A 30 MHz software trigger and reconstruction for the LHCb upgrade

Louis Henry, on behalf of the LHCb-RTA project
Connecting the dots , 20/04/2020

European Research Council
Established by the European Commission

Outline

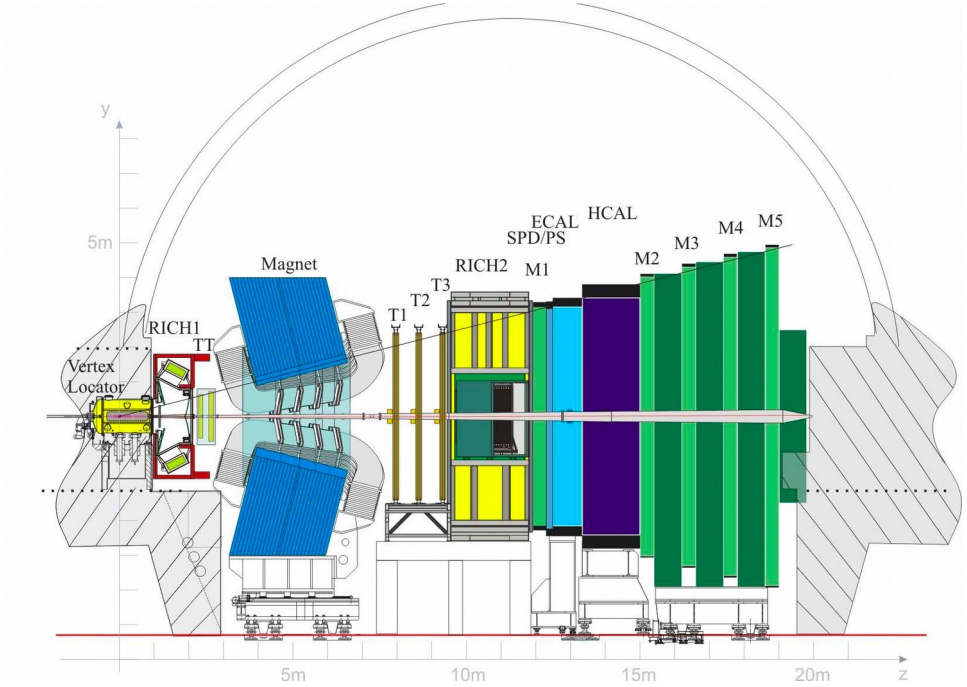
- The LHCb detector and trigger system
- The HLT1 reconstruction sequence
- HLT1 trigger & performance
- Conclusion

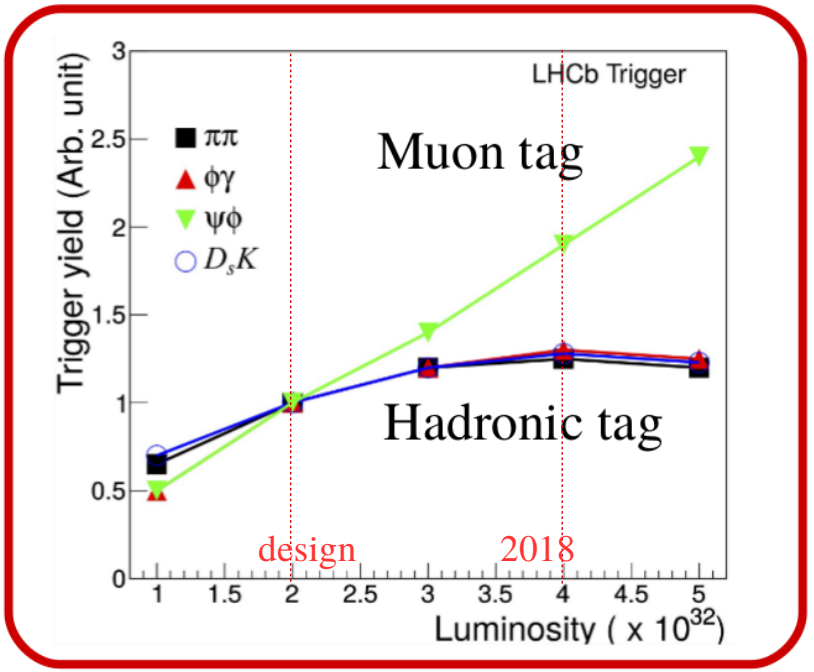
Outline

- The LHCb detector and trigger system
- The HLT1 reconstruction sequence
- HLT1 trigger & performance
- Conclusion

The LHCb detector

- LHCb is a detector along the LHC, specialised in the study of beauty and charm hadrons.
  - Upgrade of the detector planned for 2021.
- Need to take decisions at a frequency of 30 MHz, and reduce data rate by 3 orders of magnitude.
  - Need for a trigger. Previous hardware trigger relied on high-p_T → some lines saturate.

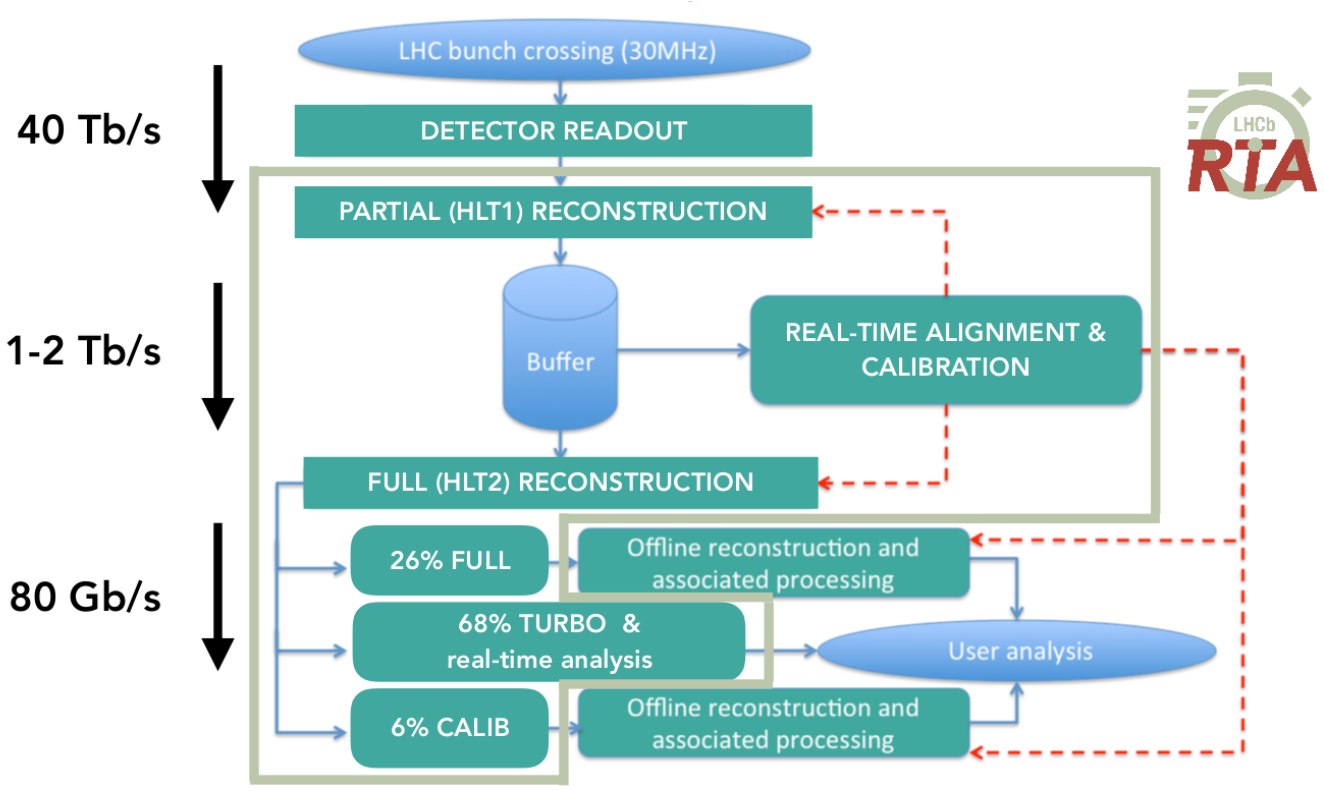




- We need more intricate physics handles in a harsher environment, and faster...
- … while keeping high efficiencies and flexibility, to ensure rich physics programme of LHCb.

LHCb data flow in the upgrade

- Solution:
  - Remove the hardware trigger, read the detector out at 30 MHz.
  - Perform partial reconstruction and selection to bring data flow from 40 Tb/s to 1-2 Tb/s: **HLT1**.
    - 40 Tb/s ~ workload of ATLAS/CMS in their high-level HL-LHC triggers… but in six years.
  - Align & calibrate the detector in real time.
  - Perform full reconstruction with offline quality in real time: **HLT2**.



Trigger of the upgraded LHCb detector: HLT1

- Goal is to use displaced tracks and/or muon signatures to bring down data flow from ~40 Tb/s to 1-2 Tb/s.
  - Similar to former LHCb HLT1 trigger, which used to run at ~1 MHz.

- We track the progress of the throughput as measured on a reference node, with a goal line that was set at 30,000 events/s/node.
  - A bit arbitrary because it compounds technology changes, budget considerations, etc.
  - Still gives an idea of where we are.

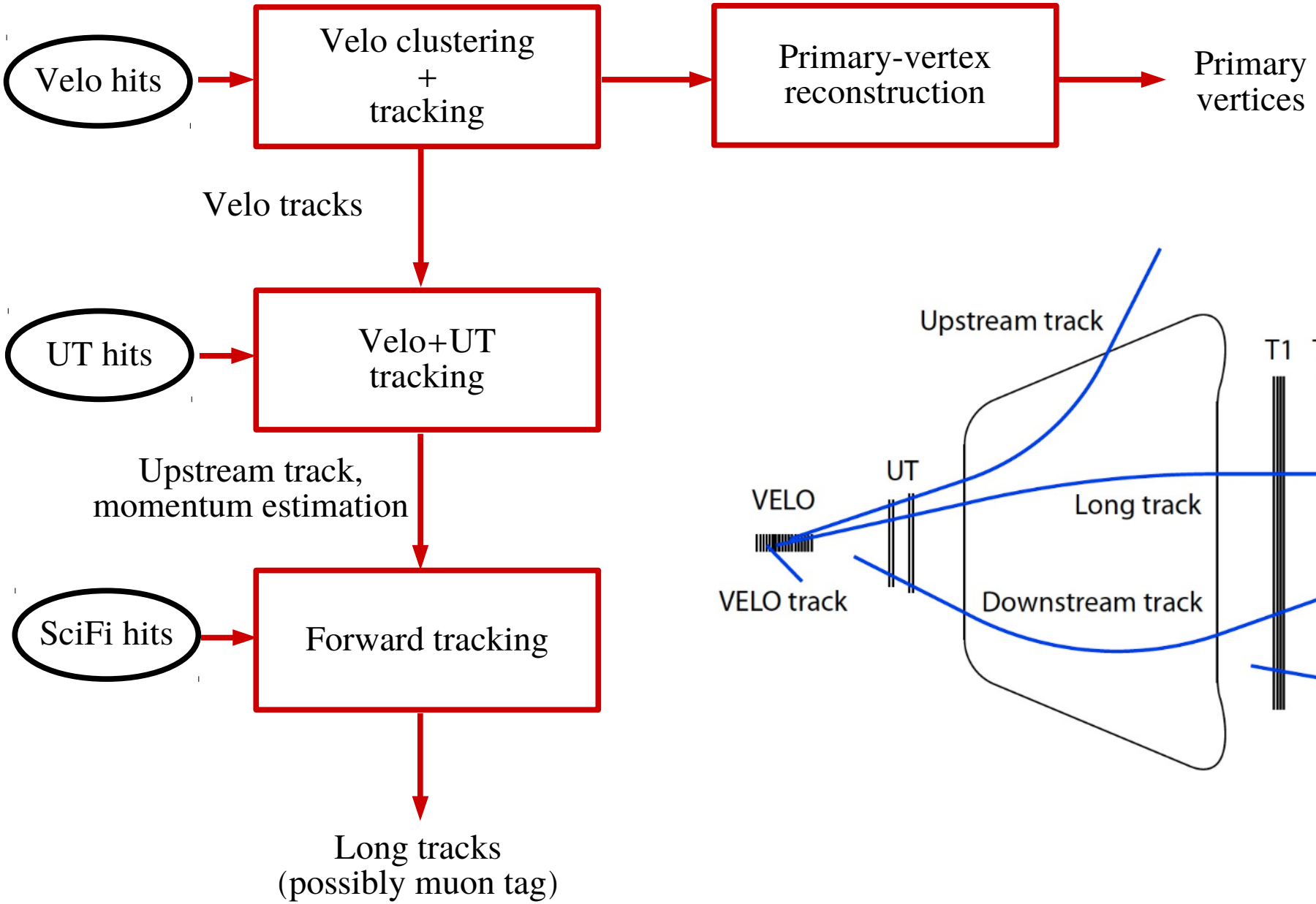
- Throughput on January 2019: ~ 8,000 events/s/node.
- Throughput in April 2020: ~38,000 events/s/node.

Over few years, better use of architecture, code simplifications and algorithm refactoring have led to huge improvements.

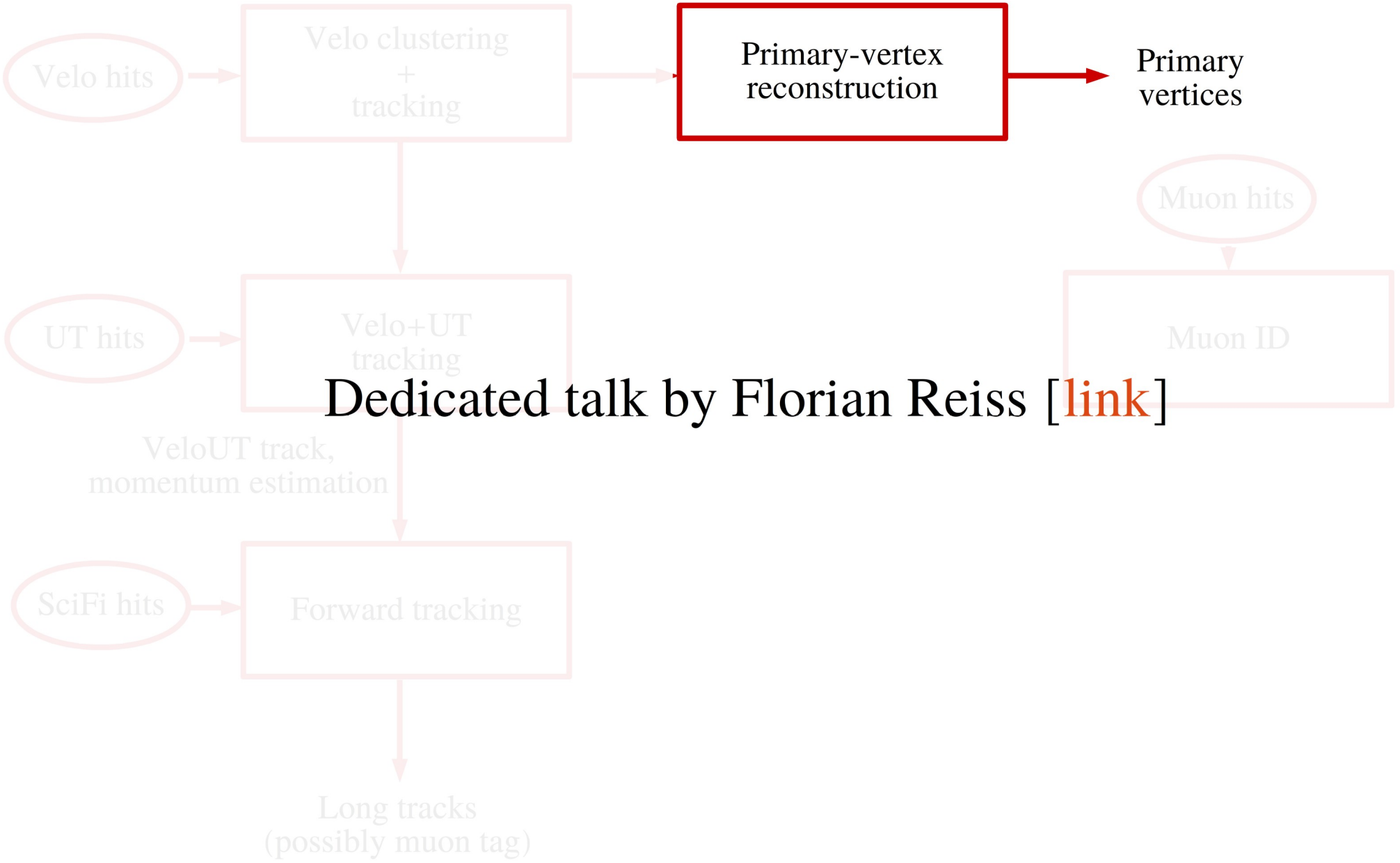
Outline

- The LHCb detector and trigger system
- The HLT1 reconstruction sequence
- HLT1 trigger & performance
- Conclusion

HLT1 reconstruction sequence

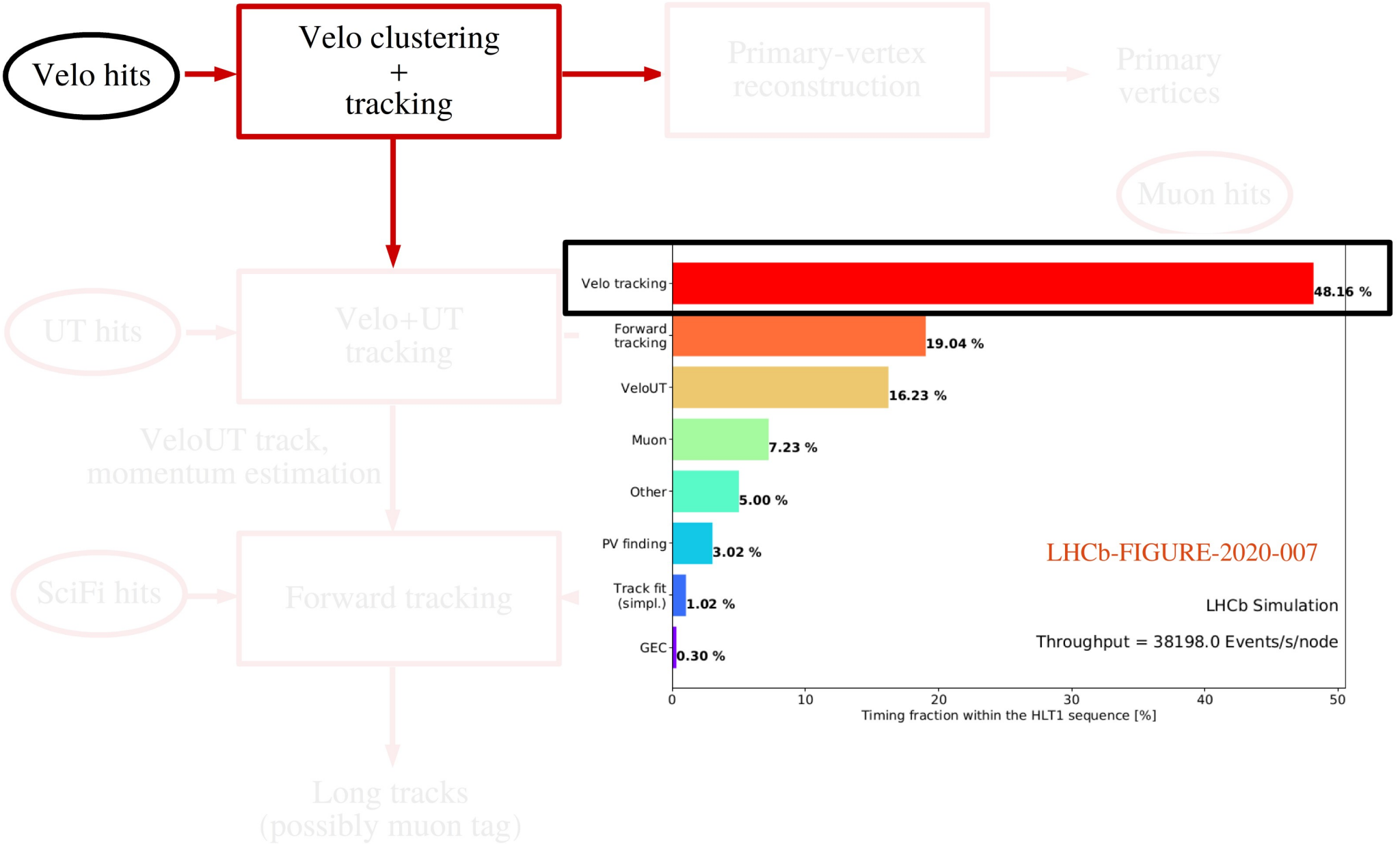


HLT1 reconstruction sequence: primary vertices



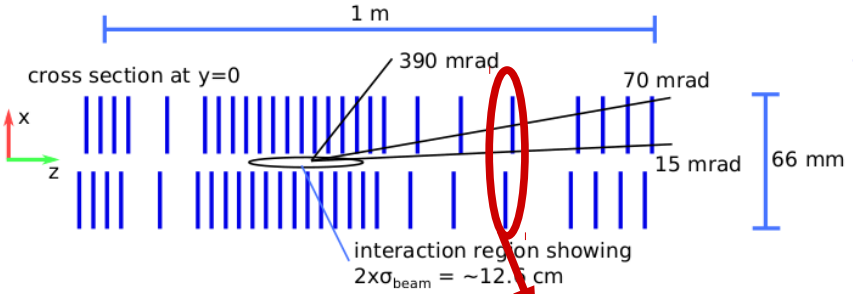
Dedicated talk by Florian Reiss [link]

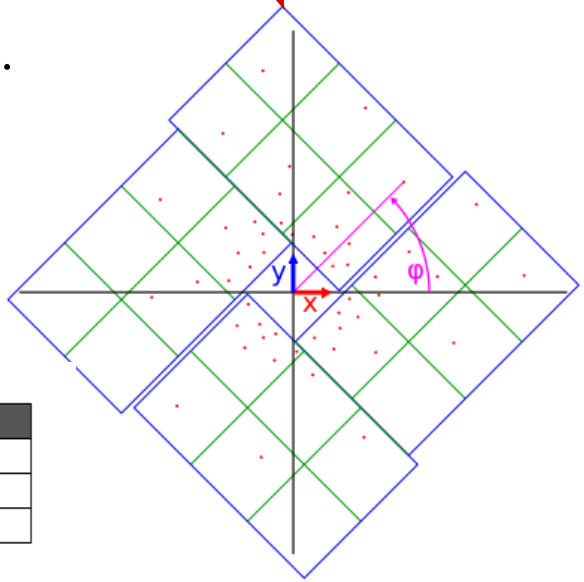
HLT1 reconstruction sequence: velo clustering and tracking

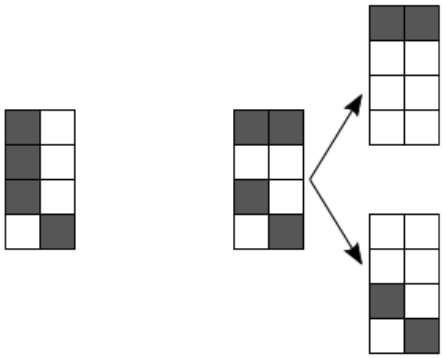


Velo clustering: current CPU implementation

- First step of the whole sequence.
- LHCb currently studying FPGA solution (RETINA).
  - Still would keep CPU as backup.
- Pixel detector → need to collect clusters into hits used by tracking.
  - Connecting labelling algorithm, adapted for sparse matrix.
- Raw banks transmitted by SuperPixels (2x4 pixel matrices).
  - Contain bit that tells if neighbouring SuperPixels or not.
  - If neighbouring superpixels, possibly split them in several that have only one connected component.
  - Test for adjacency and merge clusters.
  - For each final cluster found, calculate statistical moments and φ, translate to local coordinates.
- Operations realised by boolean masks and on raw banks (no decoding) → very fast.

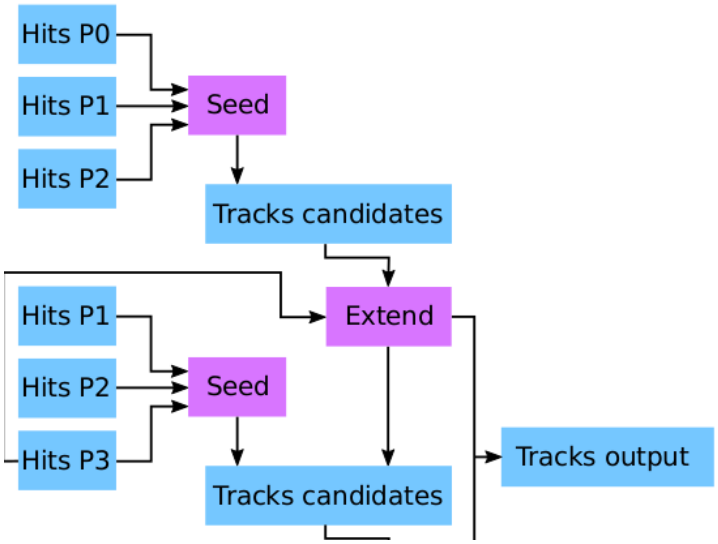


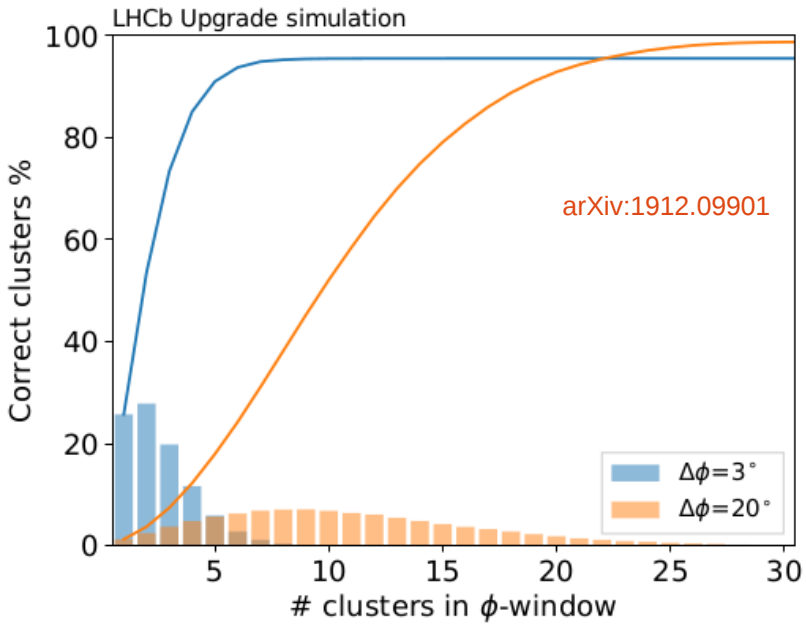


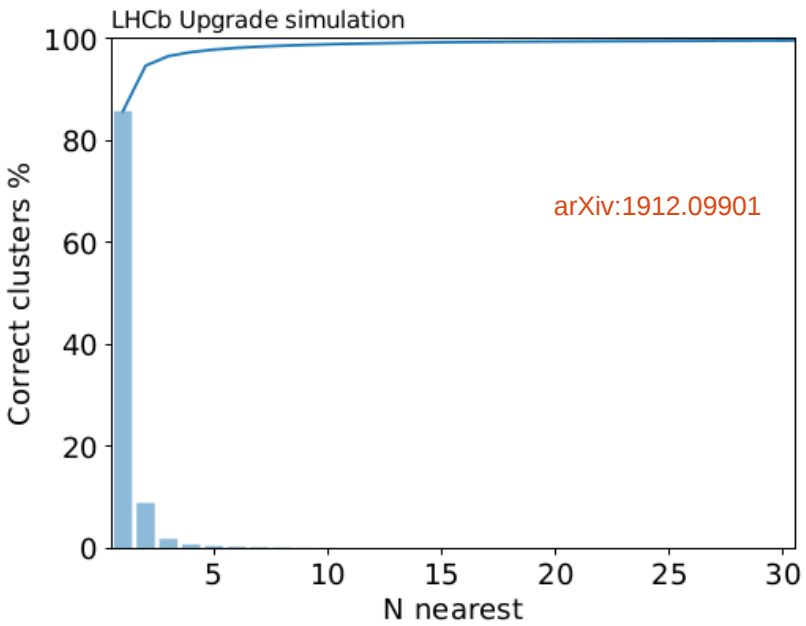


Velo tracking: triplet search principle

- Triplet searching: O(100) hits/plane → O(100^3) triplets.
  - Unmanageable, even extending pairs is expensive (O(100^2)).
  - Need for acceptance window.
- Tracks come more or less from the beam line.
  - Hits of the same track have similar φ coordinate.
  - Issues: we want displaced tracks, which require large windows, variable number of hits → inefficient in a parallel architecture.
  - **Solution: seed the N closest hits in φ.** Constant number of candidates + better efficiency in sparse regions.

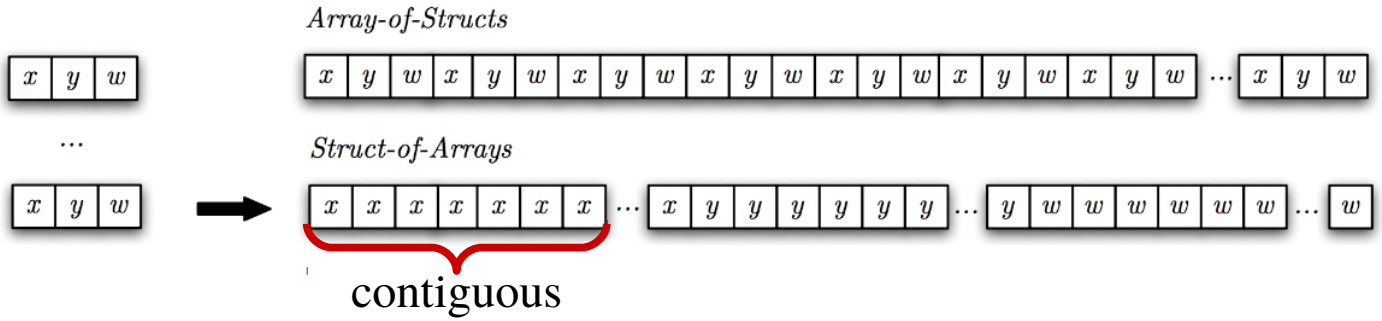




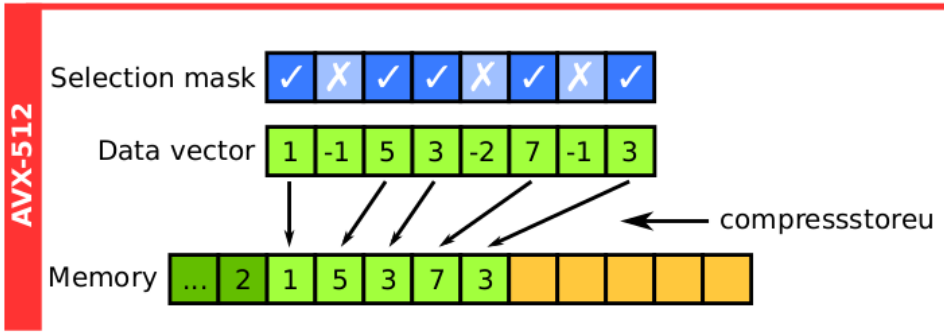


Velo tracking: triplet search implementation

- Two mains ways of getting faster: more cores, or exploit intrinsic parallelism (SIMD).
  - Needs data to be the most "local" possible, and full vectors.
- Data arranged in structure-of-array (SoA) layout:

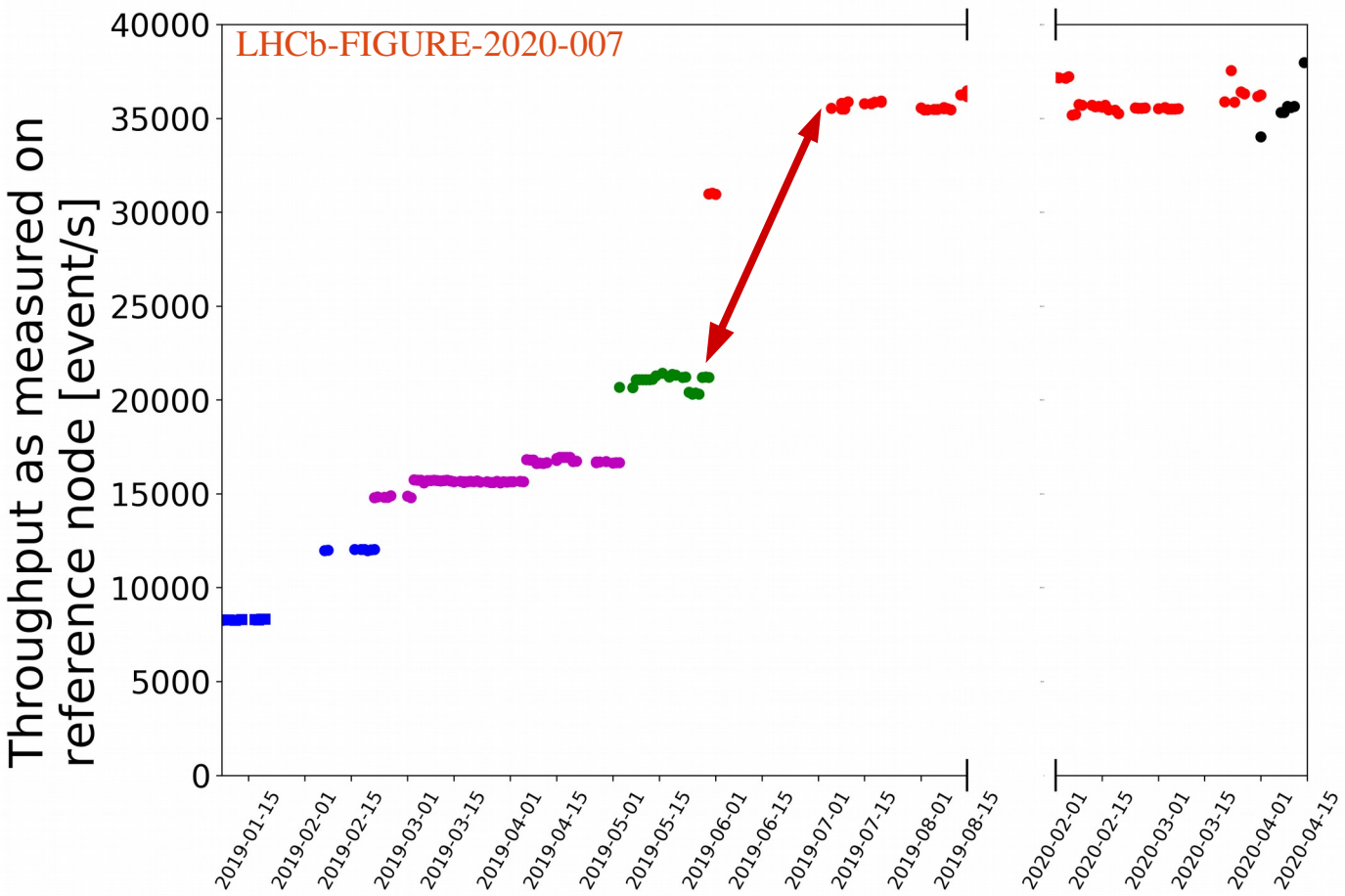


- Seeding is only done on three layers at a time, and changes one layer per one layer.
  - Only 3 layers allocated on the stack .
- At each step, data compressed to remove used hits → no wasted element, contiguous data.
  - Has a cost, but costs less than having empty registers.

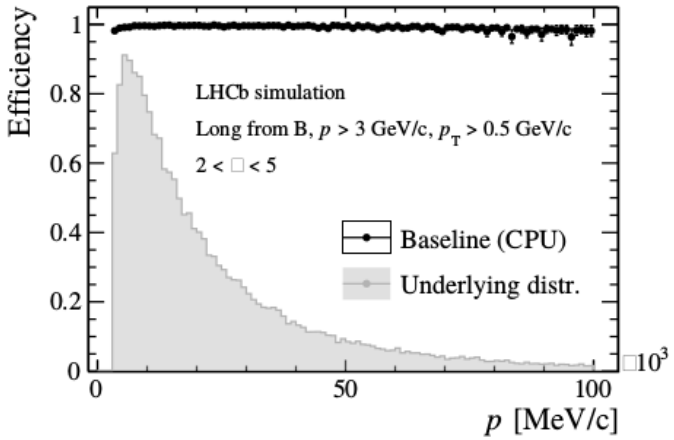


- Developped custom SIMD wrappers to support all the backends (SSE, AVX2...).

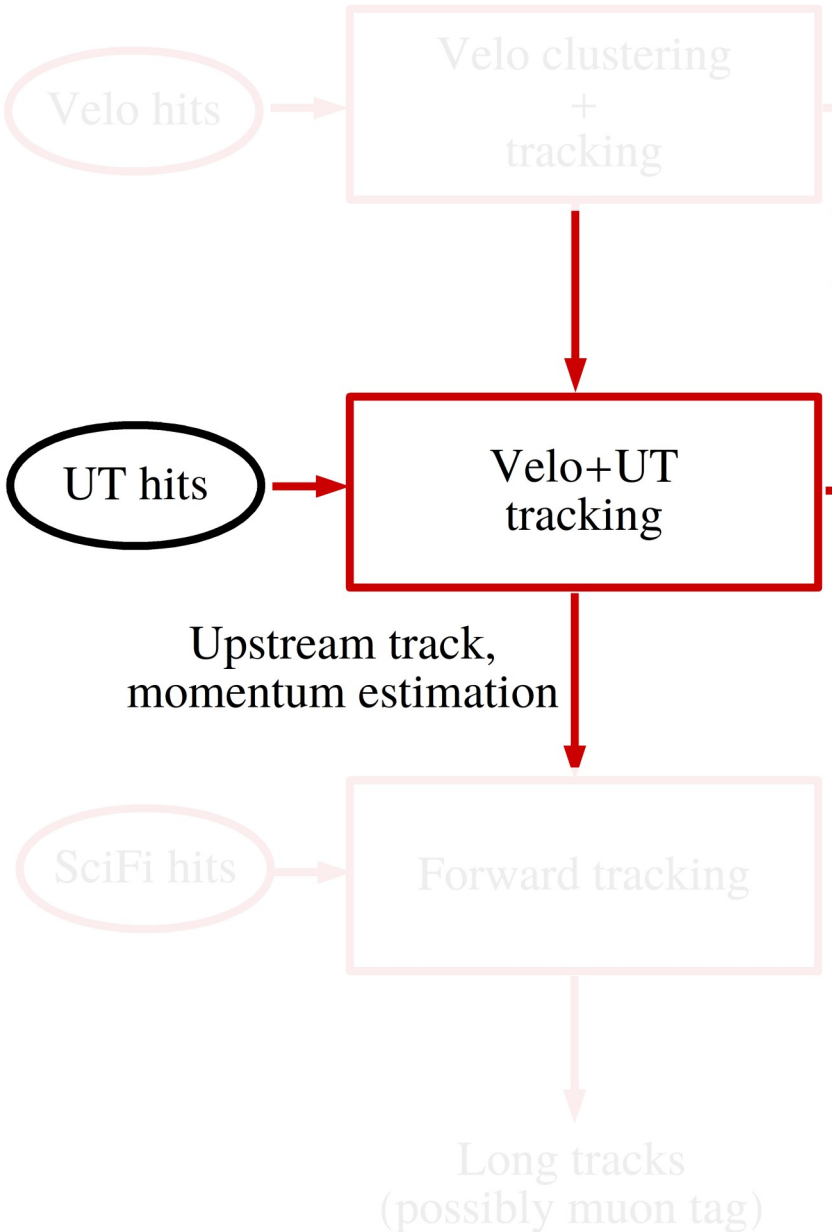
Velo clustering and tracking: current status

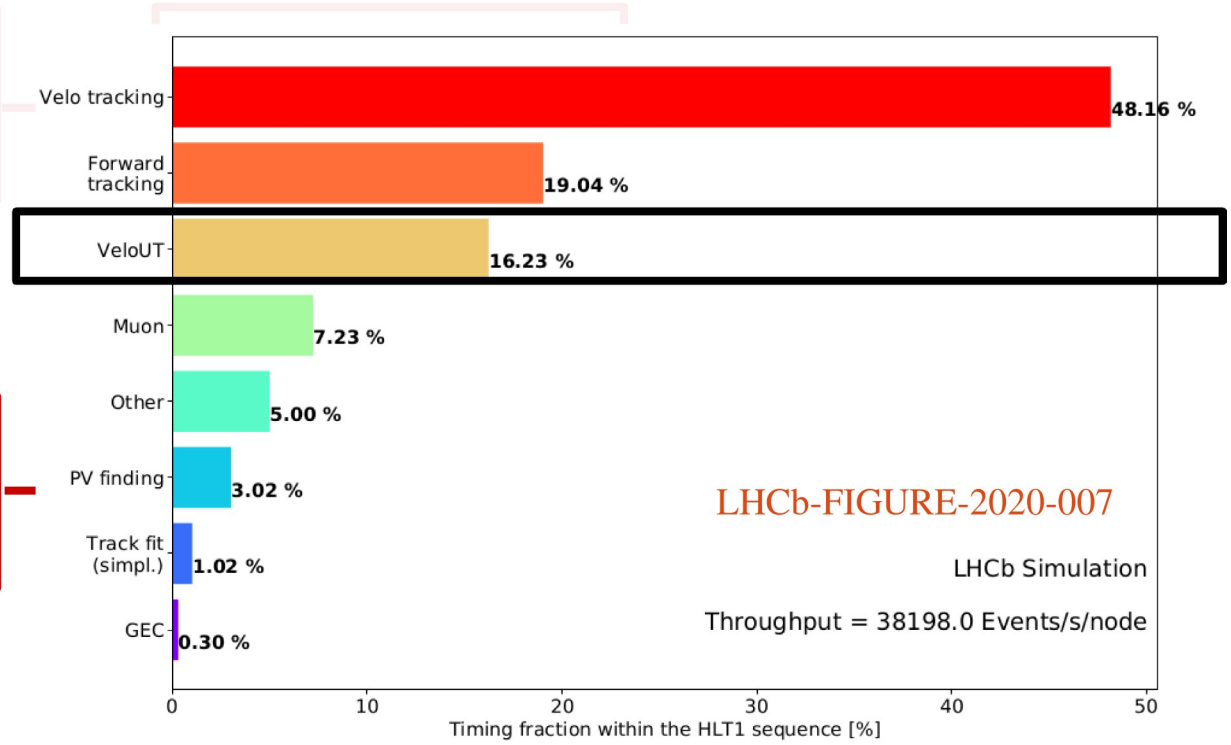


- Described changes main contributors to the green-red jump in throughput (along with lighter event model).
- Still largest contribution: not surprising considering the workload.
- Excellent efficiency → does not cap performance of algorithms downstream.

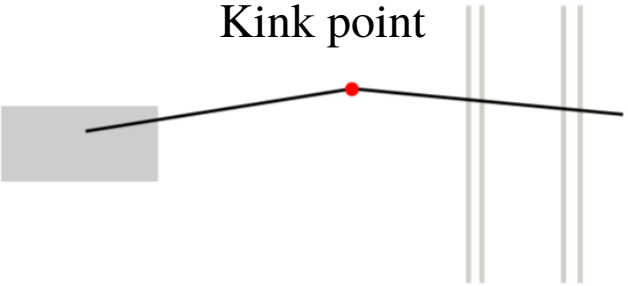


HLT1 reconstruction sequence: adding UT hits





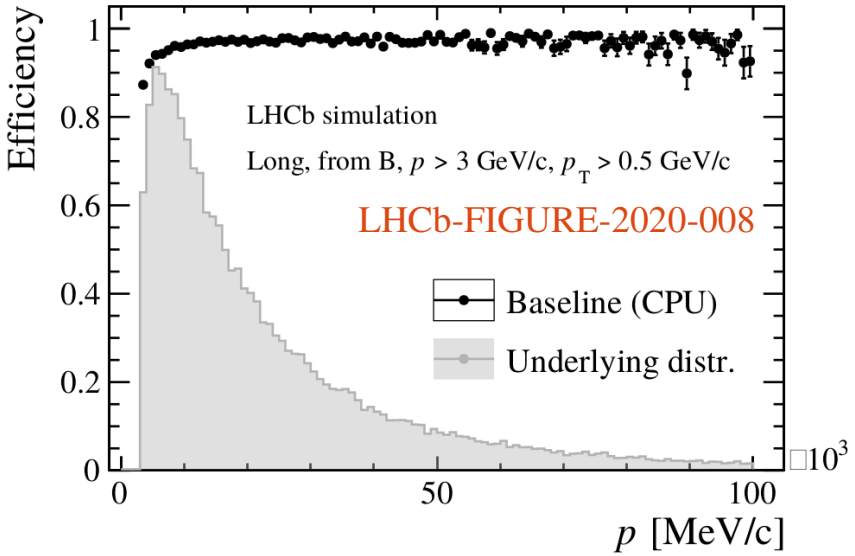
Velo+UT tracking: refining the model



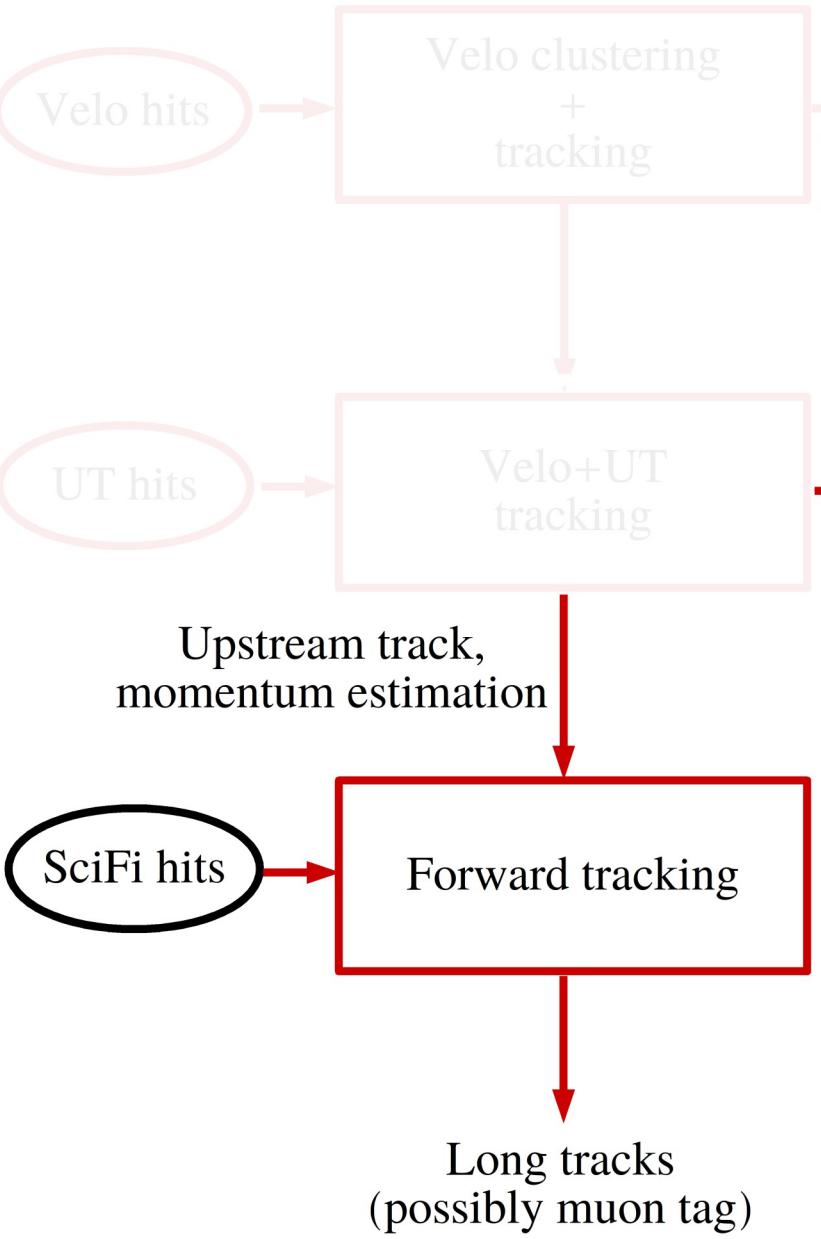
- Velo stubs can be extended to the Upstream Tracker (UT):
  - Small magnetic field → **momentum estimation**.
  - Goal: provide forward tracking with high-purity sample of tracks, with good momentum estimation.
- Scattering, VELO resolution, momentum → here, kink position is a parameter of the fit.
  - Dependency of uncertainties with respect to momentum parameterised on Monte-Carlo.
- Additional complexity, but improved resolutions, χ^2.
  - Ghost rejection using Fisher discriminant. Inputs: p, p_T and χ^2/ndof.
- Fiducial cuts applied → veto tracks that would go inside of the beam pipe.

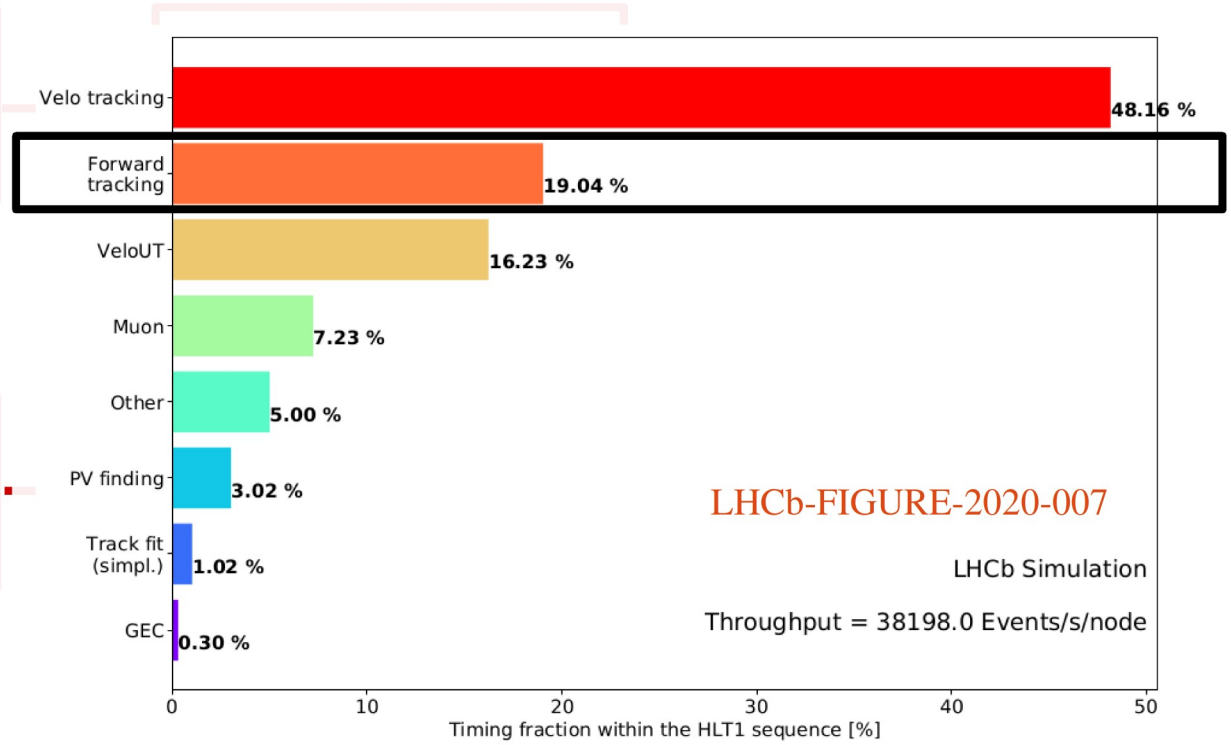
- Algorithm recently largely vectorised over Velo tracks (extrapolation, mapping, selection, fit) and hits (search).

- Right: still a large efficiency on target type of tracks.



HLT1 reconstruction sequence: forward tracking





Forward tracking

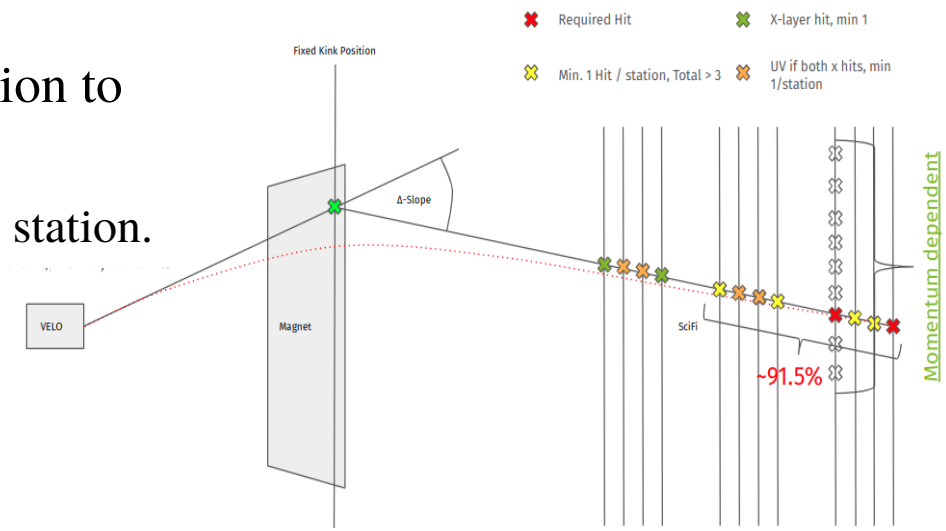
- VeloUT provides (x,y,t_x,t_y,p) → allows extrapolation to last forward tracker station.
  - Fixed kink position, linear extrapolation to last SciFi station.
  - Search for doublets in last station + 1 U/V.
- Trajectory in y described by:

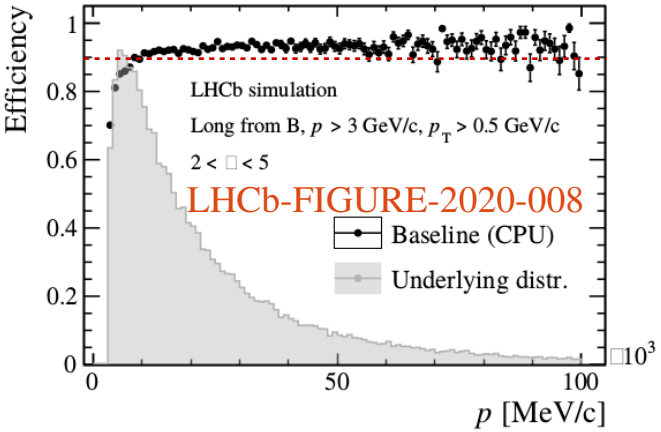
$$y_{pred} = y_0 + t_y \cdot z + y_{corr}$$

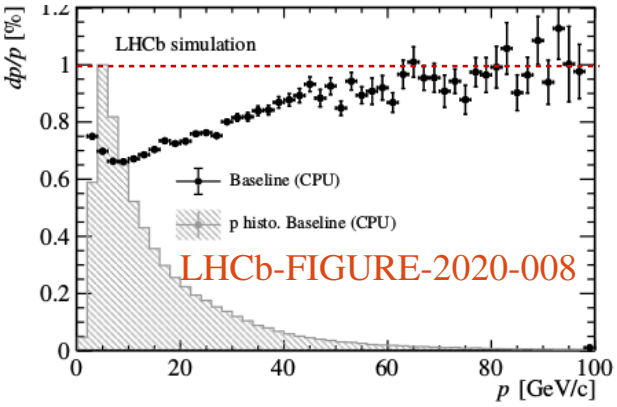
y_{corr}: polynomial in t_x, t_y, trained on MC.

- Trajectory in x described by MC-trained polynomial, but uncertainty dominated by VeloUT resolution on p.

- Efficiency over 90% for high-momentum tracks.
- Recently vectorised, but most speed gains through better tolerance windows.
- Relative momentum resolution below 1%
  - Fitting only on x hits.





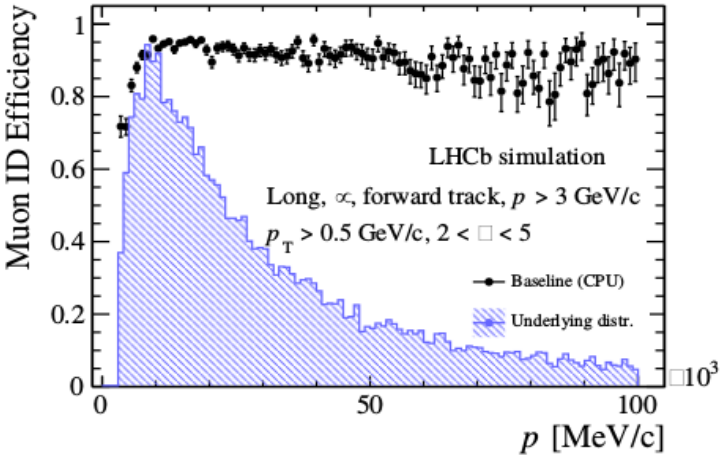


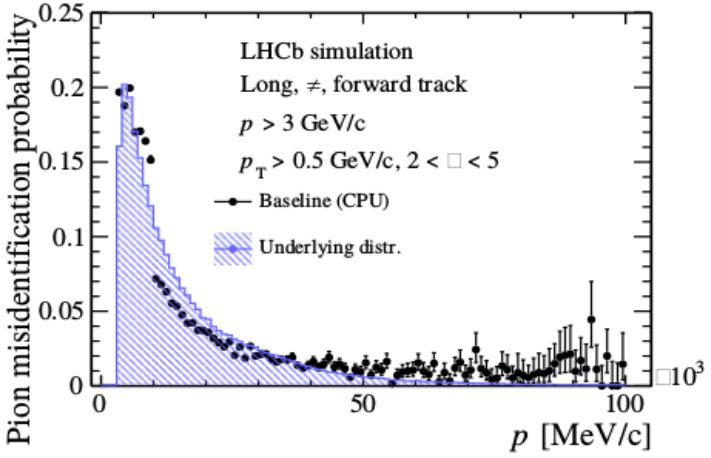
Outline

- The LHCb detector and trigger system
- The reconstruction sequence
- HLT1 trigger & performance
- Conclusion

HLT1 trigger: identifying muons

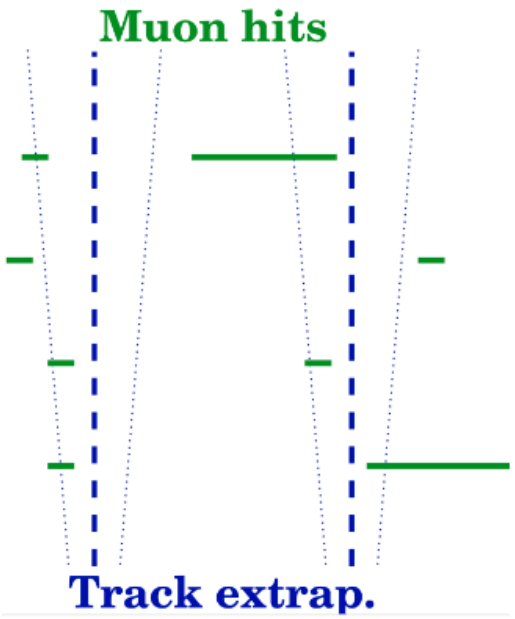
- Muon tag **essential** for trigger: events with *b* or *c* often have high-p_T μ, displaced J/ψ.
  - We expect 2 times more misidentification background than in Run 2.
- Improved decoding, adapted Run 2 muon tagging (use the uncorrelated sum of spatial residuals) → running at ~7% of HLT1 with performance below:





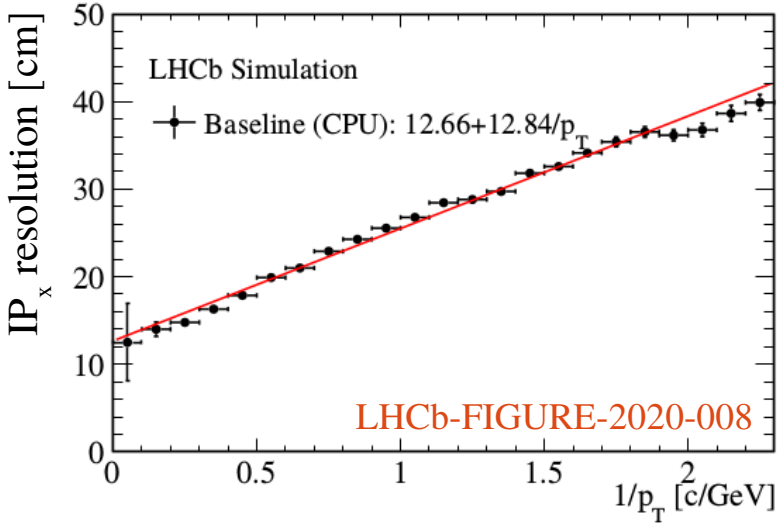
LHCb-FIGURE-2020-008

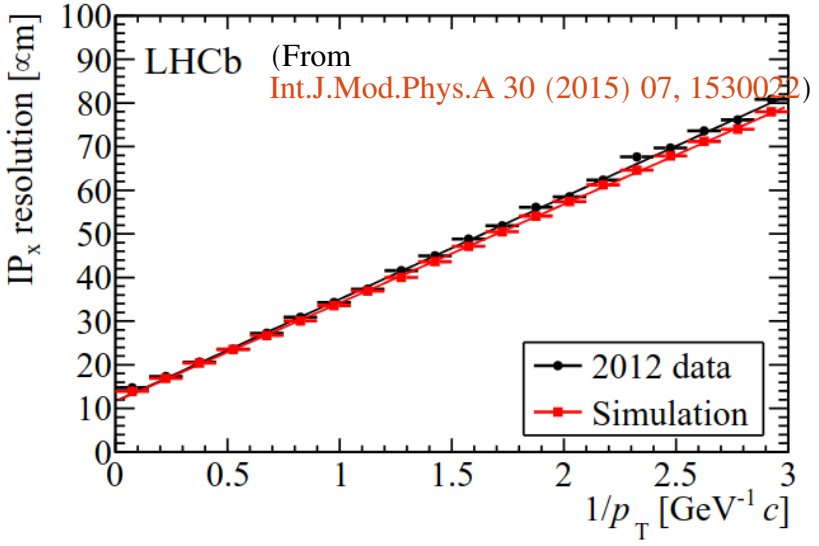
- New idea: insert a correlation matrix between hits, taking scattering into account.
  - Random hits are more scattered → less correlation.
  - Picture on the right show real muon (left) and hadron (right) with comparable uncorrelated value, but clearly different χ^2_{corr}.
- Results are better in terms of ID power, timing is comparable.
  - Paper in preparation.



HLT1 trigger: fitting VELO tracks

- HLT1 does not need best momentum resolution available, but needs:
  - Good two-track combinations.
  - Rejecting prompt backgrounds for some lines → good IP resolution and significance.
- Both need reliable estimate of $\sigma(t_x)$ and $\sigma(t_y)$ → refit VELO tracks.
- Perform a line-with-kinks model fit over VELO measurements, implemented using Kalman filter, scattering parameters obtained from simulation.
- Uses the momentum estimate from forward tracking.





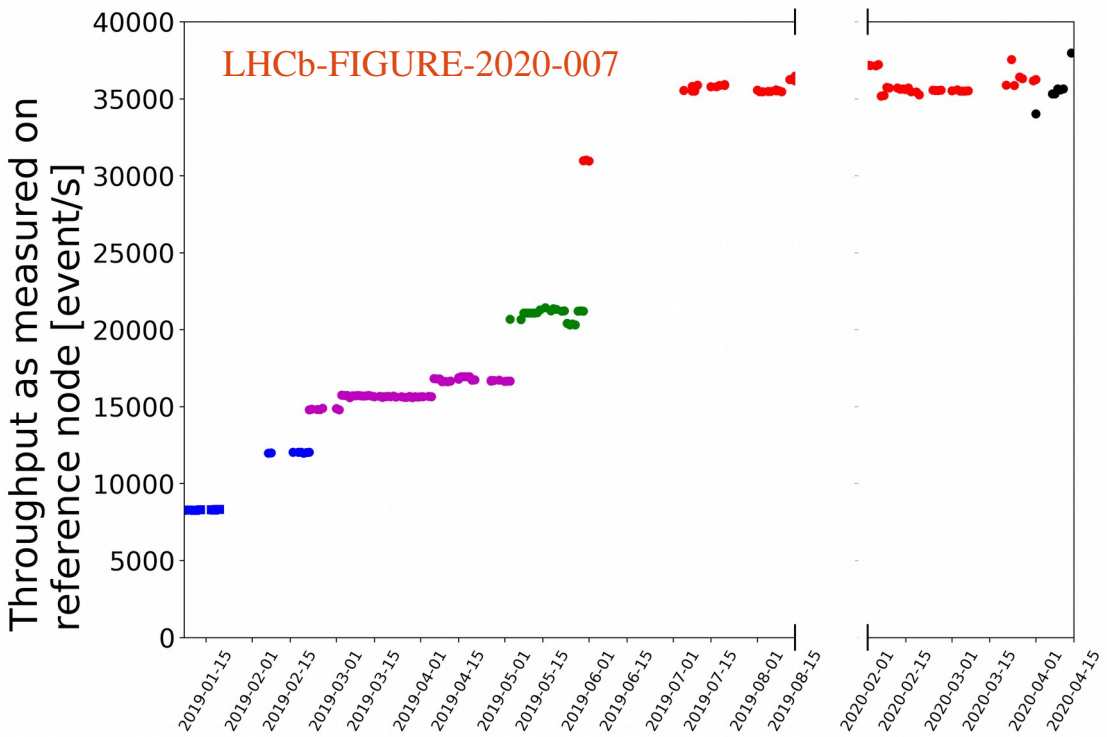
- Resolution of the same order of magnitude than in Run 2 full reconstruction.
  - Cuts are however different: it is not straightforward to compare the two.

Outline

- The LHCb detector and trigger system
- The HLT1 reconstruction sequence
- HLT1 trigger & performance
- Conclusion

Conclusion: status of the HLT1 CPU

- Throughput of the HLT1 CPU has been improved by nearly a factor 5 in a bit more than a year without cutting on performance.

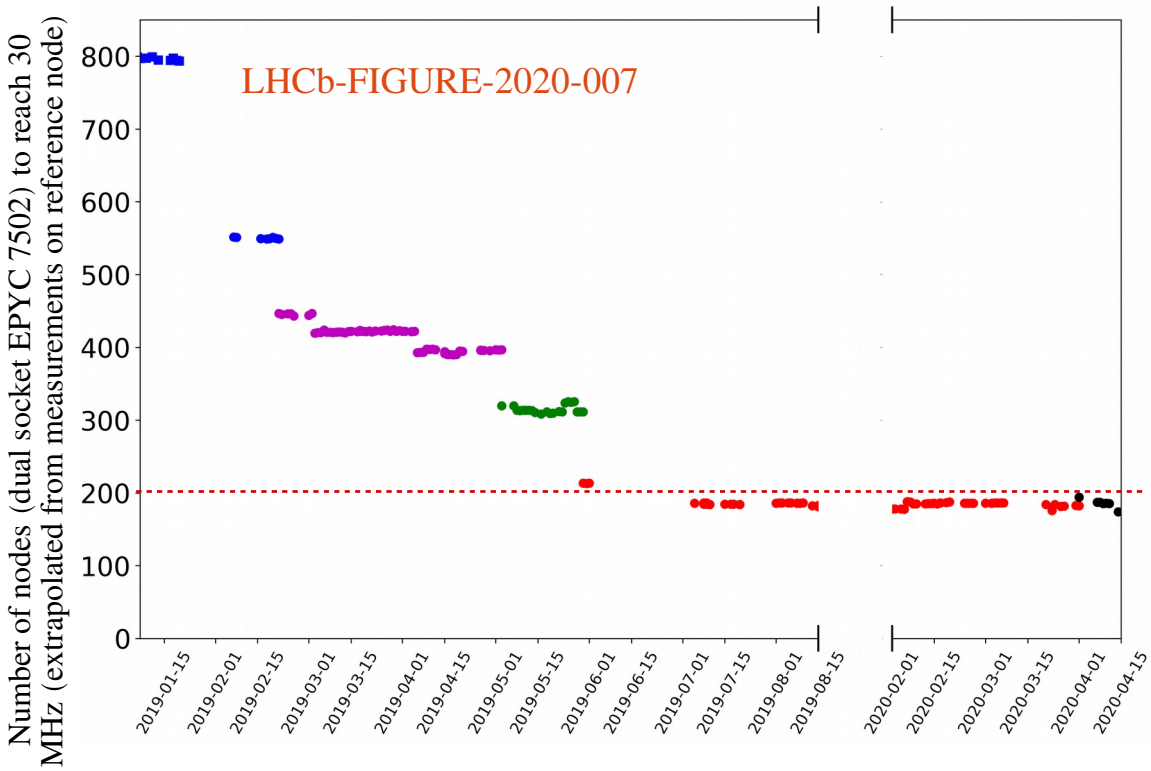


- This has been made possible by:
  - **Rewriting algorithms** whose performance used not to be critical (e.g. decodings).
  - **Improved use of architecture** and intrinsic parallelism, through coding and algorithm design (e.g. velo tracking).
  - Use of our **experience** operating this detector: trade-offs, modelling have been revisited (e.g. simplified Kalman fit, forward tracking).
- And, for most algorithms, **all of the above** → no "one fits all" procedure.

- Currently have reached a reasonable throughput: improvements can be used to add new pieces or relax some requirements.
  - For instance, adding cuts & muon identification counterbalanced by vectorised VeloUT & forward.

Conclusion: status of the HLT1 CPU

- Taking the throughput measurements and scaling studies documented in LHCb-FIGURE-2020-001 (relevant plot also in backup) we see that a full CPU HLT1 would need fewer than 200 EPYC 7502 servers (AMD CPUs).

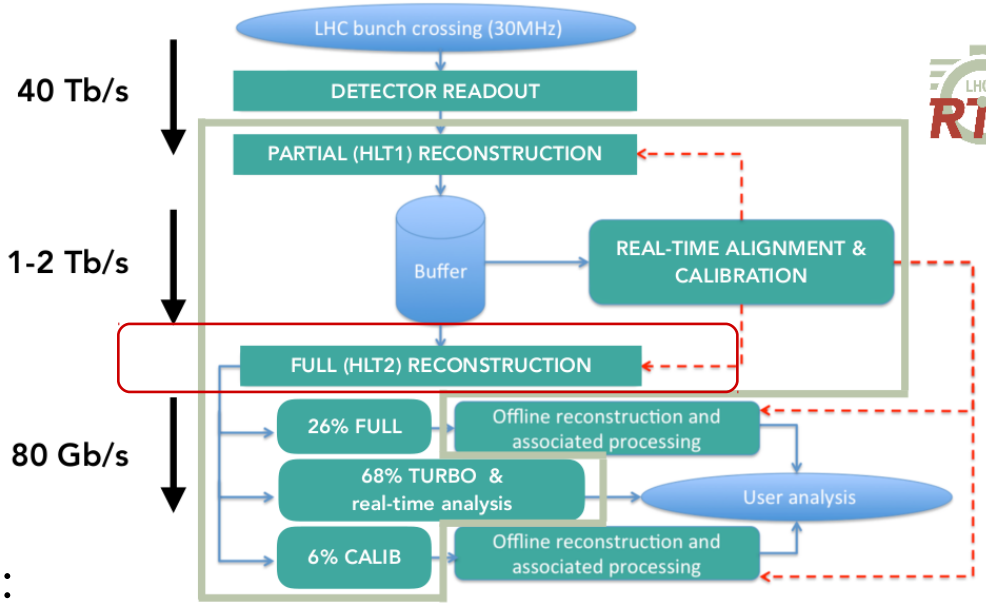


- In other words, the HLT1 CPU is able to **deal with 40 Tb/s data rate for a O(1M) CHF budget.**

- Currently, the LHCb collaboration is choosing between CPU and GPU (Allen, see Thomas's talk, arxiv:1912.09161).

Conclusion: turning to HLT2

- After HLT1, still have to deal with 1-2 Tb/s.
- The second level, HLT2, then performs the full reconstruction of the event at **offline quality**.
  - Computing challenge of its own.
- One of the main ideas behind HLT2 design is:



Data quantity/s = n(events to write/s) * Size(event)

- Events that go to the TURBO stream allow to **save only the interesting part of the event** → reduces average size of the event → allow for many more saved events.
  - Especially useful in the context of a 1 MHz charm production.
- TURBO stream already put in place with success during the Run 2 of LHCb [JINST 14 (2019) P04006].

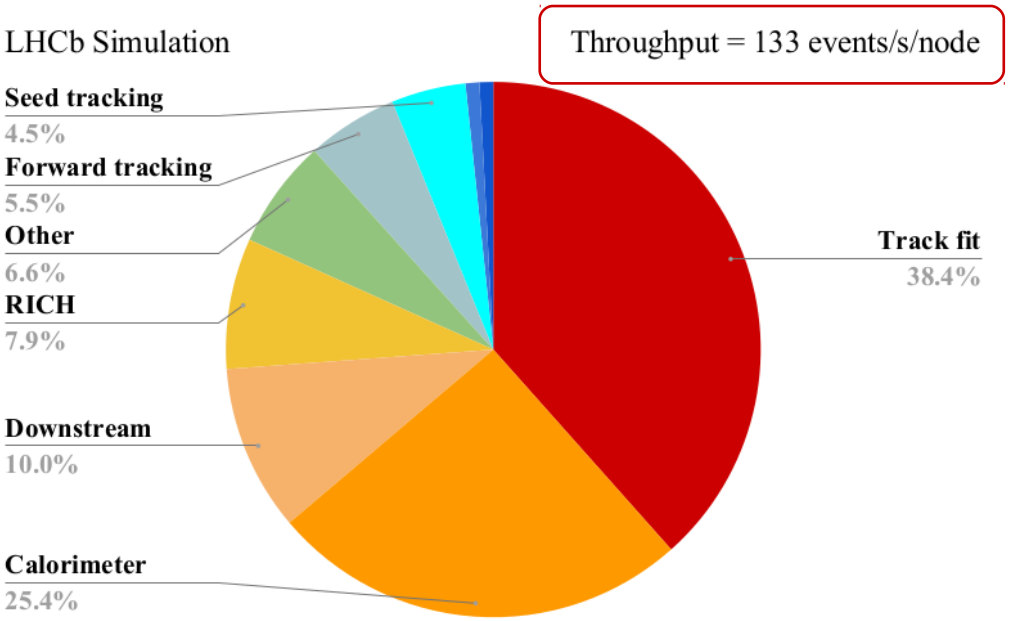
HLT2 status: sequence and throughput

- The HLT2 reconstruction sequence is quite different:
  - Forward tracking without any p_T threshold.
  - Addition of T-station track reconstruction (see this talk), downstream reconstruction, matching algorithm.
  - Full Kalman fit of the tracks.
  - Addition of RICH and Calorimeter information, more evolved muon treatment.

- On top of this, have to run hundreds of exclusive lines.

- Current HLT needs a significant speedup to deal with data volumes delivered by HLT1 → becoming a priority.

- Recent improvements have shown that significant gains are to be expected.
  - And, if anything, story of HLT1 has shown it as well.



LHCb-FIGURE-2020-007

LHCb trigger: an allegory

… and so runs fast but saturates

… so allows real-time analysis if given the time

… so that the fox can run

Thank you!

… or running conditions!

Credits to SMBC [link]

Backup: throughput scaling

- Reference throughputs calculated using Intel(R) Xeon(R) CPU E5-2630 v4
- Calculated factor is ~4.5