Data flow and data processing at LHCb

Zan Ren

School of Physical Sciences, UCAS

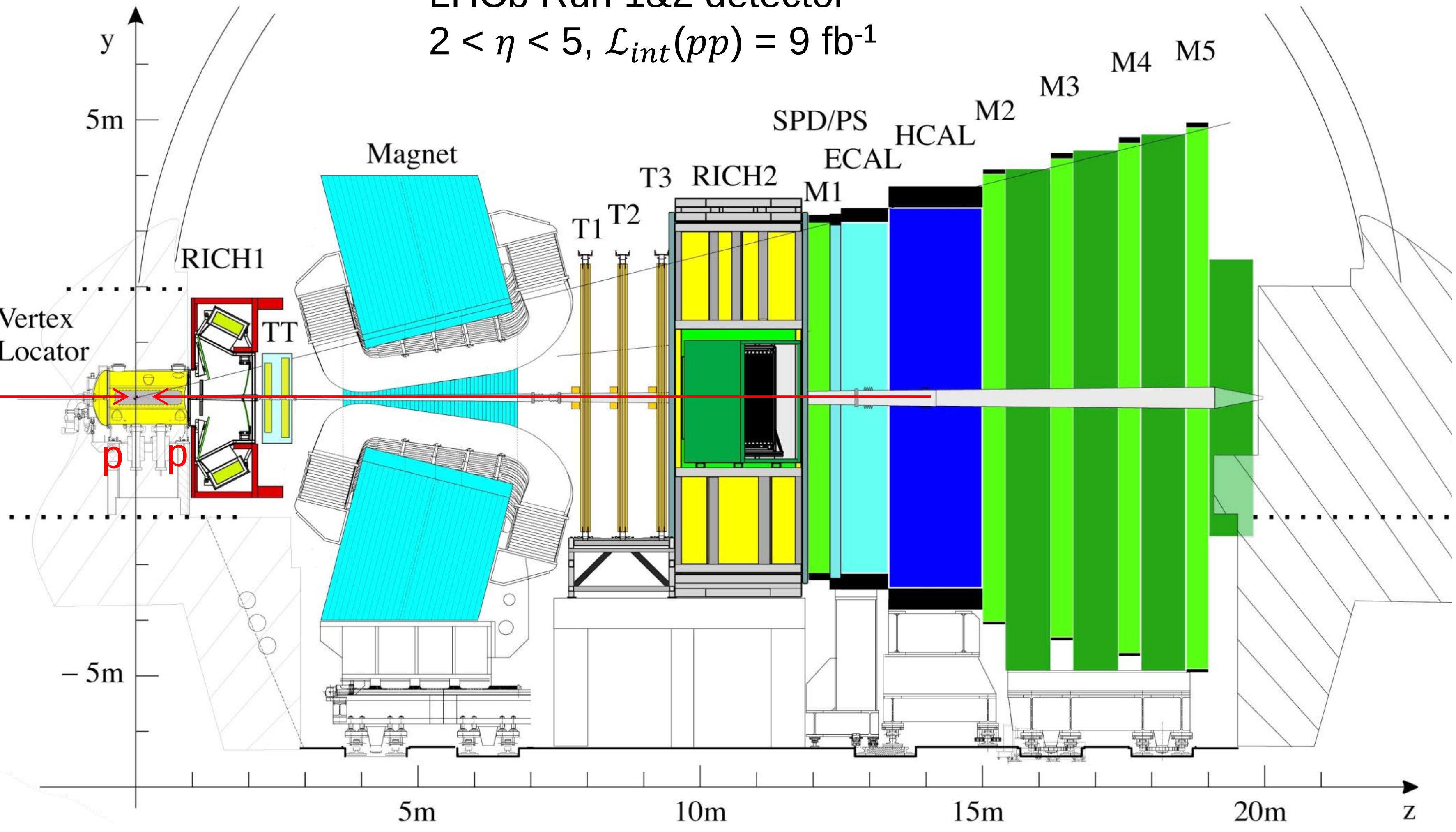
May 20

Chengdu, Sichuan

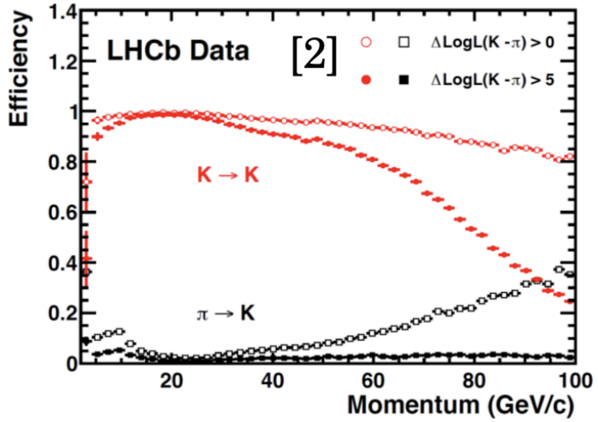
About LHCb detector

- Single-arm, forward. Specifically designed for heavy-flavour physics.

LHCb Run 1&2 detector
$2 < \eta < 5$, $\mathcal{L}_{int}(pp)$ = 9 fb^{-1}



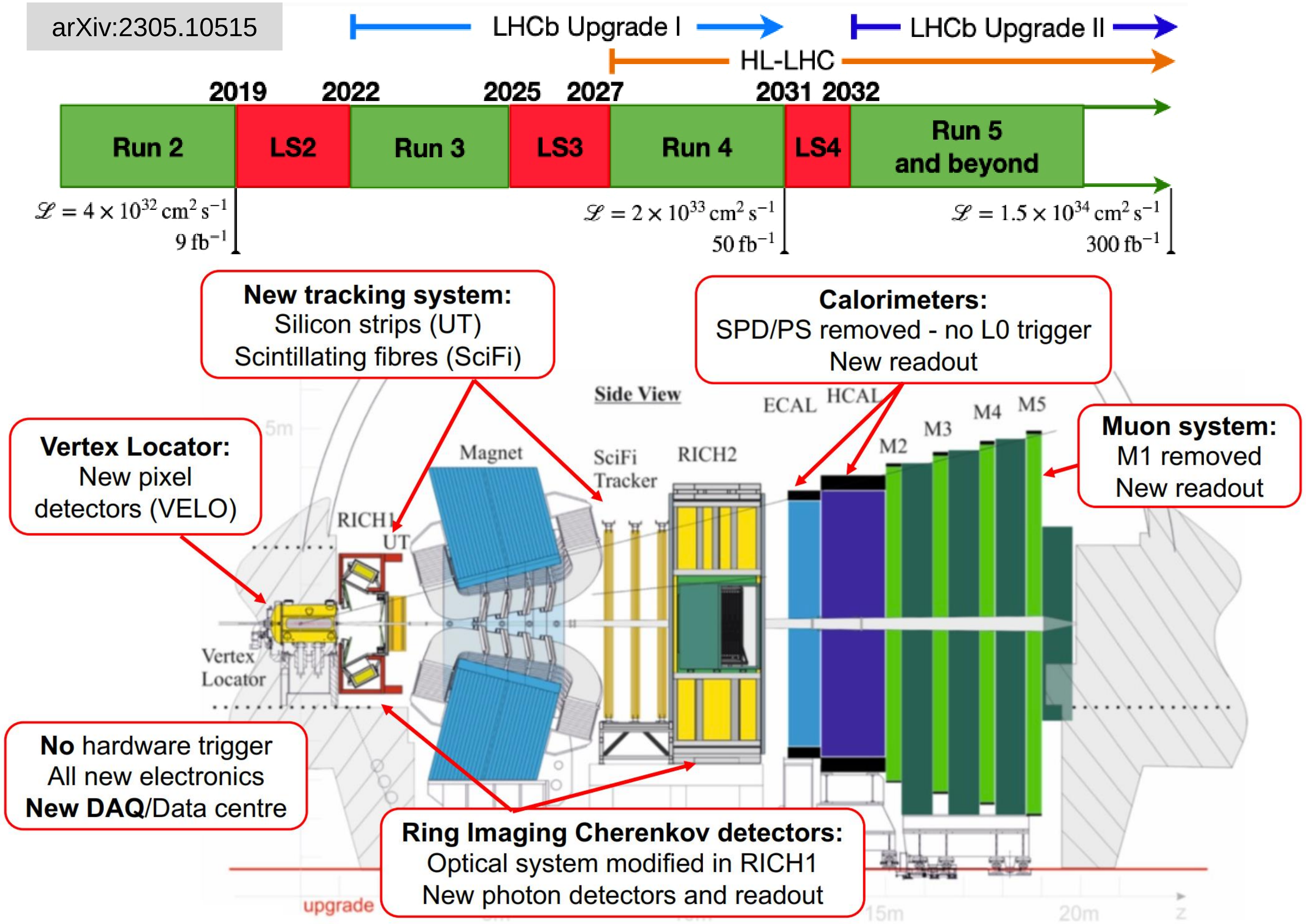
- Excellent tracking and vertexing
  - $\sigma(p)/p < 1\%$ @ $\epsilon_{track} > 96\%$
  - $\sigma(\mathrm{IP}) = (15 + 29/p_T)\ \mu m$
- Excellent PID
  - $\epsilon_{\mathrm{PID}}(K) \approx 95\%$ @ $\mathrm{MisID}(\pi \to K) \approx 5\%$
  - $\epsilon_{\mathrm{PID}}(\mu) \approx 97\%$ @ $\mathrm{MisID}(\pi \to \mu) \approx 3\%$



JINST3 (2008) S08005
IJMPA 30 (2015) 1530022

LHCb upgrade (Run3)

- The data flow of Run2 will be briefly reviewed in this talk, with a focus on Run3.



Almost a new detector!

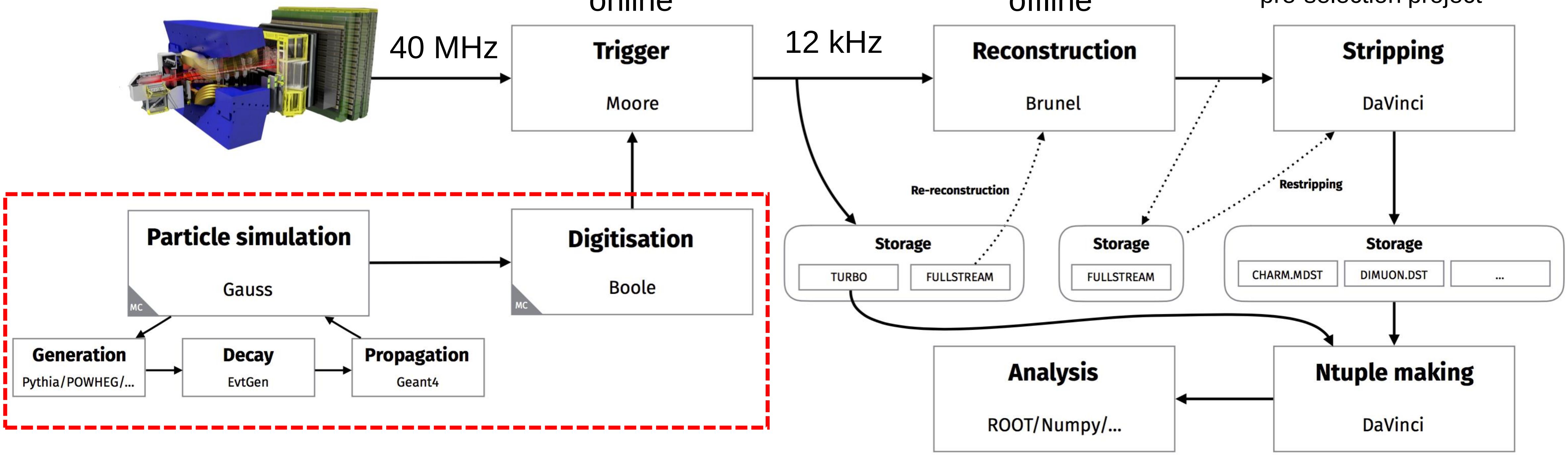
- A factor of 5 luminosity increase.
- $L = 2 \times 10^{33}$ cm^{2}s^{-1}
- Expect 23 fb^{-1} by 2025 (Run 3)
- Expect 50 fb^{-1} by 2031 (Run 4)
- Pile-up ~6 interactions.

More data, more challenges!

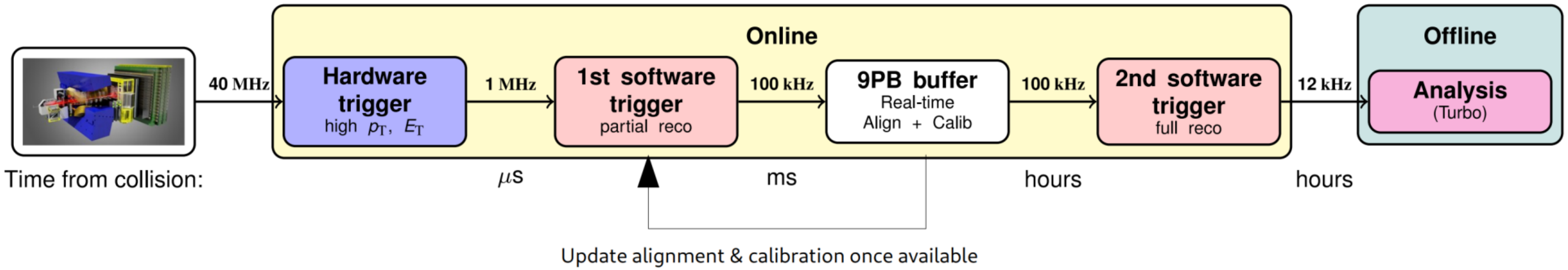
- Storage (space)
- Bandwidth (speed)
- Algorithm (data quality, UE, …)

Overview of old LHCb data flow

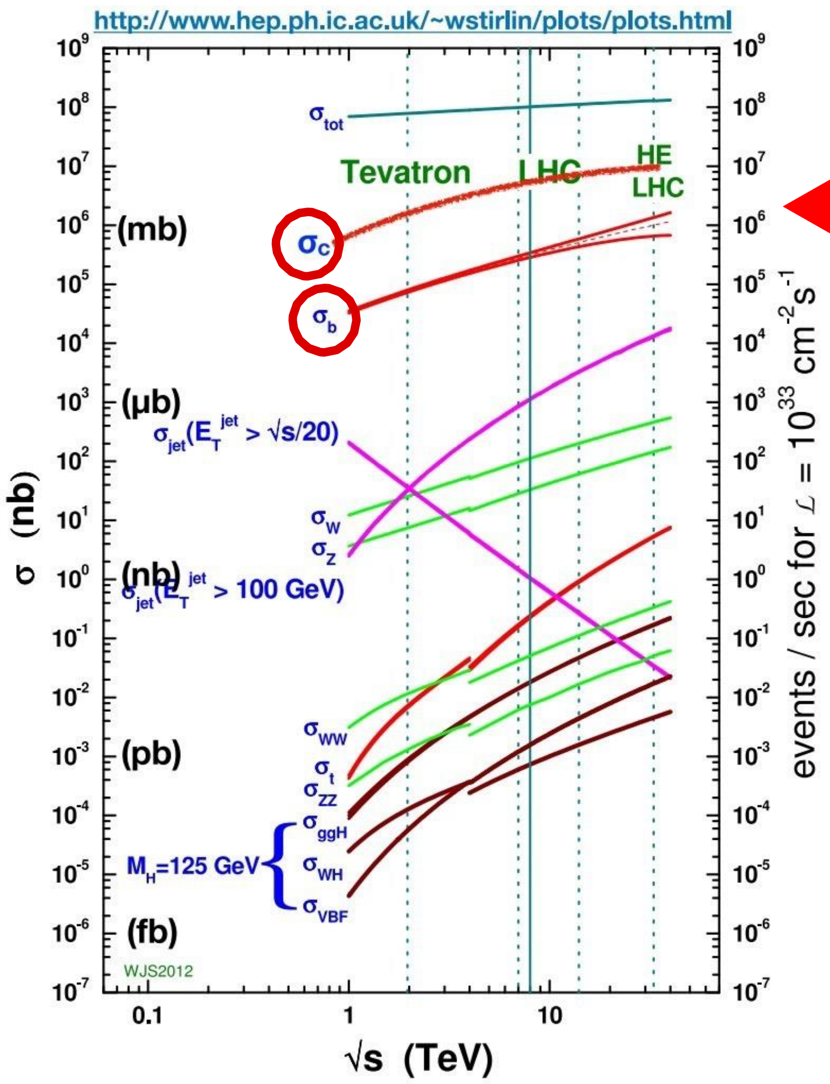
- LHCb Run 2 data flow



Triggers:



Challenges from the MHz era



Run 3: Luminosity of $2x10^{33}$ $cm^{-2}s^{-1}$, $\sqrt{s}$ = 14 TeV

LHCb Run3 is here! (@MHz level).

Bandwidth [MB/s] ~ Trigger output rate [kHz] x average event size [kB]

- Read out the full detector
  - No “simple” local selection criteria
    - ➔ Efficient hardware trigger not possible!
- Selective persistency events as output to storage
  - Up to 100 kB event size, can only transfer 10 GB/s to long-term storage
    - ➔ At most 100 kHz if full raw event is stored!

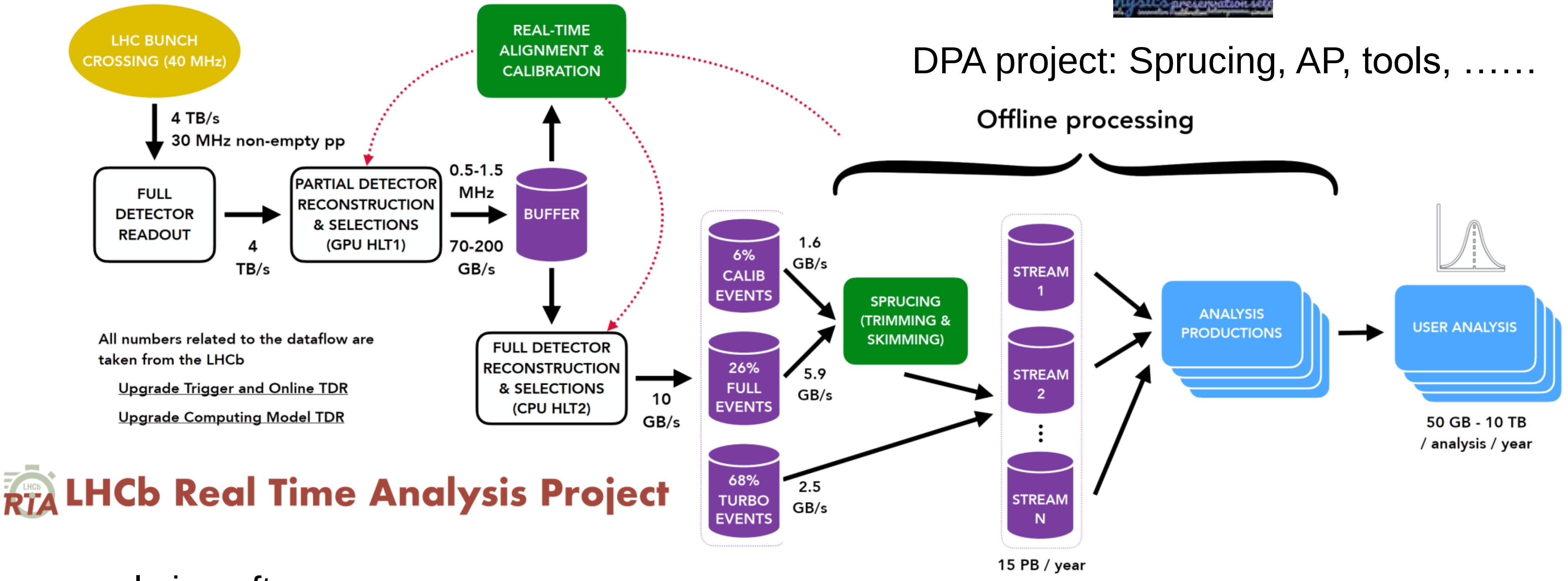
Trigger design goal:

30 MHz (4TB/s) → high-efficiency full-software trigger → 100 kHz (10GB/s)

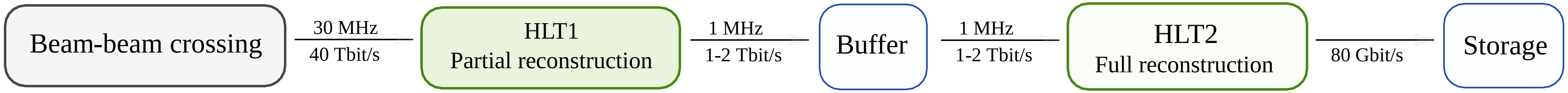
LHCb data flow in Run3

- LHCb Run 3 data flow

DPA project: Sprucing, AP, tools, ……

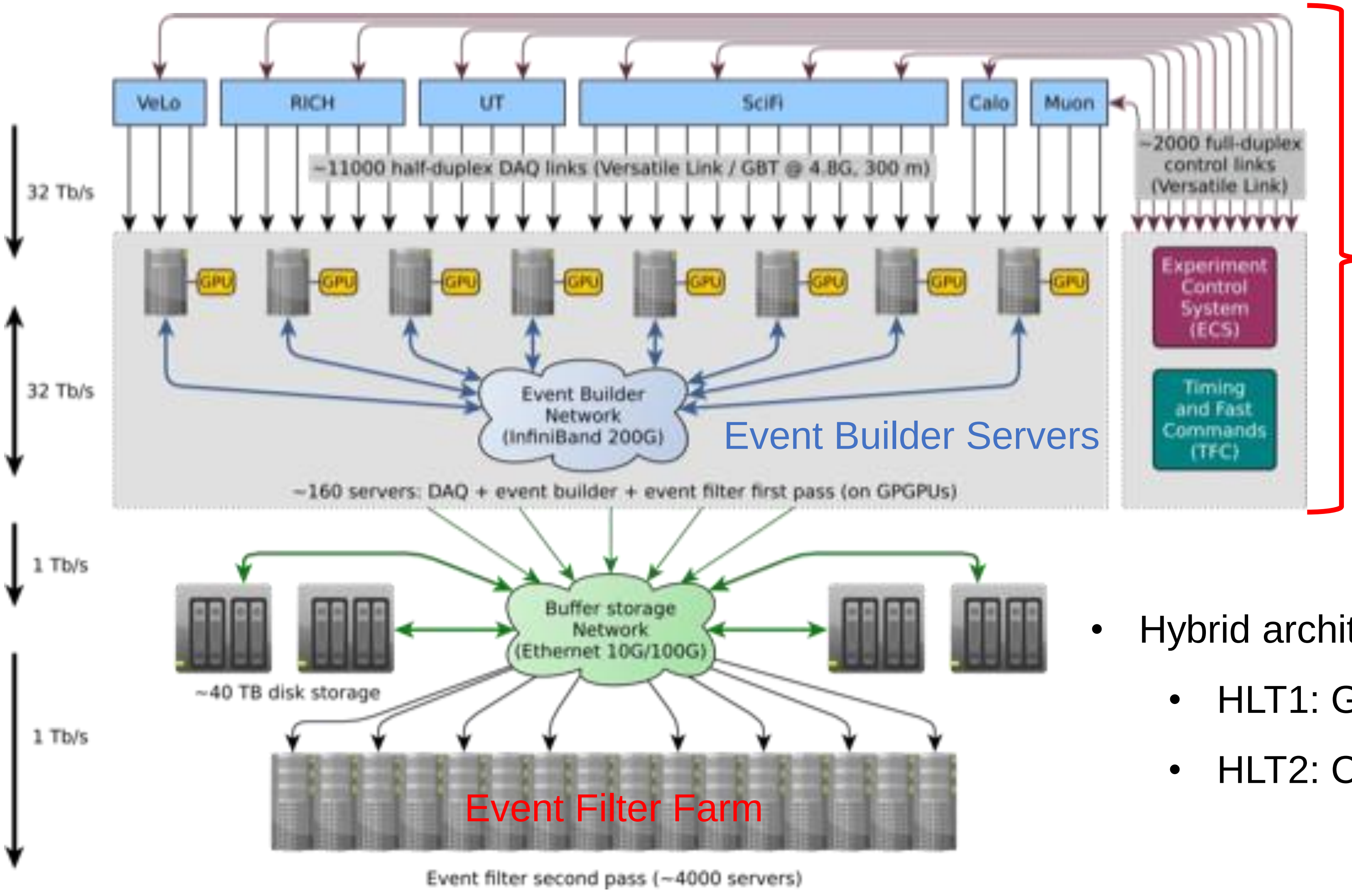


Triggers only in software:



Part 1
Online data processing:
HLT1 & HLT2

Hardware infrastructure of DAQ system



- Data acquisition system
  - Gather information from 1M electronic channels from the full detector
  - ~160 computer servers (equipe with 480 custom electronic cards)
  - Output rate to HLT1: 4~5 TB/s (30 MHz)

- Hybrid architecture:
  - HLT1: GPUs installed in Event Builder Servers
  - HLT2: CPUs in Event Filter Farm

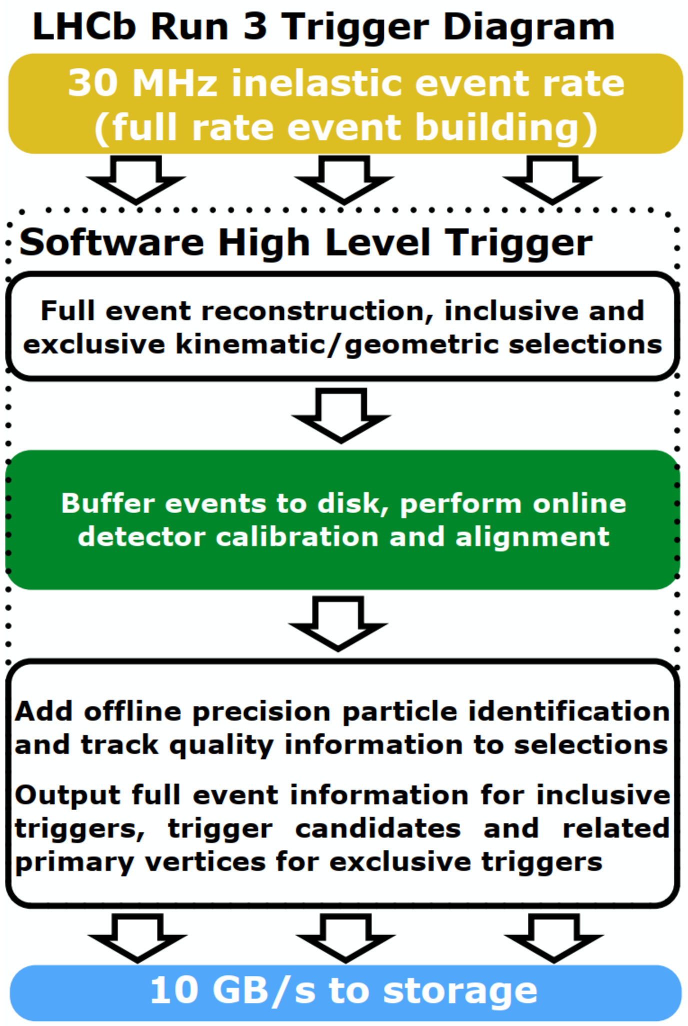
Full software triggers

- High Level Trigger 1 (HLT1):
  - Full charged particle track reconstruction
  - Few inclusive single and two-track selections
- High Level Trigger 2 (HLT2):
  - Aligned and calibrated detector
  - Offline-quality track reconstruction
  - Particle identification
  - Full track fit

Comparison to Run 2 trigger

- 5x higher pileup
- 30x higher rate into HLT1
- Disk buffer reduces from $\mathcal{O}$(weeks) $\rightarrow$ $\mathcal{O}$(days)
- Up to 10x efficiency improvement for some physics channels

Huge computing challenge



HLT1 reconstruction & selection sequence

Beam-beam crossing → 30 MHz / 40 Tbit/s → **HLT1 Partial reconstruction** → 1 MHz / 1-2 Tbit/s → Buffer → 1 MHz / 1-2 Tbit/s → HLT2 Full reconstruction → 80 Gbit/s → Storage

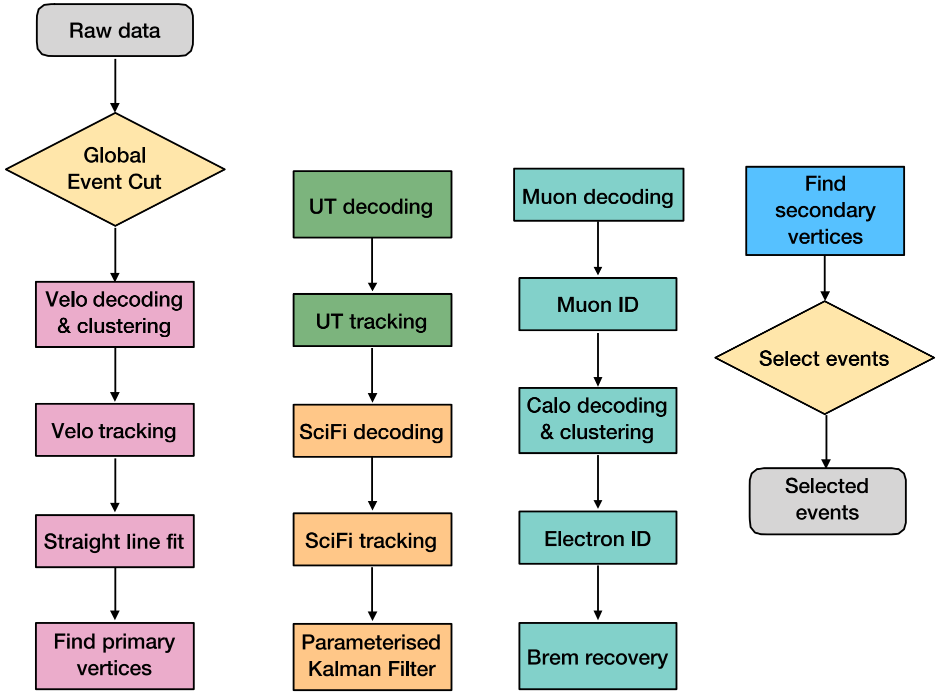
Reconstruction

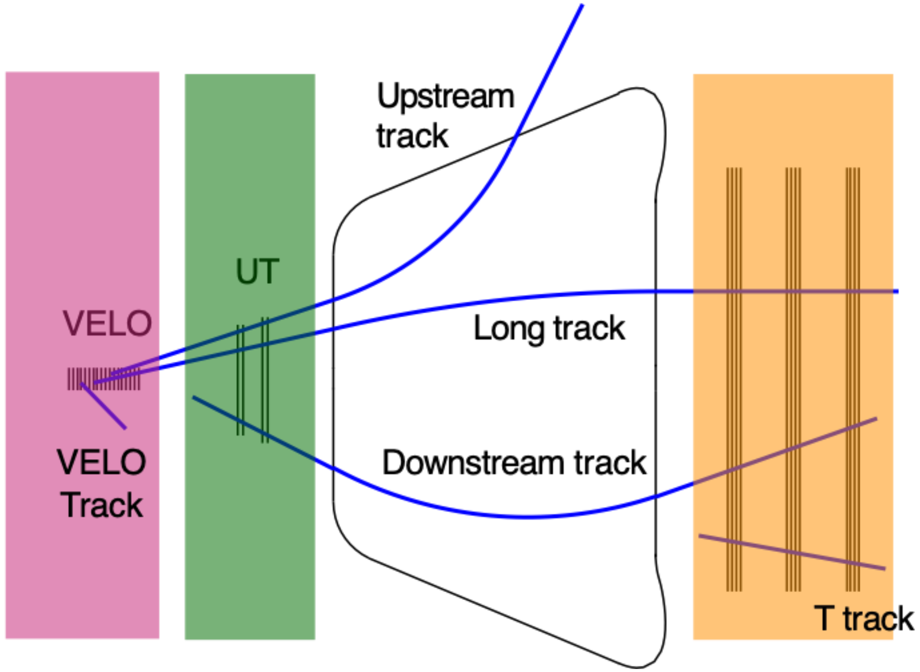
Tracking at the core of the HLT1 reconstruction

- VELO: clustering, tracking, vertex reconstruction
- UT: track reconstruction, momentum estimate, fake rejection
- SciFi: track reconstruction, momentum measurement

PID from muon stations & Calo

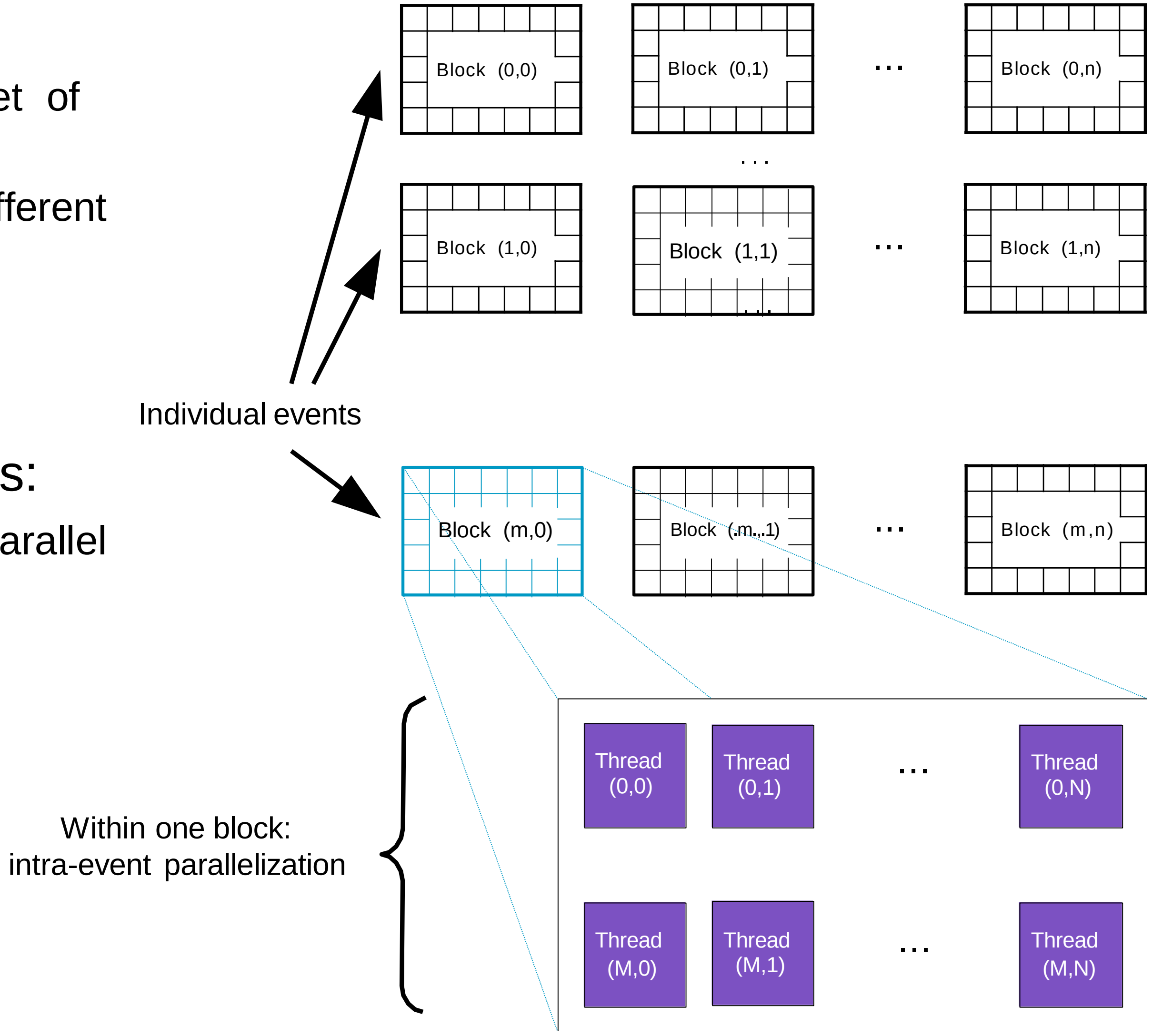
HLT1 lines → **Event selection**

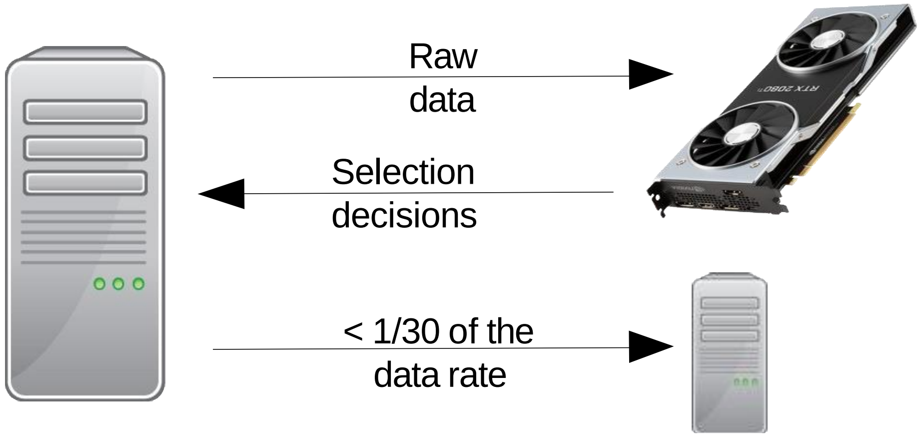




HLT1 on GPUs

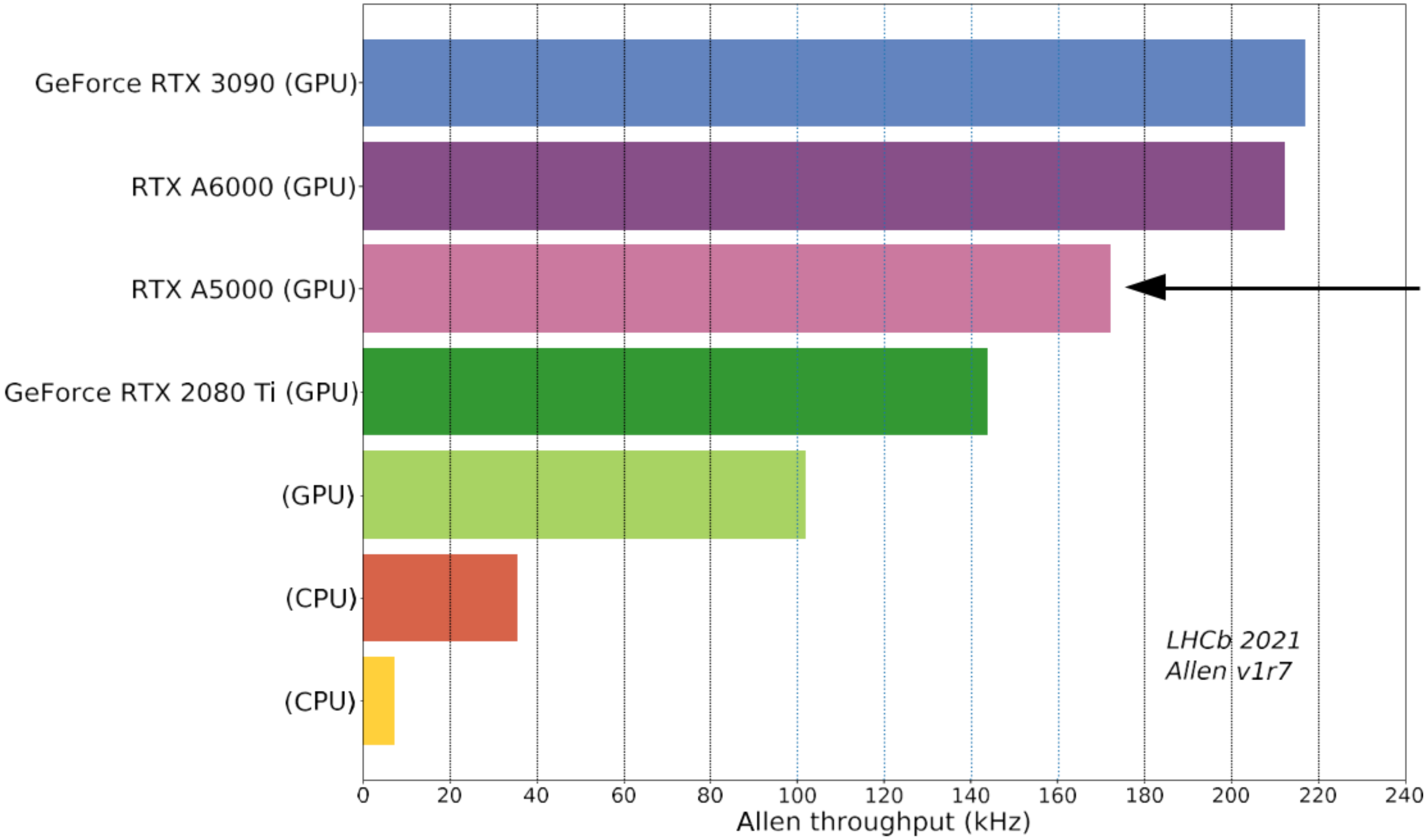
- GPU code is executed on many “threads”
  - Threads are organized in a “grid”, where a fixed set of threads is grouped into one “block”.
  - Each thread processes the same instructions, but on different data.
- Thousands of events are processed in parallel
- Only single precision is used
- Memory transfers are hidden behind calculations:
  - Several pipelines of HLT1 sequences are processed in parallel on “CUDA streams”





HLT1 computing throughput

- 30 MHz goal can be achieved with O(200) GPUs (maximum the Event Builder server can host is 500)
- Throughput scales well with theoretical TFLOPS of GPU card
- Additional functionalities are being explored



Chose RTX A5000 for the beginning of Run3

LHCb-FIGURE-2020-014

The Allen project

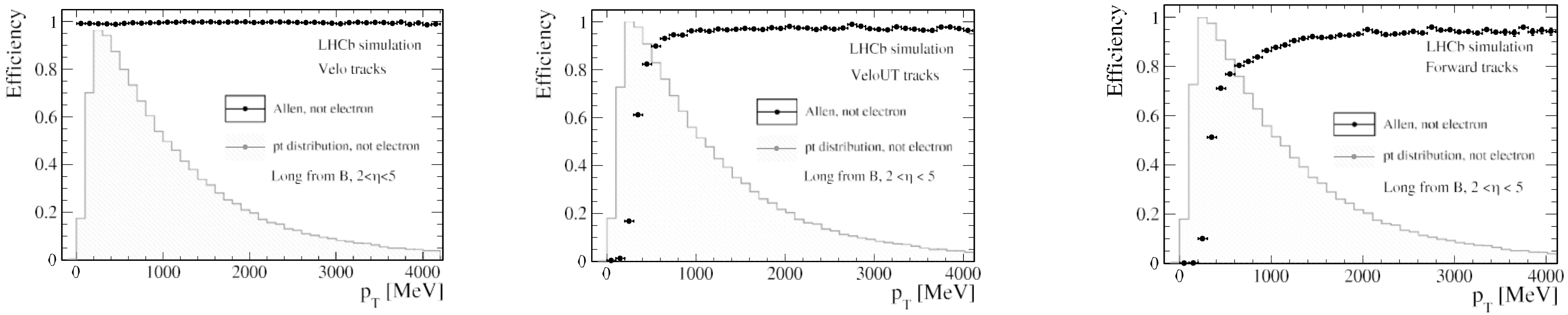
Frances Allen
1932~2020

- Named after *Frances E. Allen*
- Fully standalone software project:
  - https://gitlab.cern.ch/lhcb/Allen
- Framework developed for processing HLT1 on GPUs
- Cross-architecture compatibility via macros & few coding guide lines
  - GPU code written in CUDA, runs on CPU, Nvidia GPUs (CUDA), AMD GPUs (HIP)
- Algorithm sequences defined in python and generated at run-time for multi-event processing
  - Sequence: algorithms to run based on required inputs & properties
- Memory manager
  - Large chunk of GPU memory allocated at start-up, pointers within this chunk assigned by memory manager

HLT1 tracking performance

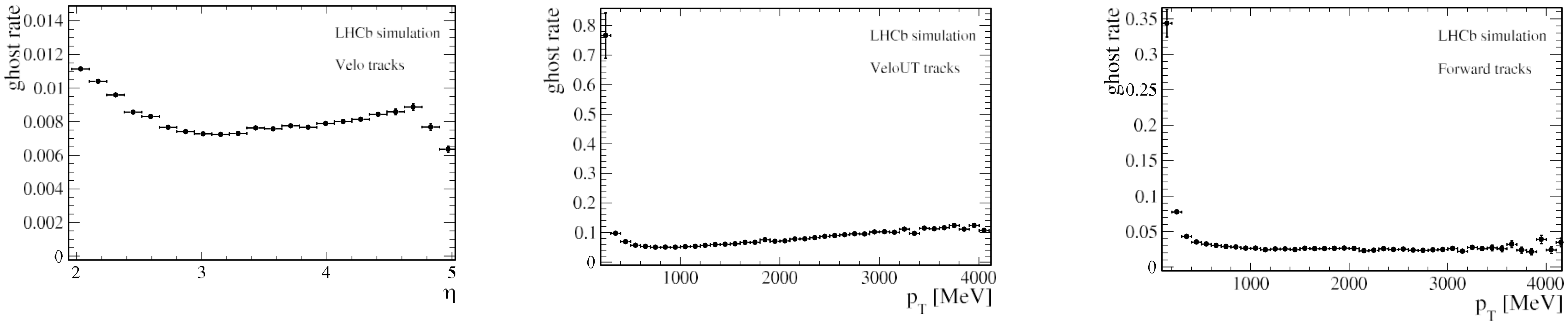
- Run 2 performance maintained at x5 instantaneous luminosity
- Excellent track reconstruction efficiency (> 99% for VELO, 95% for high-p forward tracks)
- Good momentum resolution and fake rejection

Track reconstruction efficiency



Fake rate

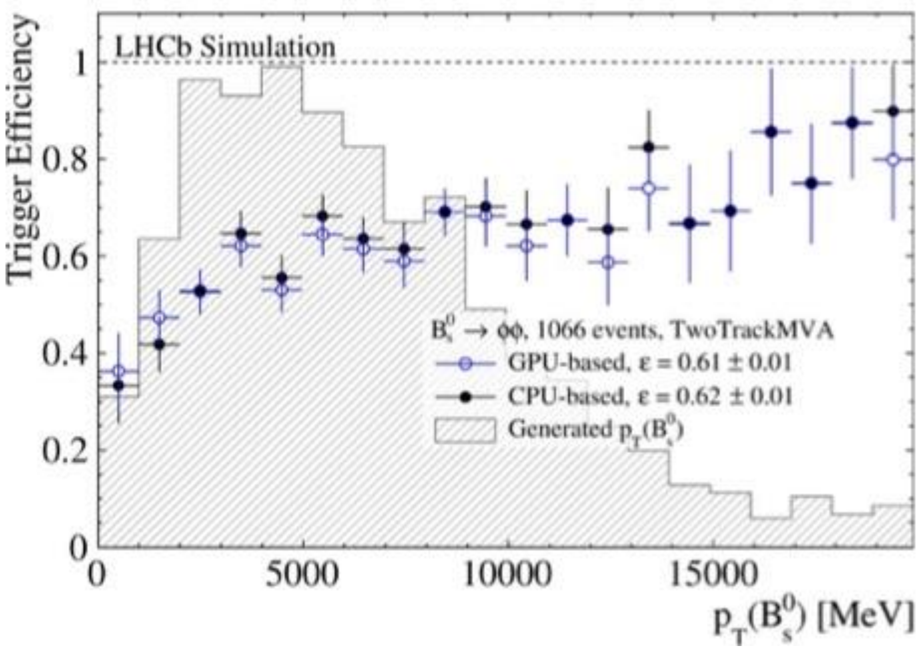
LHCb-FIGURE-2020-014



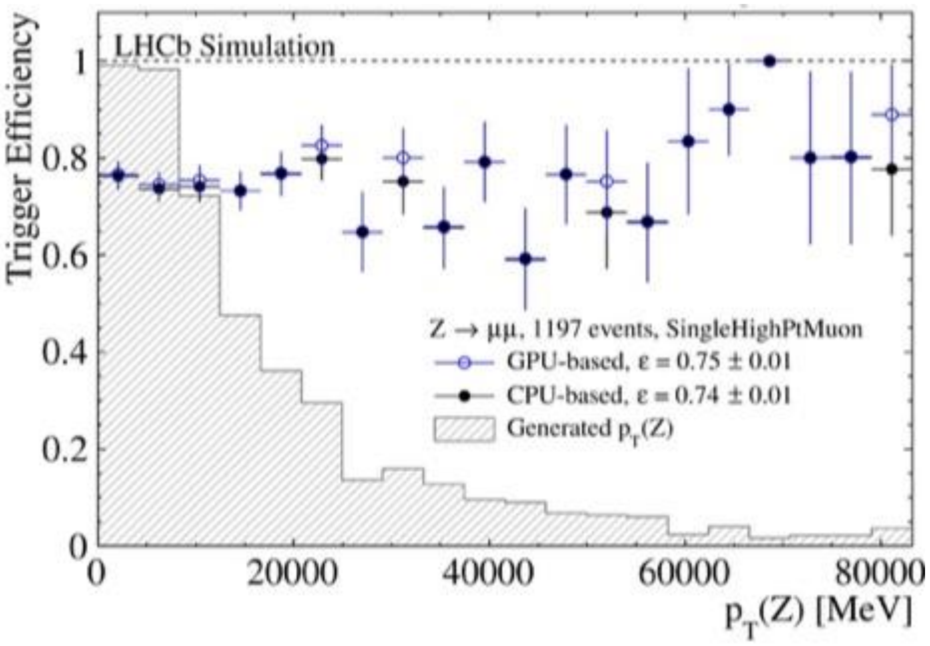
HLT1 selection performance

- Inclusive rate for the main HLT1 lines ~ 1 MHz
- O(30) lines implemented so far:
  - Cover majority of LHCb physics program (*B*, *D* decays, semileptonic, EW physics)
  - Special lines for monitoring, alignment and calibration
  - Additional trigger lines under development

TwoTrackMVA line for $B_s^0 \rightarrow \phi\phi$

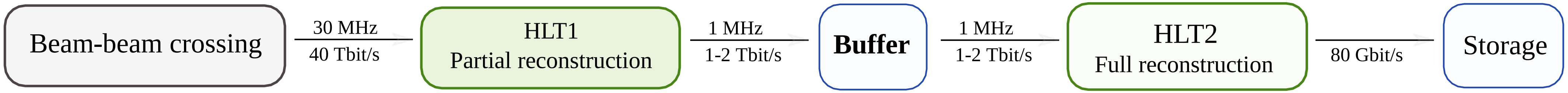


SingleHighPtMuon line for $Z \rightarrow \mu^+\mu^-$



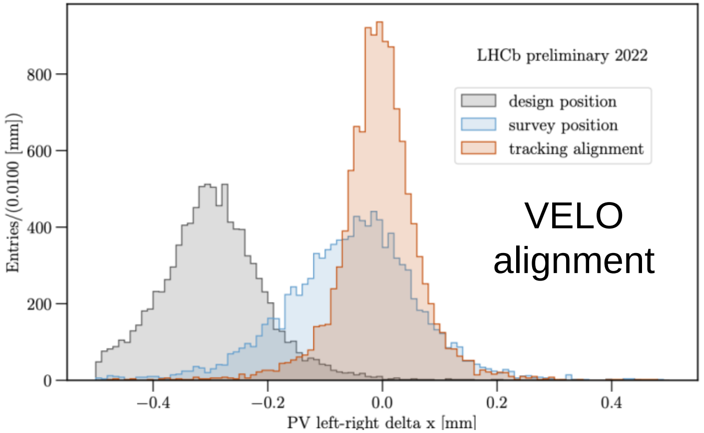
Comput. Softw. Big Sci. 6 (2022) no.1, 1

Online alignment & calibration

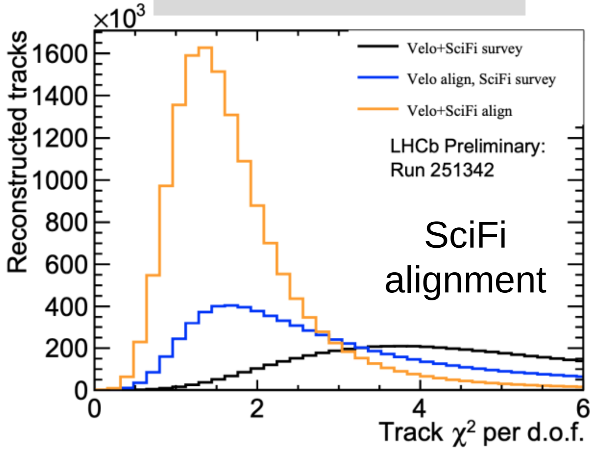


- Efficient and pure selections require offline-quality reconstruction at the HLT2 level

LHCb-FIGURE-2022-016

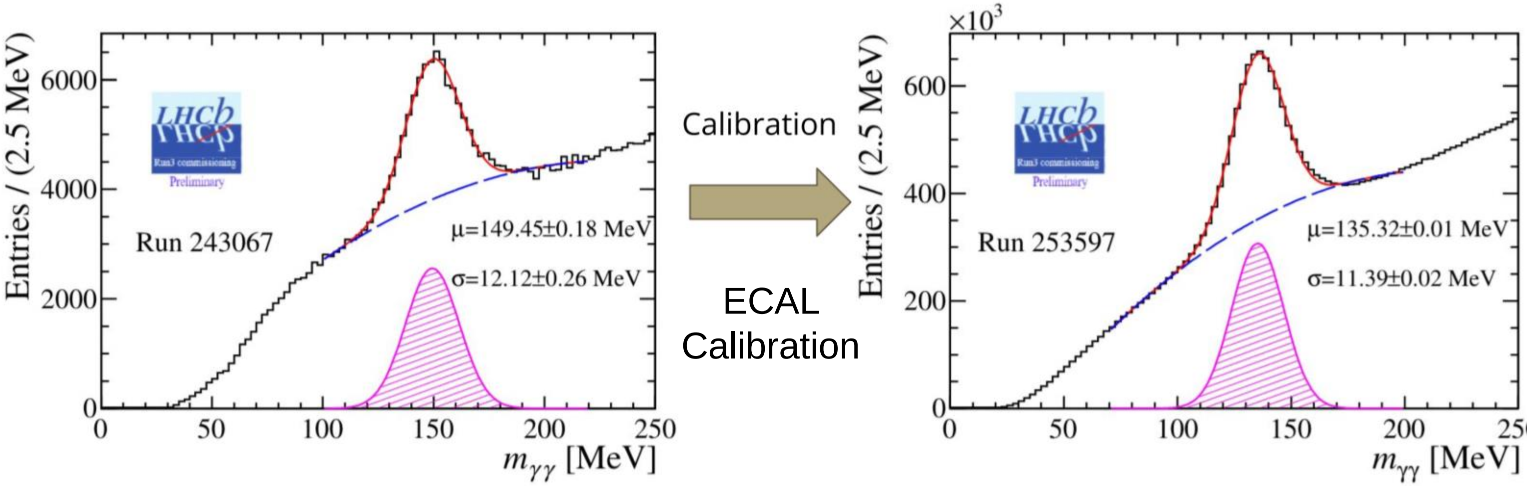


LHCb-FIGURE-2022-018



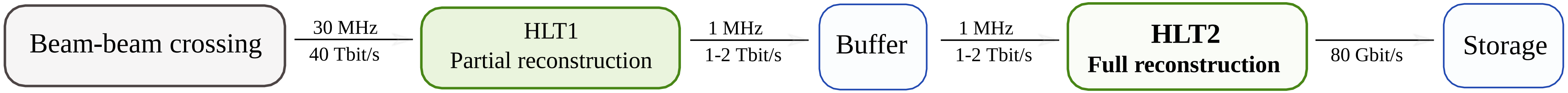
- Run alignment & calibration before HLT2
  - Better mass resolution
  - Better track quality
  - Less background

→ use output bandwidth more efficiently

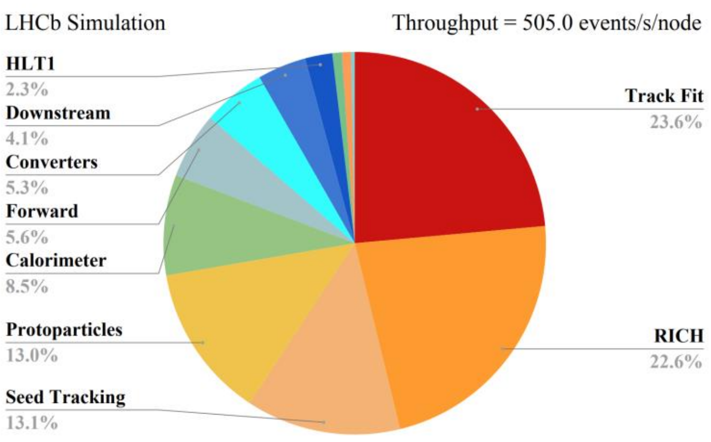


LHCb-FIGURE-2022-019

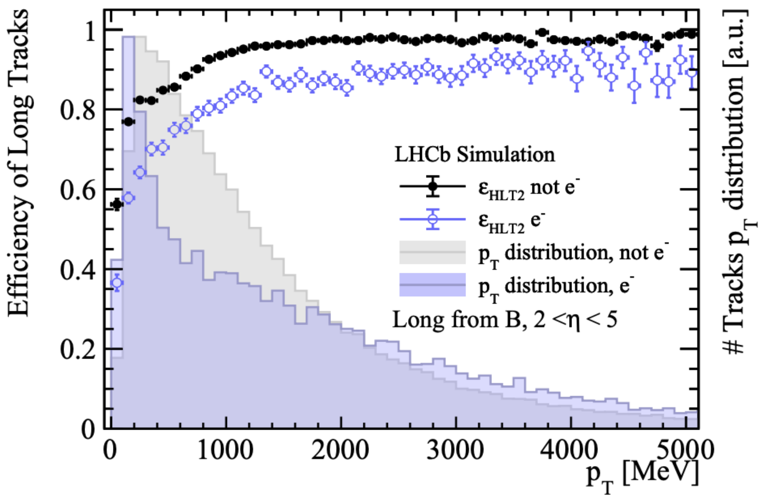
HLT2 on CPUs



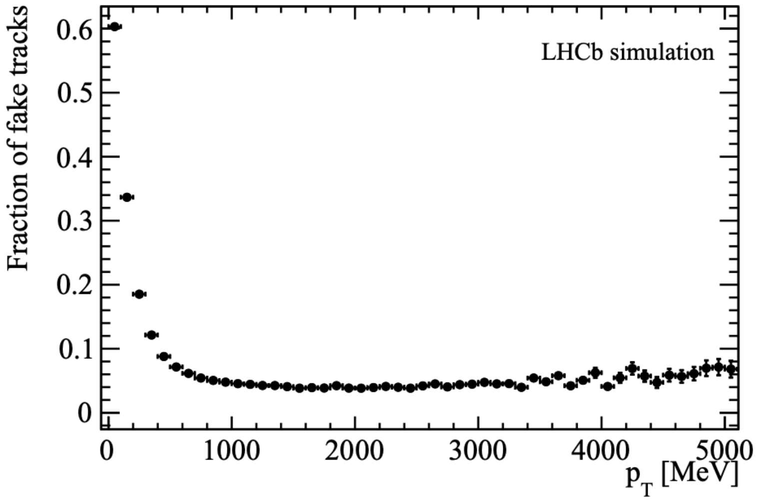
- Fully aligned & calibrated detector, offline quality track fit & particle identification @ 1MHz
- HLT2 throughput significantly improved over last years
- Hundreds of exclusive selections being written for specific analyses, using new multi-threaded framework



HLT2 computing throughput



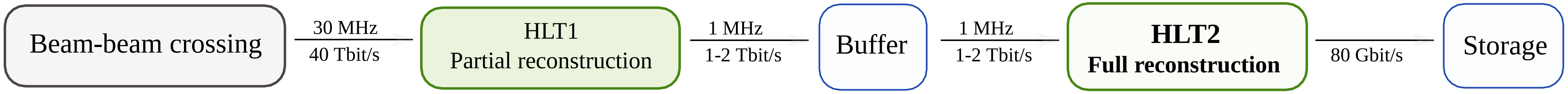
HLT2 performance: tracking efficiency

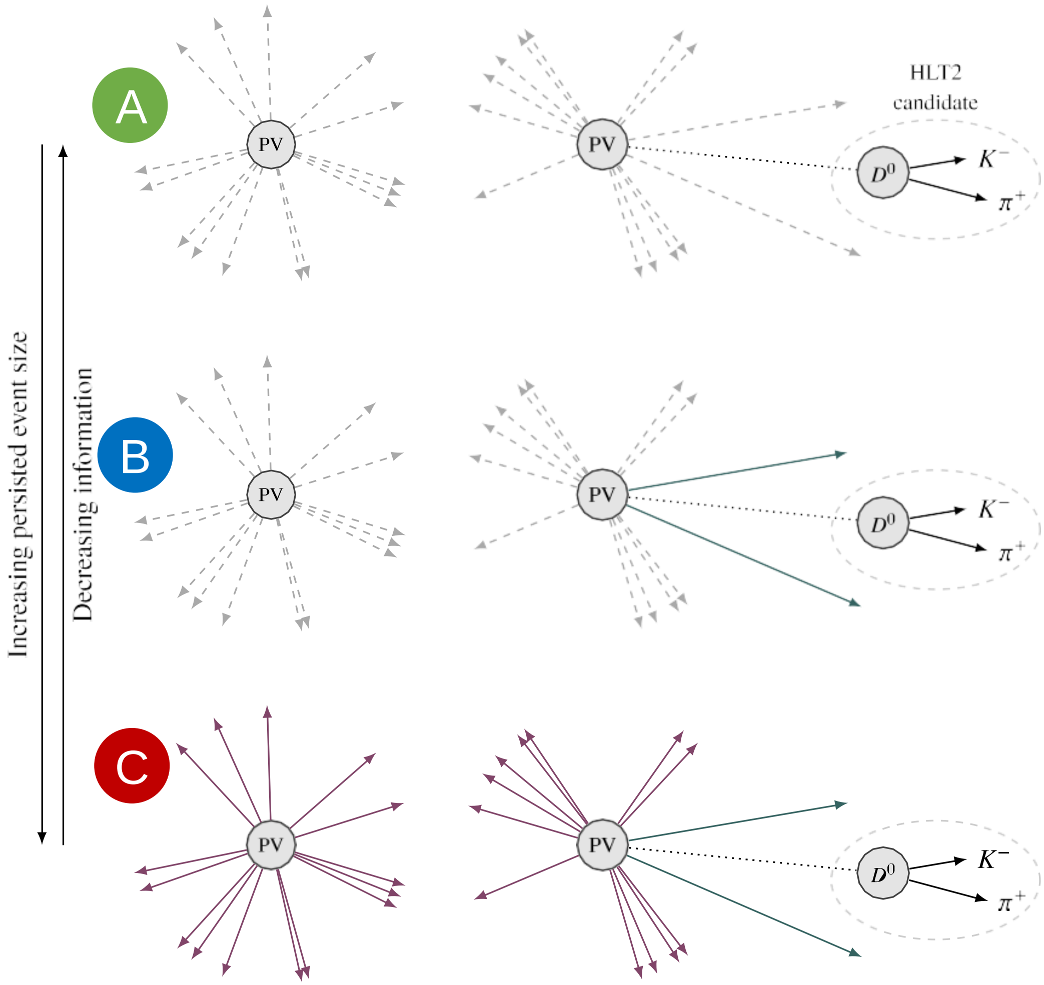


HLT2 performance: tracking fake rate

LHCb-FIGURE-2022-005

HLT2: Selective persistency (Turbo stream)





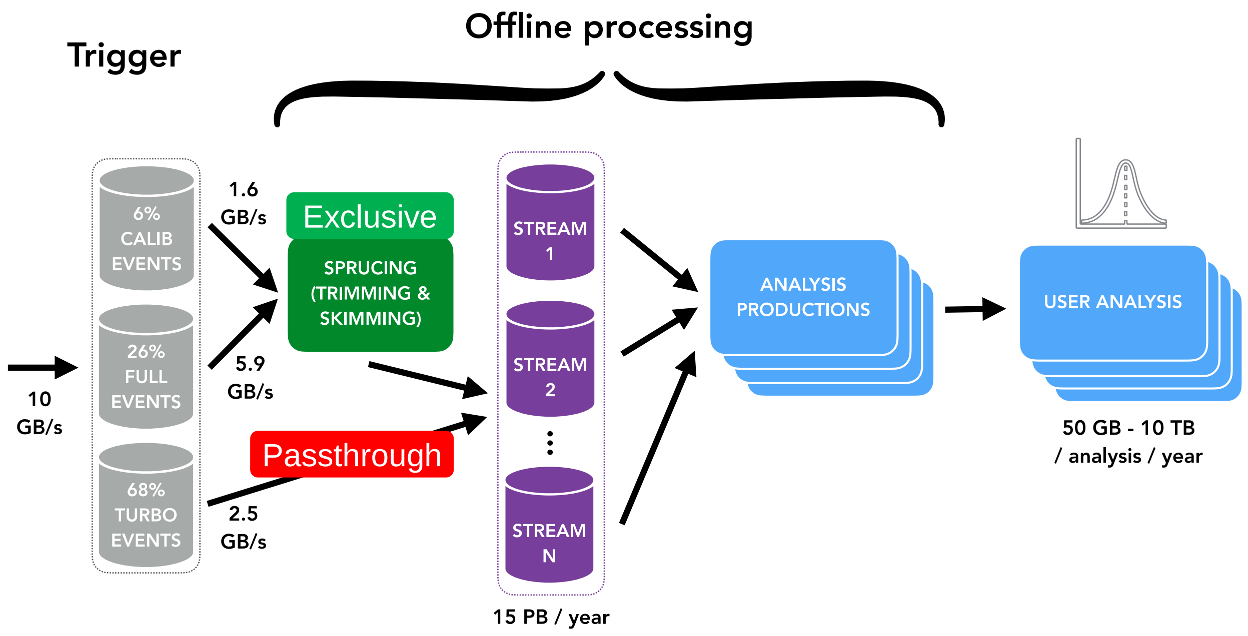
- Trigger bandwidth is limited
  - Huge signal rate → reduce event size
- Only store high-level objects reconstructed in real-time
- High degree of flexibility:
  - A Only objects used in trigger selection
  - B Objects used in trigger selection & user-defined selection
  - C All reconstructed objects
- Raw data only stored in calibration stream

JINST 14 (2019) P04006

Part 2:
Offline data processing:
Sprucing, AP, user analysis

Sprucing

- Centralized offline data processing, selections and streaming that runs on the output of HLT2 in Run 3 and beyond. The Sprucing runs in two forms:
  - Passthrough is for the HLT2 TURBO stream. The Sprucing serves a similar purpose as Tesla of Run 2, changing the file format from MDF to DST and creating summary records for luminosity information.
  - Exclusive is for the HLT2 FULL stream. This data is too “big” (in terms of bandwidth) to go straight to disk and so a second set of physics selections are run to **S**lim and **PRUNE** the data.



The exclusive Sprucing lines are used when

(A) **all of the following 4 conditions are met.**

1. Inclusive HLT2 triggers on “interesting” events
2. Full reconstruction of triggered events saved to tape
3. The line re-analyses (trimming & skimming) the events offline
4. Output saved to disk and available to analysts

Or (B) **for running intensive data selection or processing algorithms offline**.

Analysis productions

- The old/legacy way: User directly submit jobs to LHCbDIRAC
  - They are imperative jobs where each one has exactly specified input data and cannot be adjusted to adapt to current grid conditions.
  - Usually thousands of jobs, affected by site downtimes and infrastructure instabilities.
- The new/modern way: Analysis Productions (AP)
  - Centrally manage the processing of LHCb data and simulation in a coordinated manner
  - Submission via YAML: Essential details such as input data bookkeeping query, job configuration, …
  - Automatically handling failures and adapting file grouping strategies
  - Comprehensive pipeline tests on the GitLab CI platform to ensure that job configurations are valid before approval, minimizing waste of computing resources.
  - Better output data accessibility and convenient analysis preservation

```yaml
defaults:
  application: DaVinci/v45r4
  wg: WG
  automatically_configure: yes
  turbo: no
  inform:
    - someone@cern.ch
  options:
    - make_ntuple.py
  output: DVNtuple.root

My_MagUp_job:
  input:
    bk_query: /some/MagUp/bookkeeping/path.DST
    n_test_lfns: 3  # only to be used in special cases

My_MagDown_job:
  input:
    bk_query: /some/MagDown/bookkeeping/path.DST
```

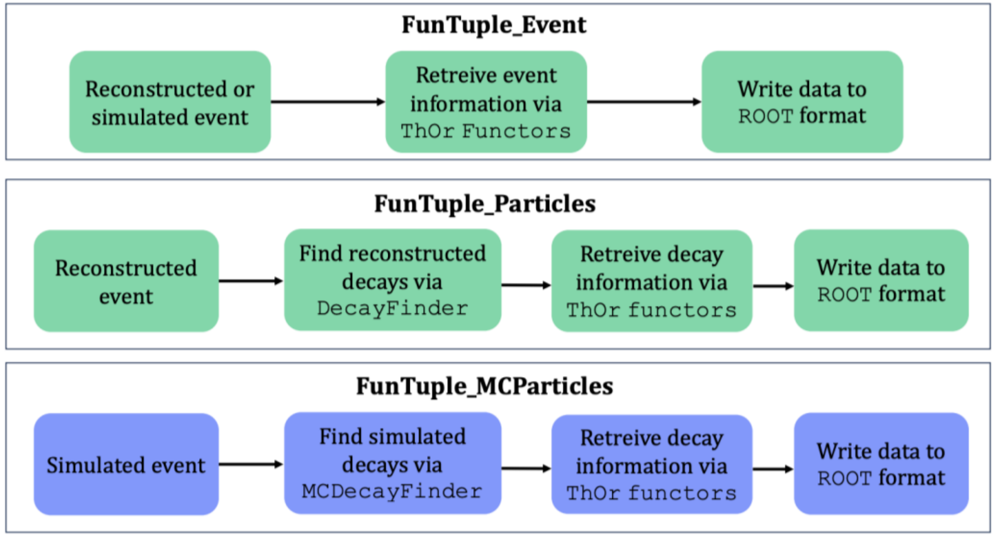
Analysis software

- DaVinci: Application for processing and tuple making via AP for further analysis.
- Tools for nTuple making: `C++` class built upon the Gaudi functional framework, and offers a user-friendly `Python` interface.

`DecayTreeTuple` (Run1&Run2) → `FunTuple` (Run3 & beyond)

Array of Structures (AoS) → Structure of Arrays (SoA)

- Many information allowed to be added:
  - Trigger info, ThOr functors, DTF, MC truth info, ….



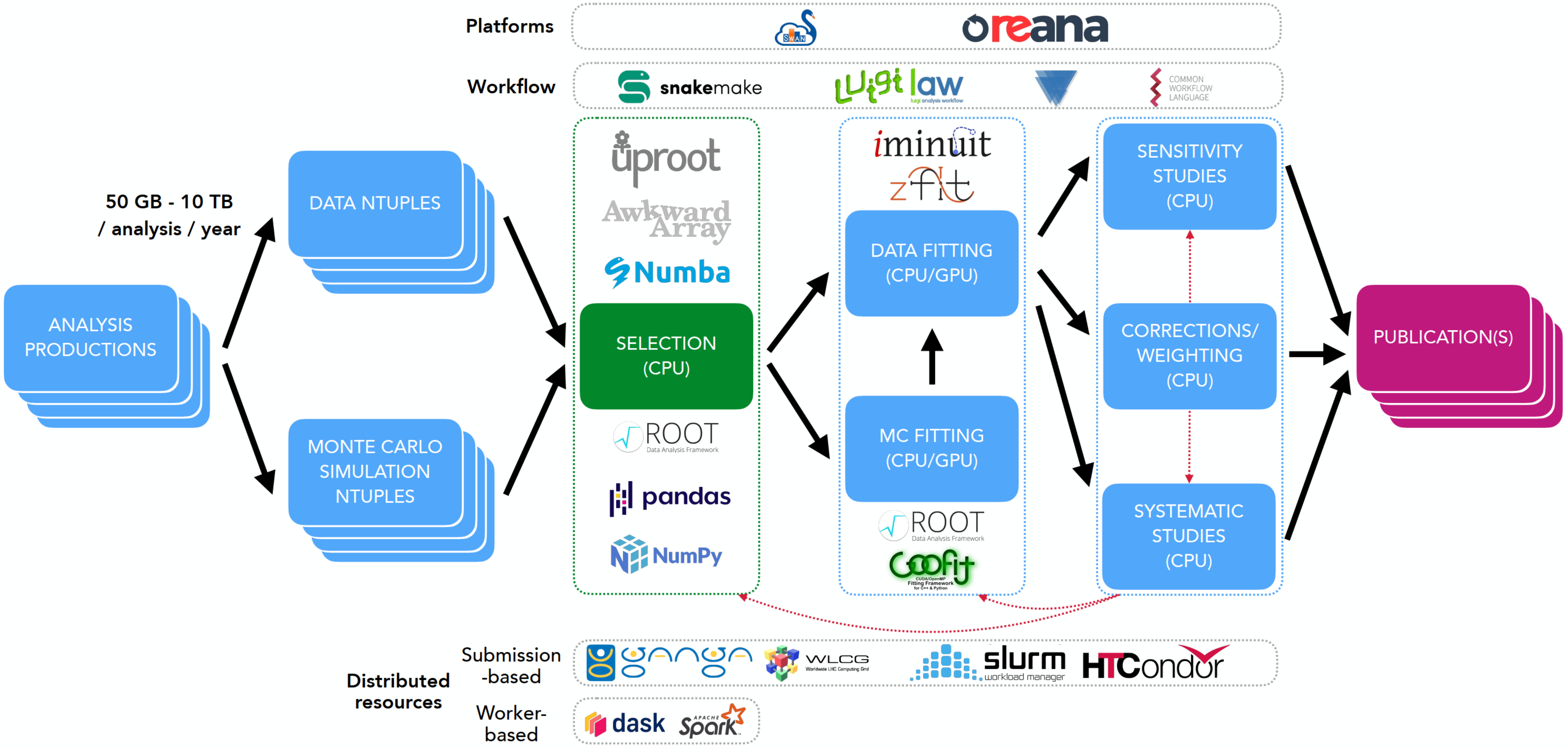
Data flow of the three flavours of FunTuple component

Comput.Softw.Big Sci. 8 (2024) 6

```python
from DaVinci import Options, make_config
from DaVinci.algorithms import create_lines_filter
from PyConf.Algorithms import PrintDecayTree
from PyConf.reading import get_particles


def print_decay_tree(options: Options):
    turbo_line = "Hlt2BsToJpsiPhi_JPsi2MuMu_PhiToKK_Line"
    input_data = get_particles(f"/Event/HLT2/{turbo_line}/Particles")

    user_algorithms = [
        create_lines_filter("HDRFilter_SeeNoEvil", lines =[ f"{turbo_line}"]),
        PrintDecayTree(name="PrintBsToJpsiPhi", Input=input_data)
    ]

    return make_config(options, user_algorithms)
```

A minimal demo of Run3 DaVinci script

User analysis



Conclusion

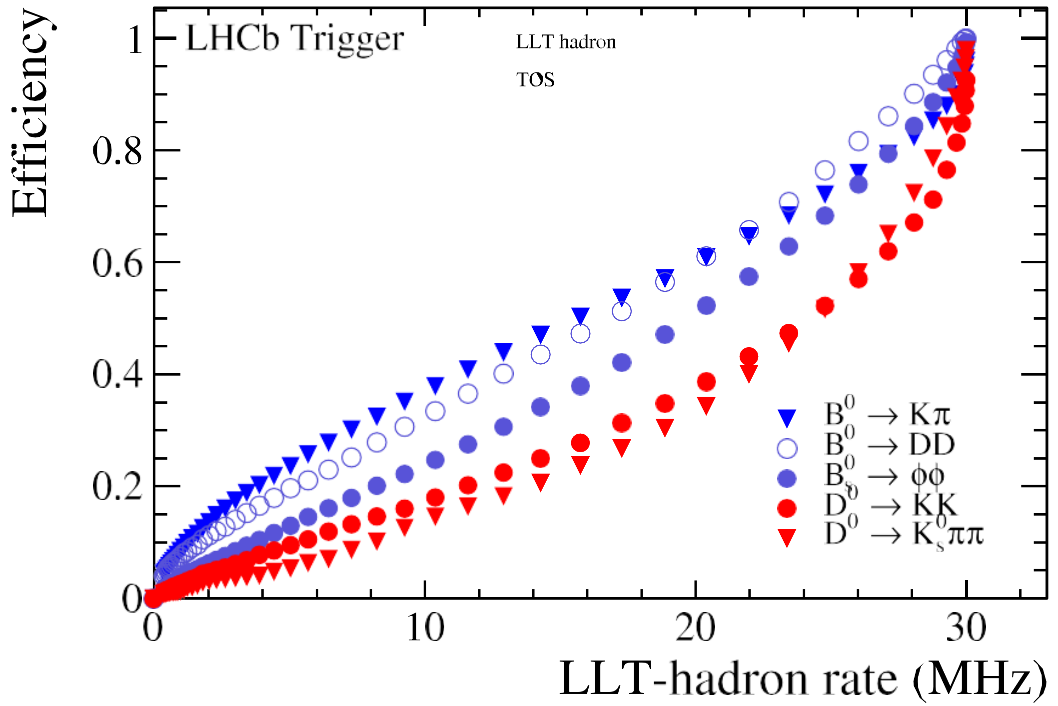
- Online:
  - Full software-based high-level trigger for LHCb Run3
  - GPU HLT1 project "Allen" is cross-platform and uncoupled for tracking, PV finding & muon ID
  - Real-time feedback for online alignment & calibration
  - HLT2 designed for selecting dedicated physical events with high efficiency and low bandwidth
  - Different levels of persistency is configured to maintain a better balance of bandwidth and physical information.
- Offline:
  - Centralized flexible trimming & skimming framework
  - Selection framework with C++ class as core and Python-based user-friendly API
  - SoA-based data structure for fast event-looping and feature-calculation
  - Centralized analysis production (AP) service with automatic I/O handling and CI test
  - Comprehensive and lightweight data processing tools for offline user analysis for various purposes

Thanks for listening!

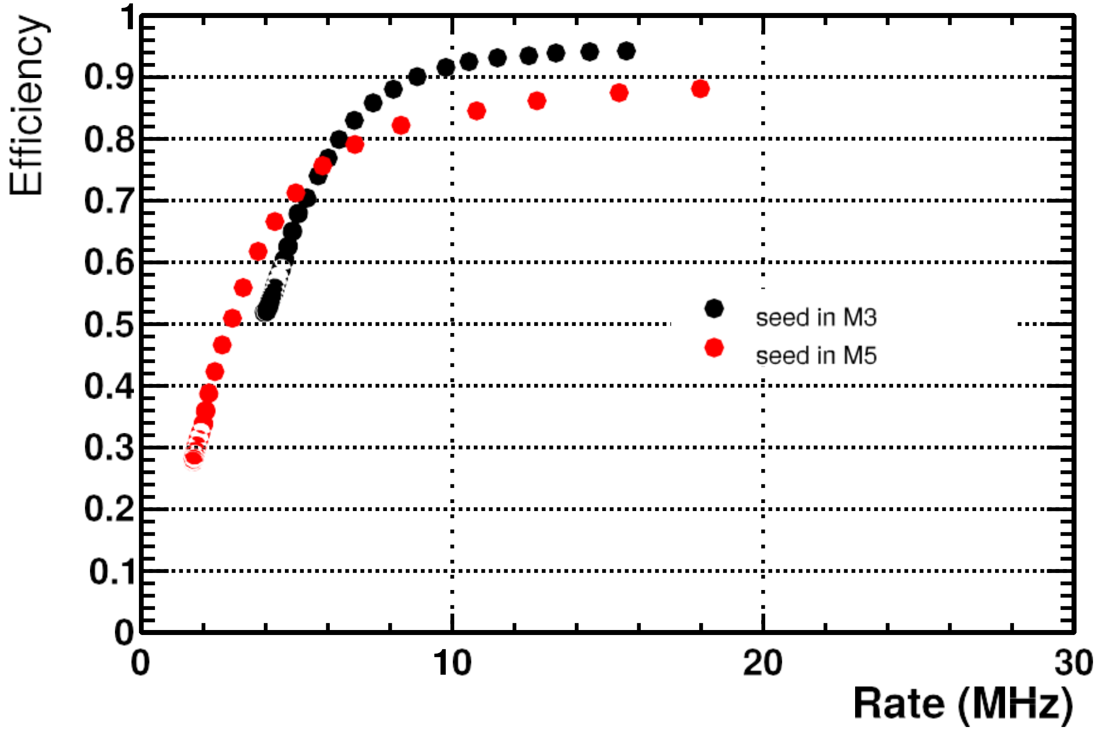
Backup

Why no low level trigger?

Low level trigger on E_T from the calorimeter



Low level trigger on muon p_T, $B \to K^*\mu\mu$

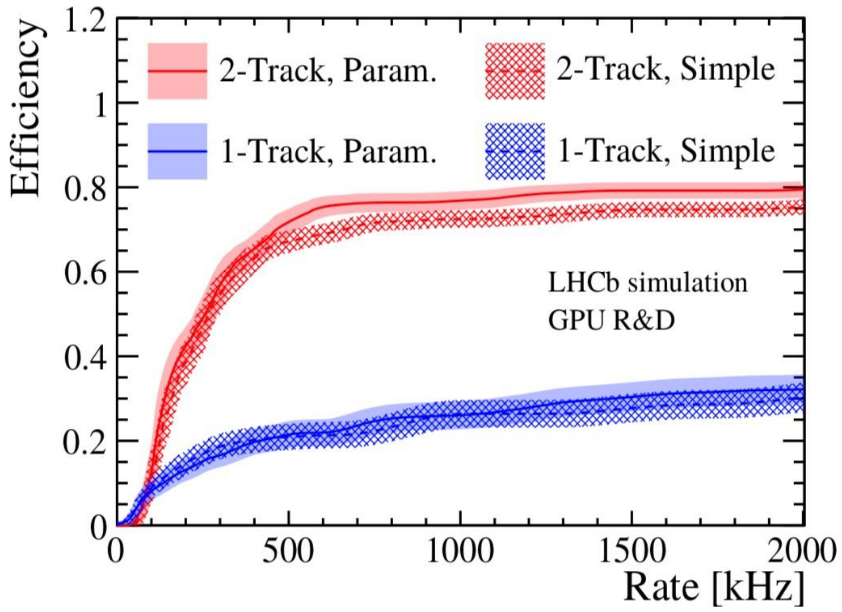


Kalman filter

- Improve Impact Parameter (IP) resolution and reduce ghosts
- Nominal LHCb Kalman filter uses Runge - Kutta extrapolator + detailed detector description

- In HLT1, for performance reason two alternatives based on parametrizations:
  - Full detector Parametrized Kalman Filter
  - Velo-Only Kalman Filter (fits only Velo segment, momentum estimate from full track)
  - IP resolution mostly impacted by Velo measurement

  -> Velo-Only option chosen, which significantly improves throughput



The track matching algorithm

- Two main inputs: **SciFi** and **VELO** seeds
- Algorithm approach
  - "Kink" approximation: Velo/SciFi seeds extrapolated to matching position as straight lines
  - Magnetic field and bending in y parametrised with truth simulation to calculate z_match(x,y)