 installed



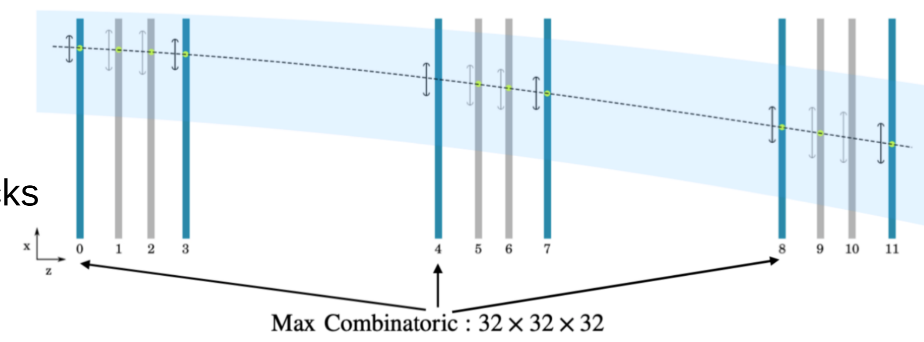
HLT1: Tracking per sub-detector

- Velo tracking [Journal of Computational Science, vol. 54, 2021]
  - 26 silicon pixel modules with $\sigma_{x,y} \sim 5\ \mu m$
  - Local paralleled clustering algorithm (Search by Triplet)
  - Tracks fitted with simple Kalman filter assuming straight line model



- UT tracking [IEEE Access, vol. 7, pp. 91612-91626, 2019]
  - 4 layers of silicon strips
  - Velo tracks extrapolated to UT taking into account B field
  - Parallelized trackless finding inside search window requiring at least 3 hits

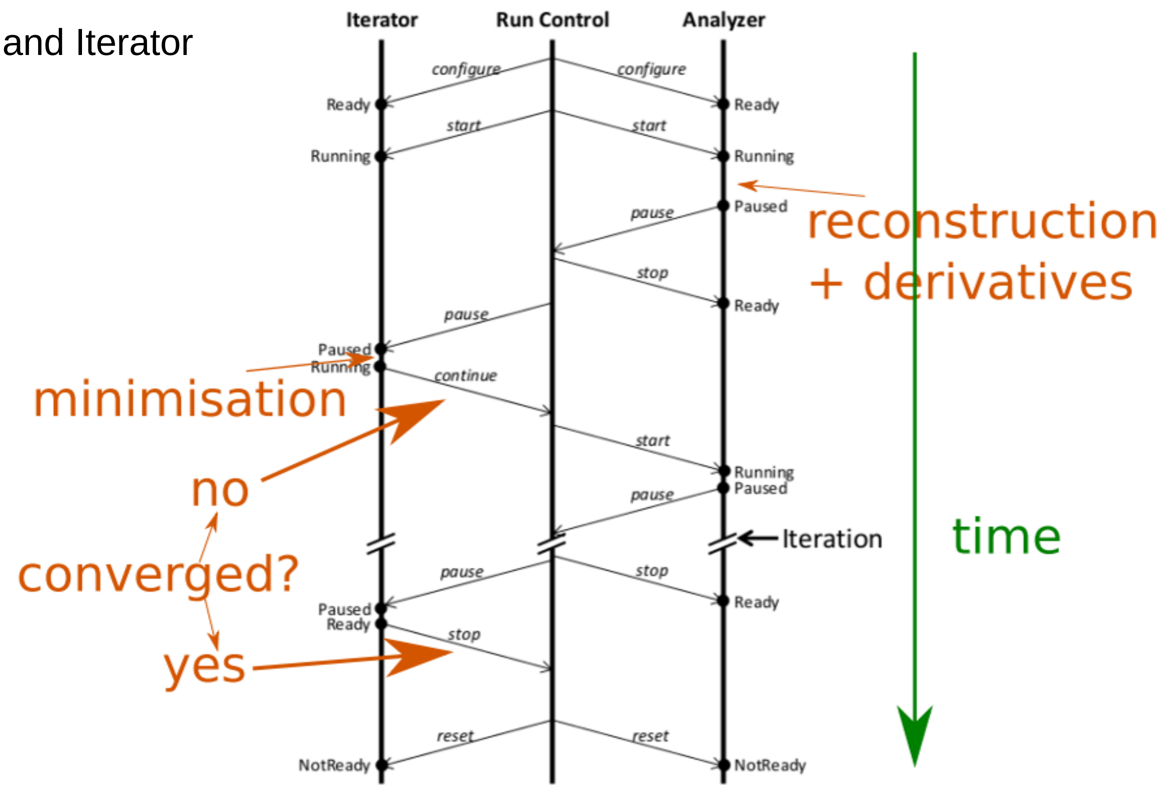
- SciFi tracking [Comput Softw Big Sci 4, 7 (2020)]
  - 3 stations with 4 layers of Scintillating Fibres
  - Velo-UT tracks extrapolated using parametrisation
  - Parallelized Forward algorithm to reconstruct long tracks
  - Search windows from Velo-UT momentum estimate
  - From triplets and extend to remaining layers



Alignment and calibration: Alignment

- Alignment and calibration procedure is running multi-threaded on roughly 160 CPU nodes
- Based on a set of HLT1 events selected by a dedicated HLT1 lines
- 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
- Iterator:
  - Collect derivatives/histograms
  - Obtain new constants
  - Convergence check

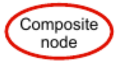


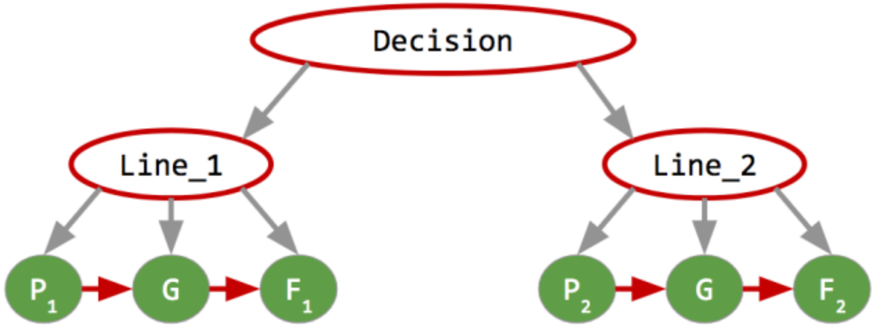
F. Reiss CTD 2022

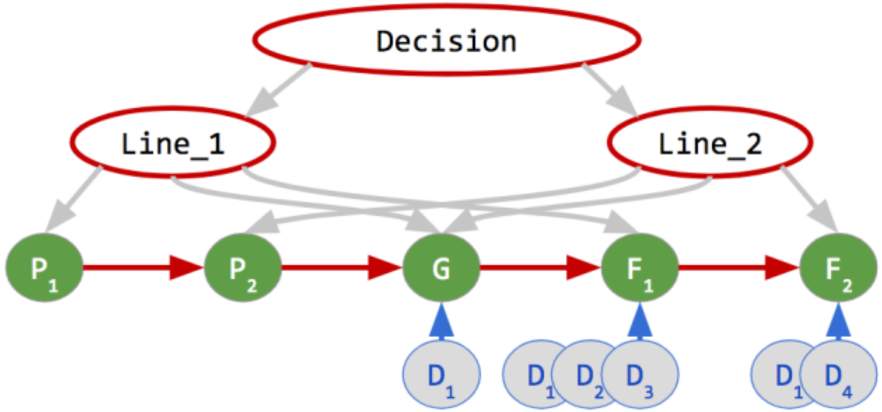
HLT2: Event scheduler

- Dealing with O(3000) 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
- Automatically resolve data dependencies by matching input / output
- Static graph with ordered nodes (respecting data constrains)
- Configured during initialization

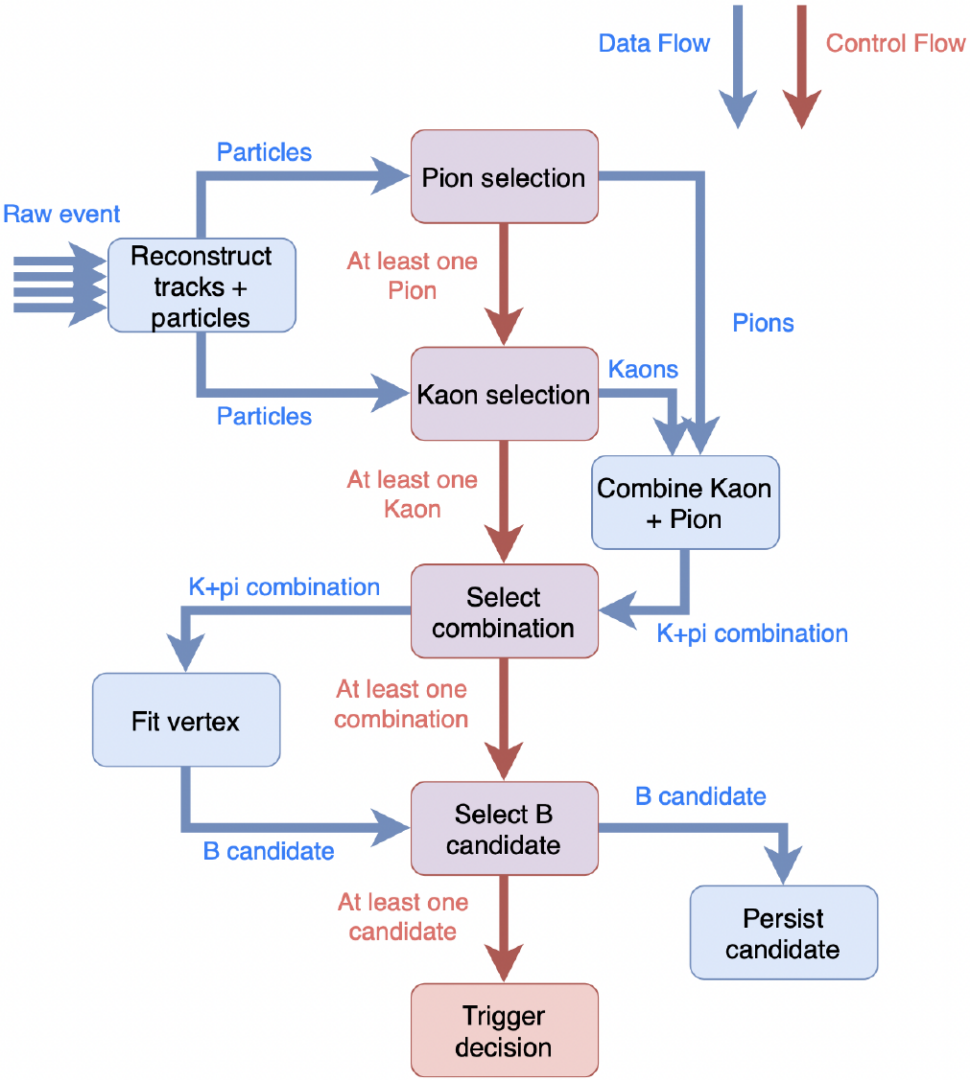
- Basic node: one algorithm with data dependencies 

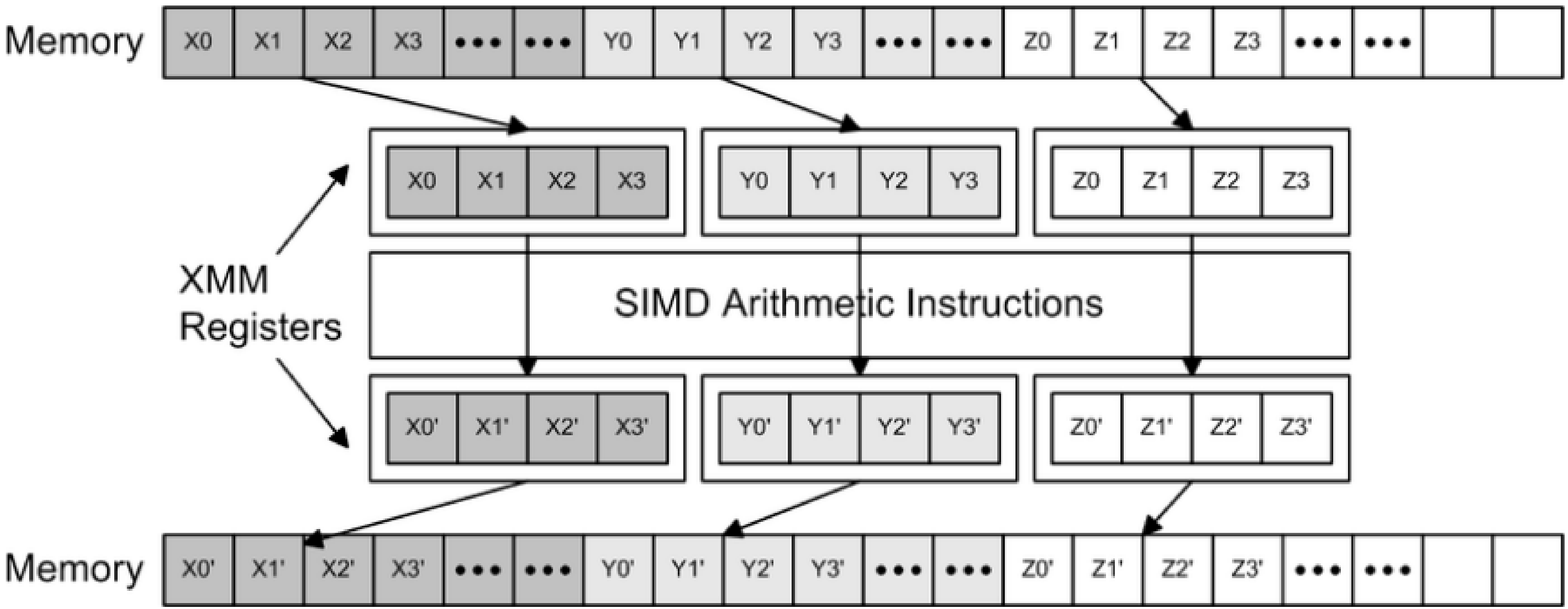
- Composite node: logic operation (AND, OR, NOT) 





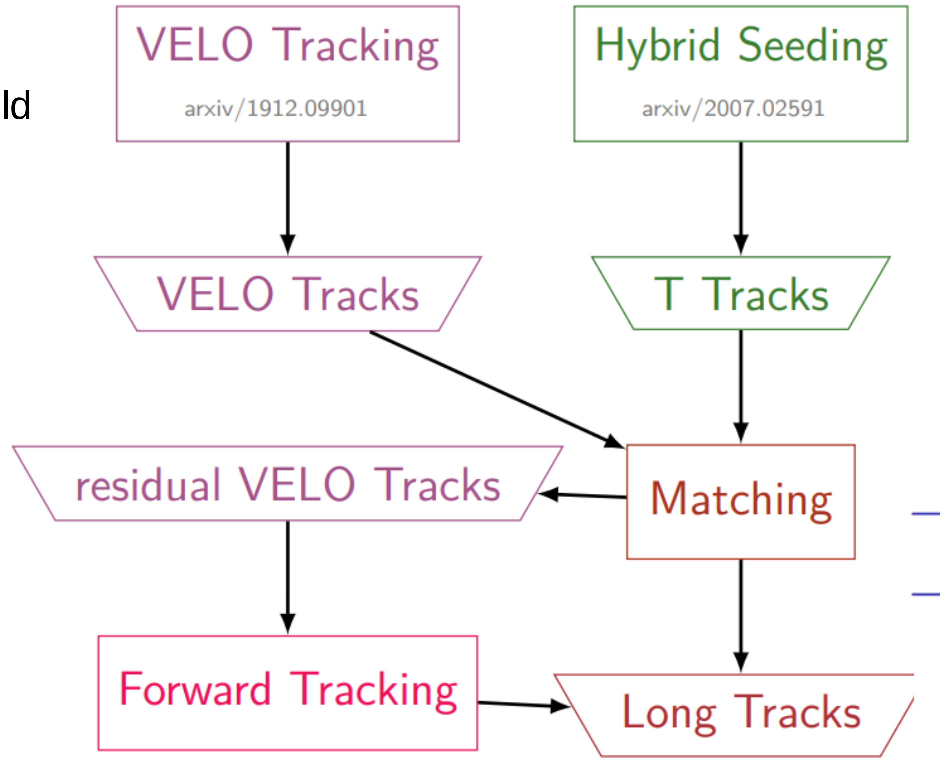
J. Phys.: Conf. Ser. 1525 012052


Data Flow
Control Flow
Particles
Raw event
Pion selection
Reconstruct tracks + particles
At least one Pion
Pions
Kaons
Kaon selection
Particles
At least one Kaon
Combine Kaon + Pion
K+pi combination
Select combination
K+pi combination
Fit vertex
At least one combination
B candidate
Select B candidate
B candidate
At least one candidate
Persist candidate
Trigger decision

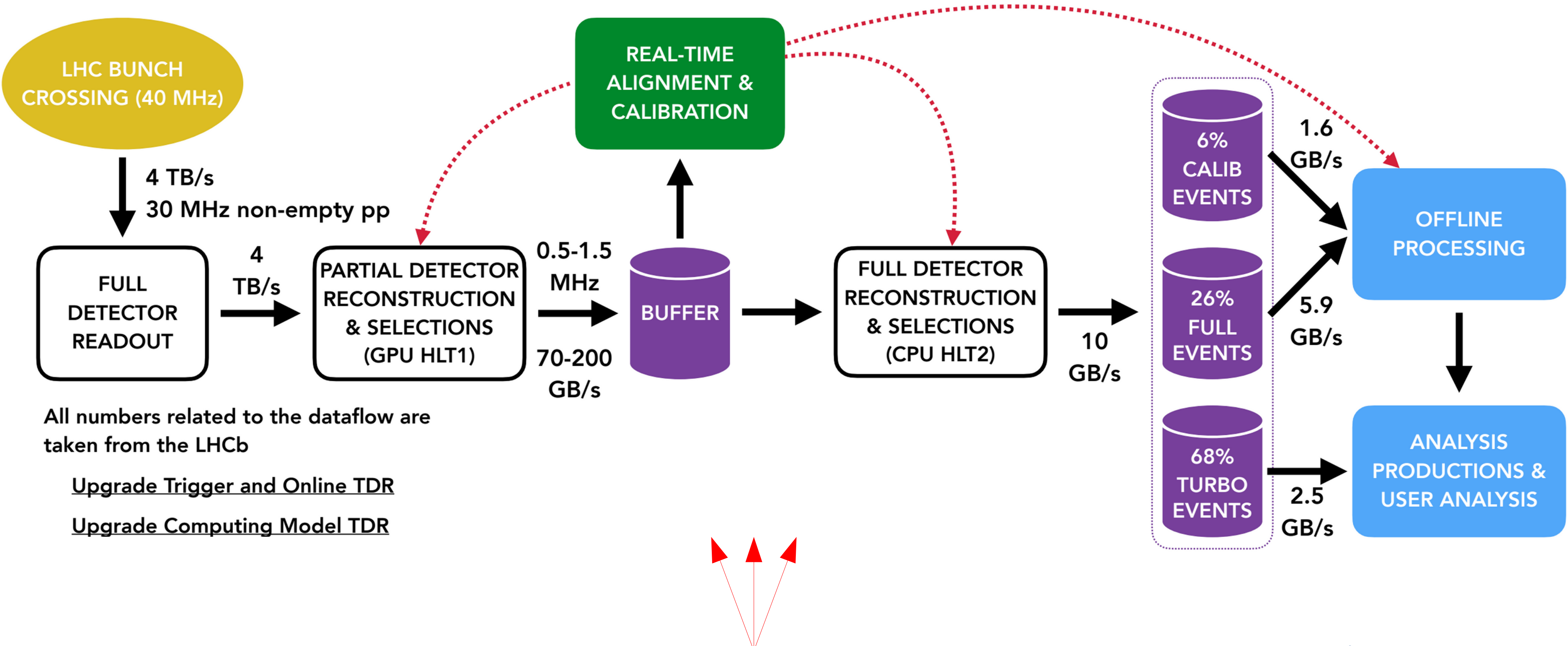

Memory
X0
X1
X2
X3
•••
•••
Y0
Y1
Y2
Y3
•••
•••
Z0
Z1
Z2
Z3
•••
•••
XMM
Registers
SIMD Arithmetic Instructions
X0'
X1'
X2'
X3'
Y0'
Y1'
Y2'
Y3'
Z0'
Z1'
Z2'
Z3'
Memory

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 trigger design - software QA



LHCb-FIGURE-2020-016

Software QA, testing and training

QA: 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

QA: 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)

QA: Software training

- Extremely important for any code development is not to only gather experts but also to pass the knowledge
  - Basic introduction to software used at LHCb: StarterKit [lhcb.github.io/starterkit/]
  - Many experiments are missing more advanced tutorials covering core online and offline software

- LHCb organized 28 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