The first-level full-software GPU-optimised trigger at LHCb

Alessandro Scarabotto

21st April 2026

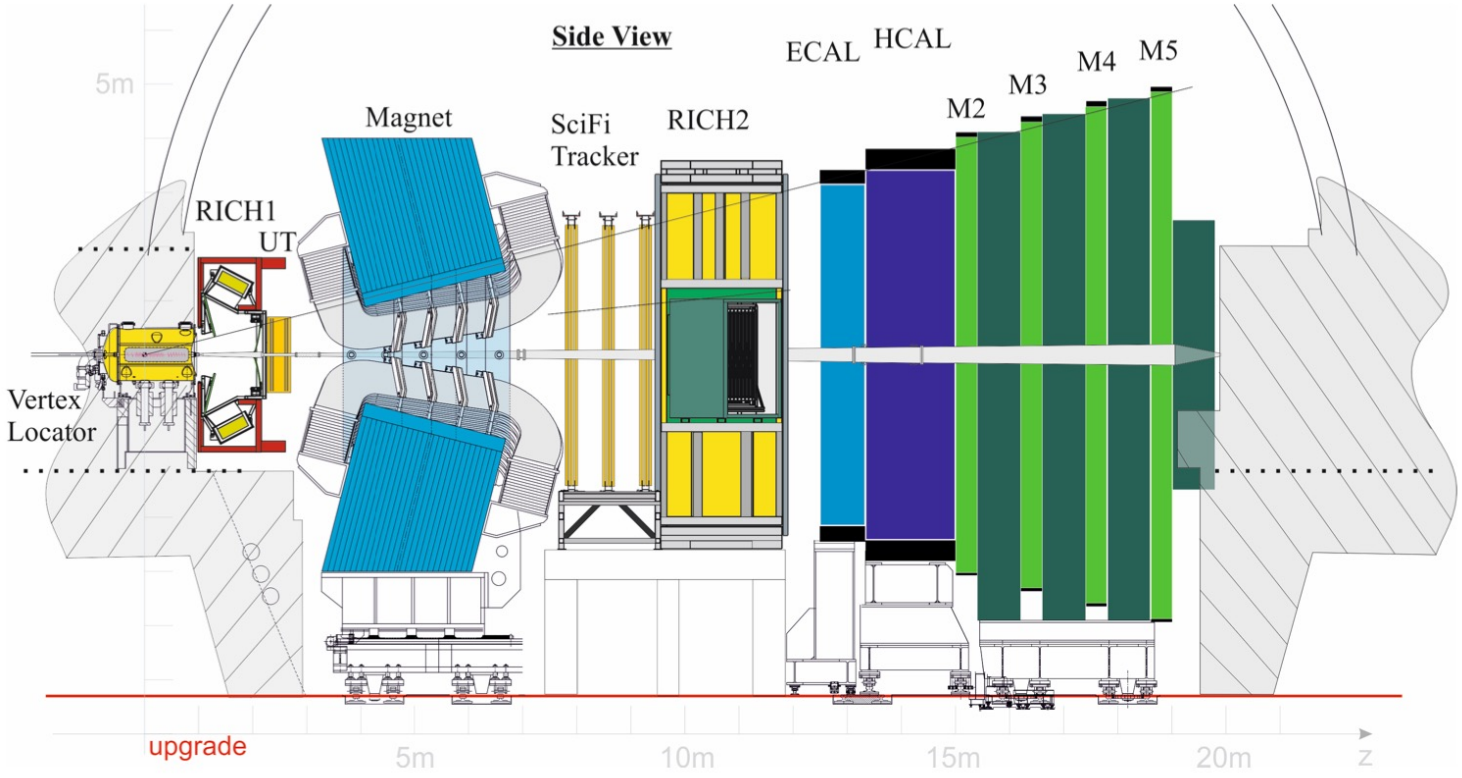
Ruhr Hadron seminar

Bochum, Germany

The LHCb experiment

- Forward spectrometer on the LHC ring designed to study flavour physics exploiting proton-proton collisions
- Main focus on beauty and charm decays → displaced signatures
- Excellent vertexing and momentum resolution (< 1%)
- Particle identification, separating pion-kaon-proton-muon-electron

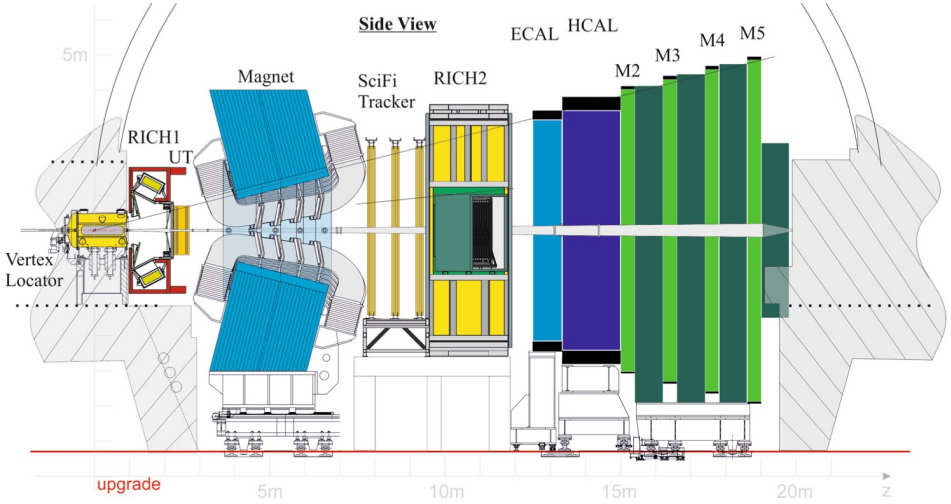
LHCb-TDR-12

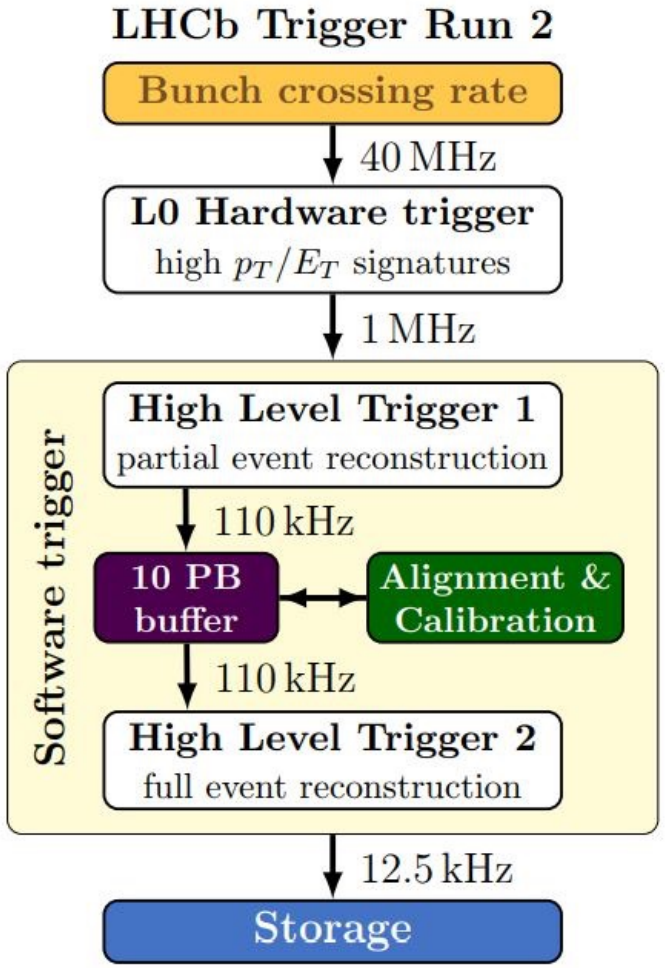


- LHC is colliding bunches of protons every 25 ns (40 MHz) → **triggering is crucial**

The LHCb experiment: trigger system

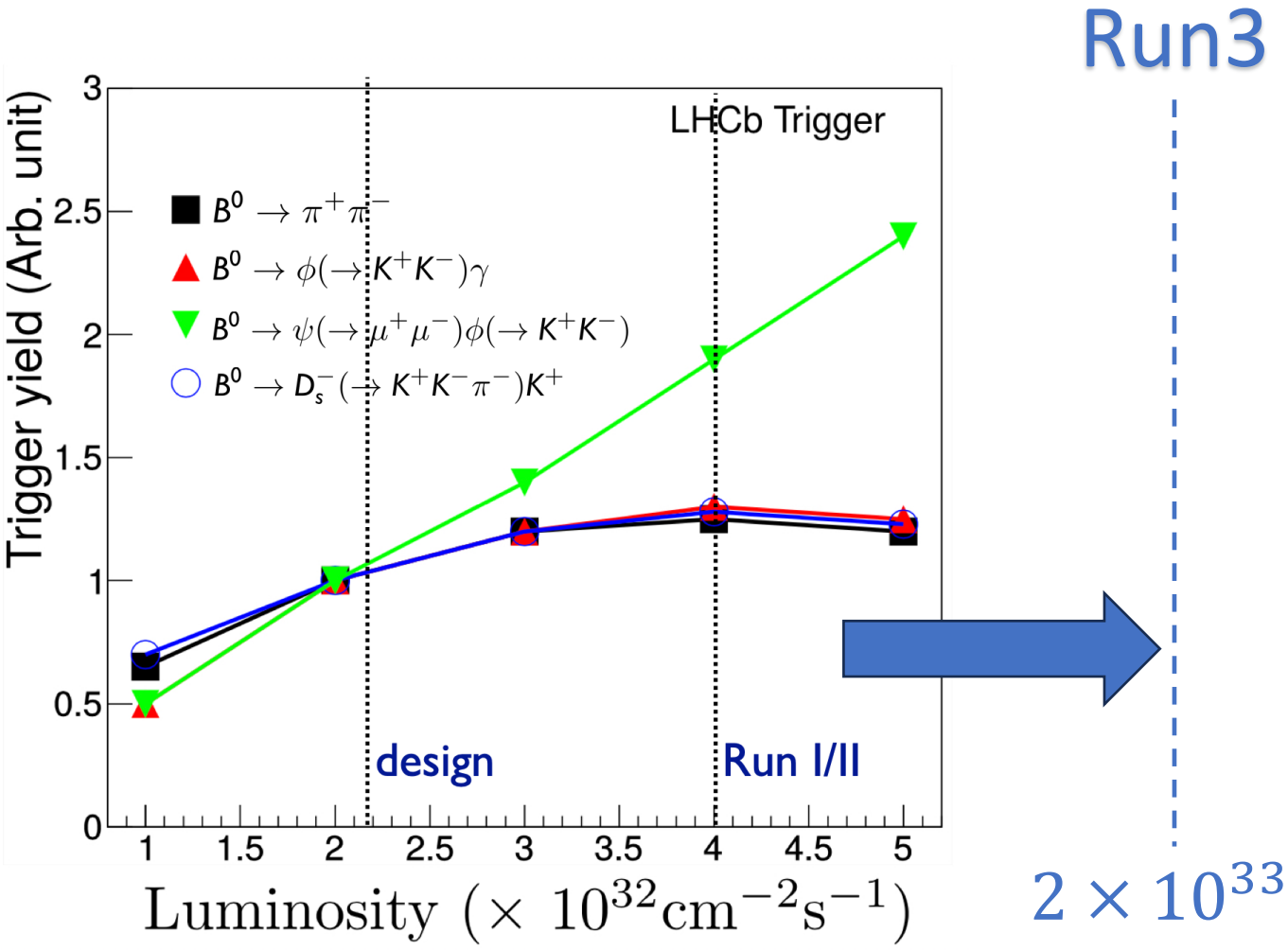
- Run 1-2 (2011-2018) took data with a hardware (L0) and software (HLT) trigger system and collected 9 fb^{-1} of data
- In order to increase the physics output, in Run3 (2022-2026) increasing the instantaneous luminosity by a factor 5 → reaching $\sim 30\ \text{fb}^{-1}$ of data in 4 years





The LHCb Run3 trigger

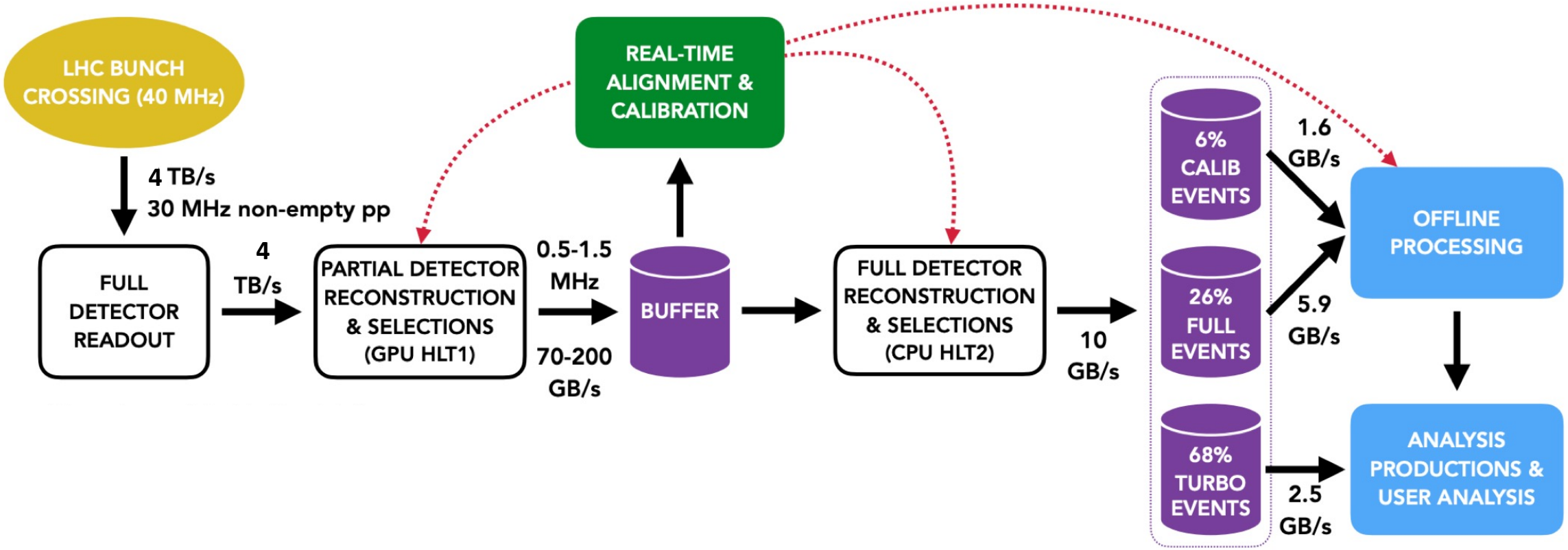
- Limitation of Run2 trigger is the first-level hardware stage (L0)
- Saturation of trigger yields by increasing luminosity
- Caused by tight momentum/energy requirements at L0 selections
- The Run3 LHCb:
  - Removal of L0! But …
  - Reconstruction at 30 MHz LHC pp collision rate for the High Level Trigger (HLT)



J. Phys.: Conf. Ser. **878** 012012

The LHCb dataflow

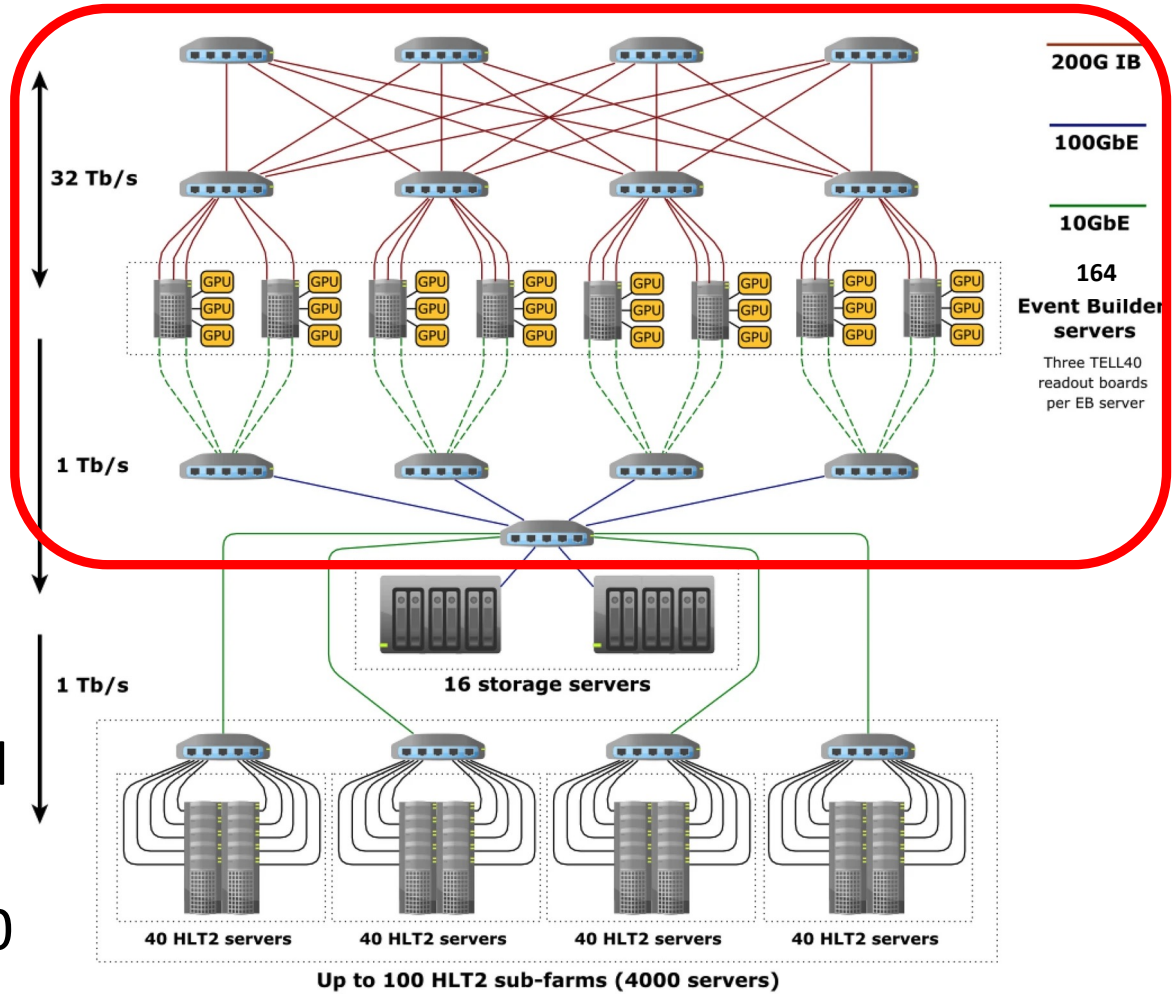
Developed full-software trigger based on a heterogeneous system (CPU-GPU)



LHCB-FIGURE-2020-016

The LHCb GPU farm

- Part of the LHCb heterogenous computing system
- Raw detector info is sent to the data processing center
- FPGA cards receive data at average 4 TB/s
- 164 Event Builder (EB) servers produce the packets of events
- Each EB server has 3 PCIe slots available for GPUs and where **HLT1** is run (zero overhead costs)
- 3 GPUs per EB server → ~ 500 Nvidia A5000



[Comput.Softw.Big Sci. 6 (2022) 1, 1]

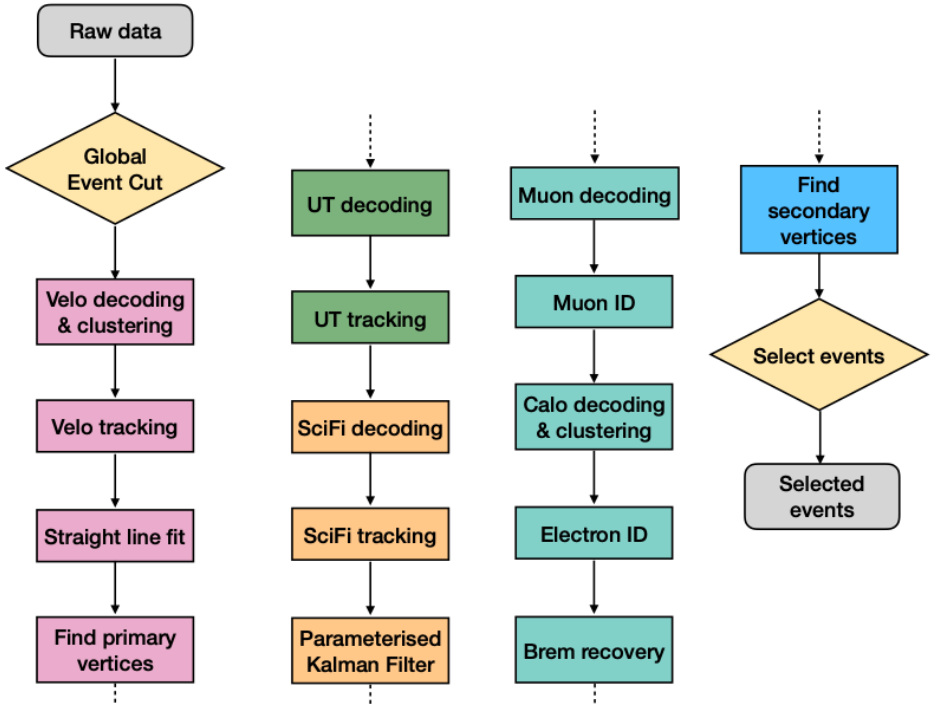
The HLT1 sequence

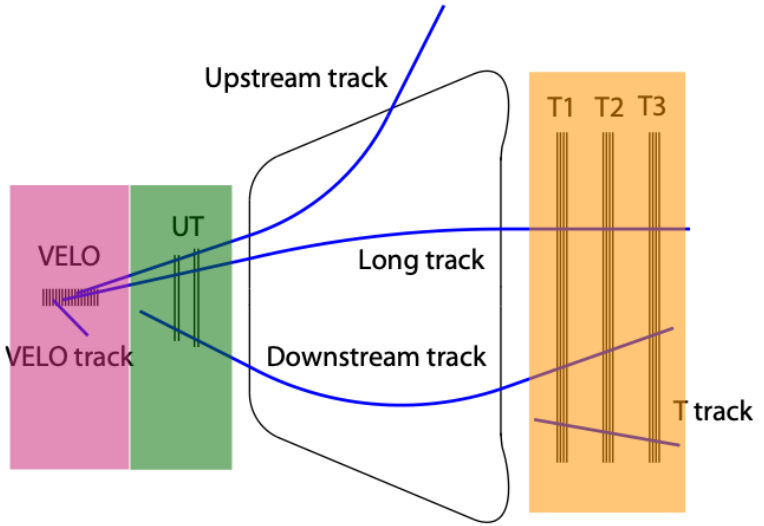
Tracking relies on

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

+

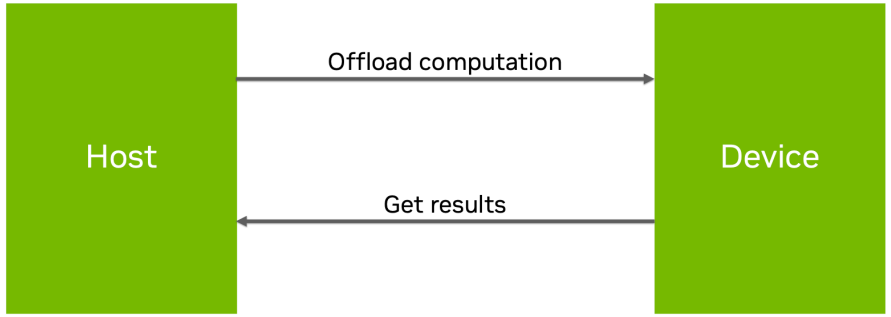
PID from **MUON** and **CALO** systems





The Allen project

- The project: https://gitlab.cern.ch/lhcb/Allen
- Named after Frances E. Allen
- Important! Cross-architecture compatibility:
  - GPU (CUDA and HIP) for validation and data taking
  - CPU for simulation production (running single-threaded per stream)
- Handles the workload and transfer of information from the CPU (host) and GPU (device)



Why GPUs?

- HLT1 tasks: perform event reconstruction at data rate of 4 TB/s and reduce it of a factor ~40 reaching 100 GB/s through trigger selections

The LHCb trigger	GPUs
Huge data load	High throughput Many FLOPS
Parallel problems: pp collisions (event) within one event: tracks and hits	Highly parallelizible
Small raw event data (~100KB)	PCIe connection: high throughput but limited I/O ~1000 events fit in GPU memory O(10)GB

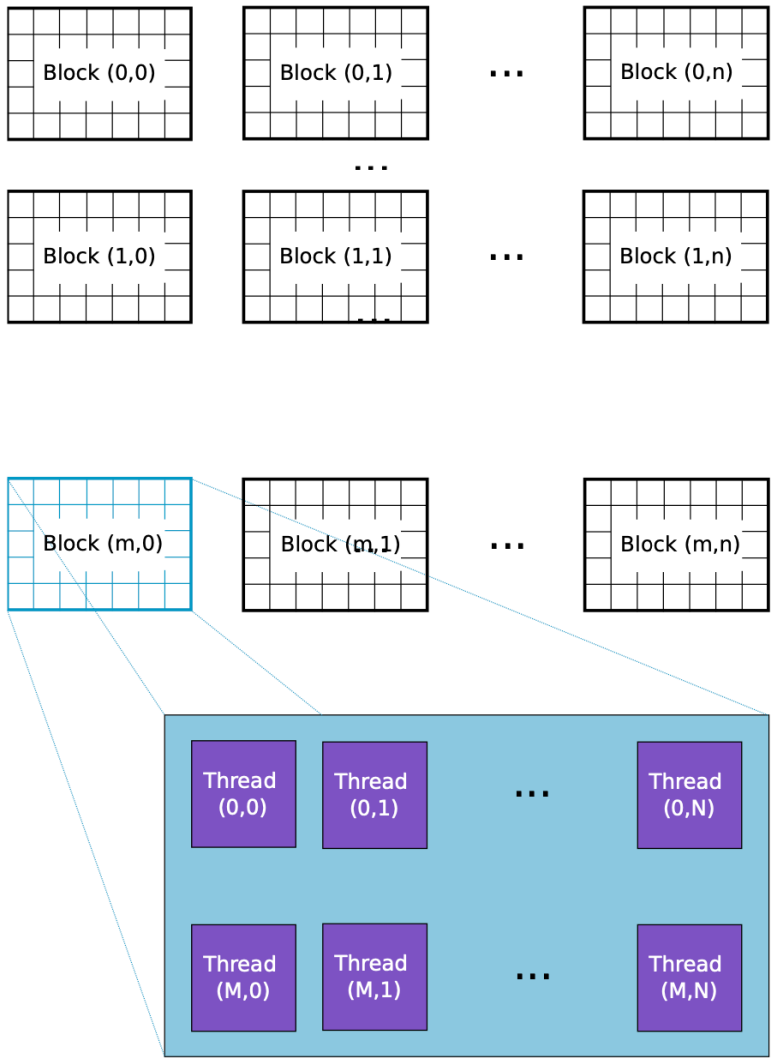
Cross-experiment comparison

- Running at lower instantaneous luminosity compared to other experiments
- Removing hardware trigger means we need to reconstruct ~1800M tracks per second
- Exploiting the GPU parallelisation is crucial
- **Challenge**: finding the optimal **parallelisation** and **memory management**
  → small computer science parenthesis ;)

	LHCb	ATLAS	CMS	ALICE
$\mathcal{L}\,[cm^{-2}s^{-1}]$	2×10^{33}	2×10^{34}	2×10^{34}	6×10^{27}
pile-up	5	60	60	1
reconstruction rate	30 MHz	100 kHz	100 kHz	50 kHz
reconstructed tracks/s	1800 M	90 M	90 M	10 M

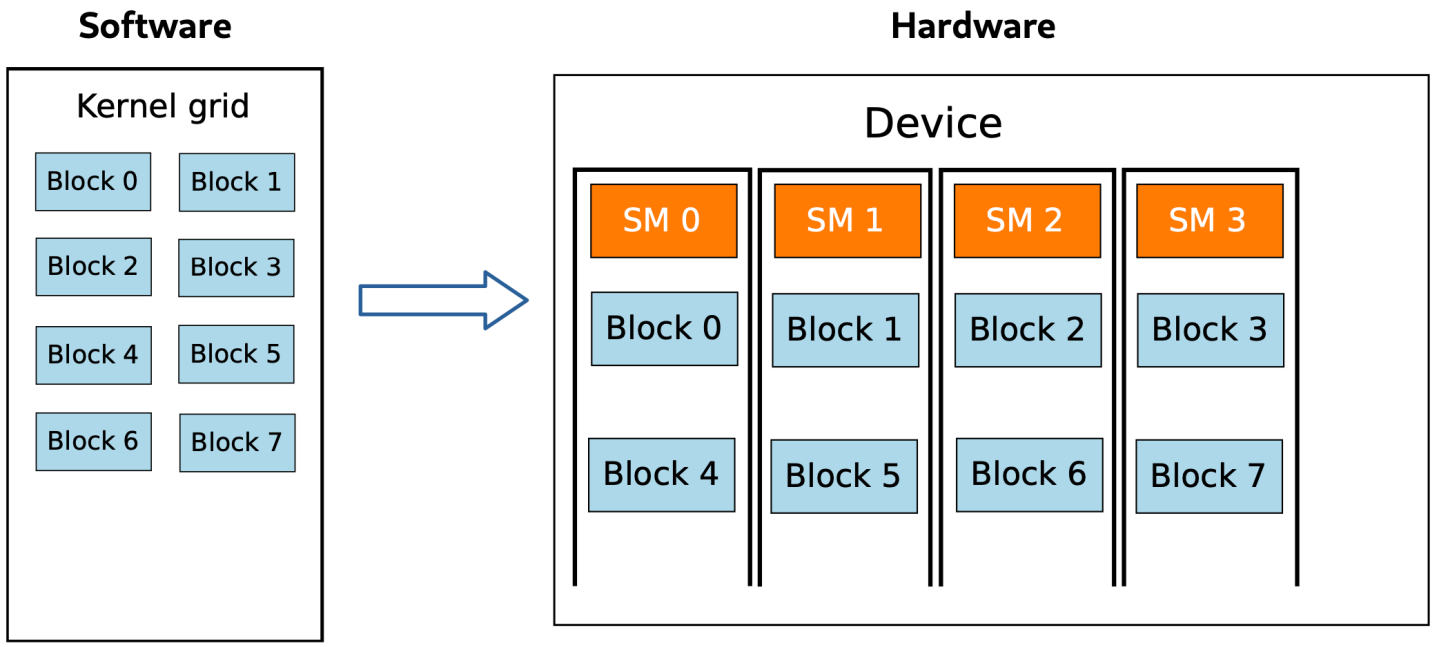
GPU parallelisation

- The GPU code (**kernel**) is executed in many **threads**
- The total number of threads are split into **blocks** (fixed set of threads, generally maximum of 1024)
- Each thread processes the same instruction, the kernel, each one on different data
- We can go up to 3 dimensions both in blocks and threads



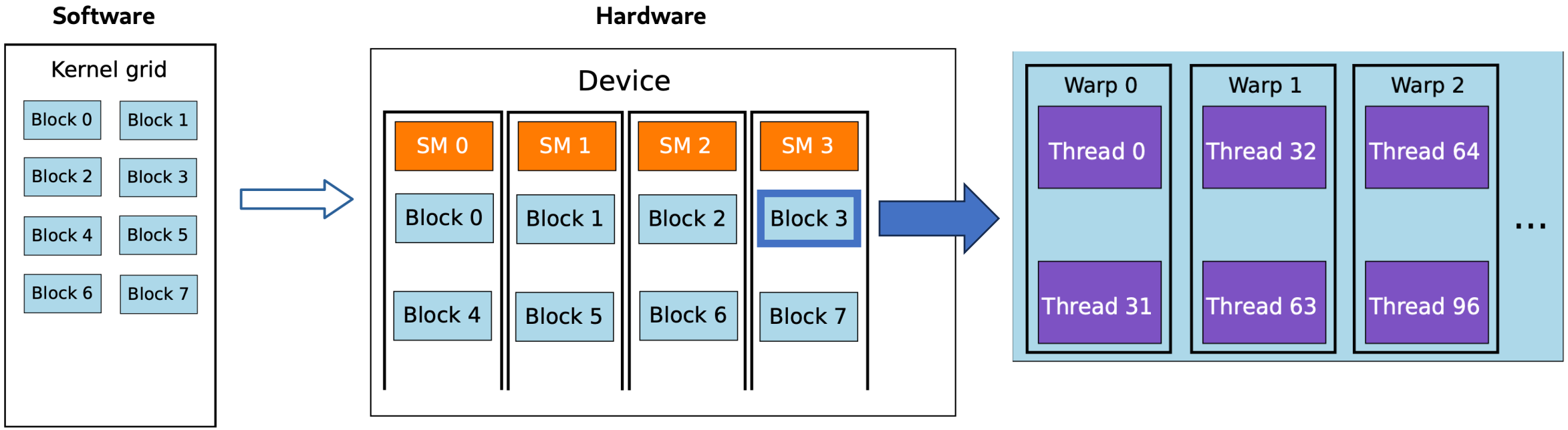
Assignment to Streaming multiprocessors

- Once the **kernel** is defined, the processes are divided into blocks a scheduled to the streaming multiprocessors (SM) of the GPU according to resource usage (memory, registers, …)
- The execution of the blocks is arbitrary



Warps assignment

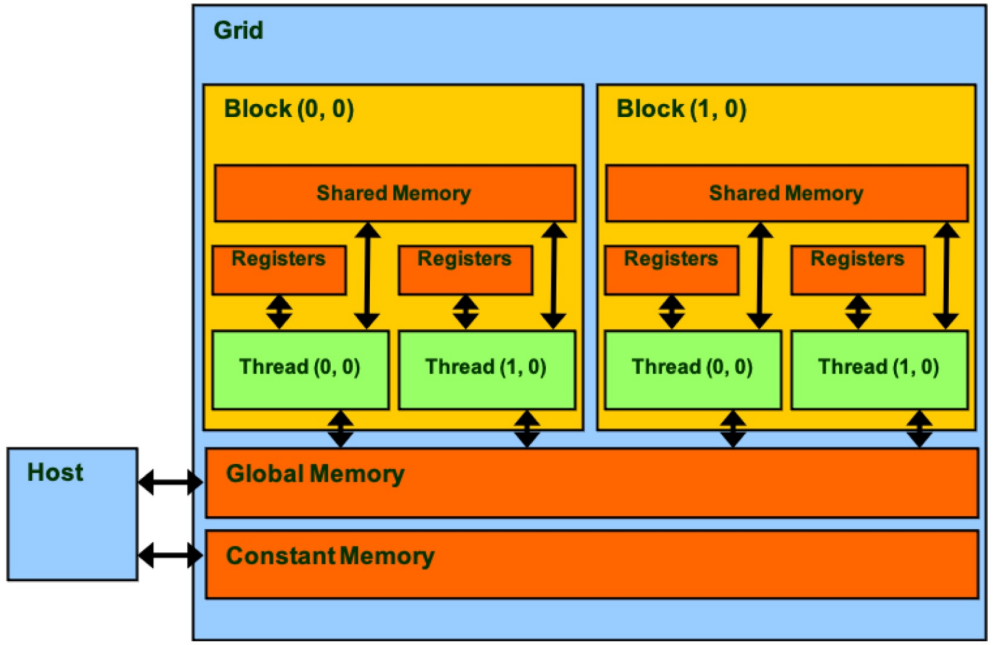
- Within a block, threads are processed in **warps** (= 32 threads in Nvidia GPUs)
- Warps are the smallest entity, meaning block size should be chosen as multiple of 32 (or warp_size)
- This ensures no threads are inherently in idle state



GPU memory management

- The GPU has 3 main kind of memory:

1. **Global memory**: high latency, GBs of space
   - Main memory
   - Communication with the CPU (host)
2. **Caches**: lower latency, KBs of space
   - **Shared memory**: allows communication among threads in one block
   - **Constant memory**: read-only memory (only write from host), used to store constants
3. **Registers**: lowest latency configurable (usually 255 registers per thread)
   - Accessible only from single thread
   - All variables defined are stored in registers
   - If exceded, can result in performance penalty

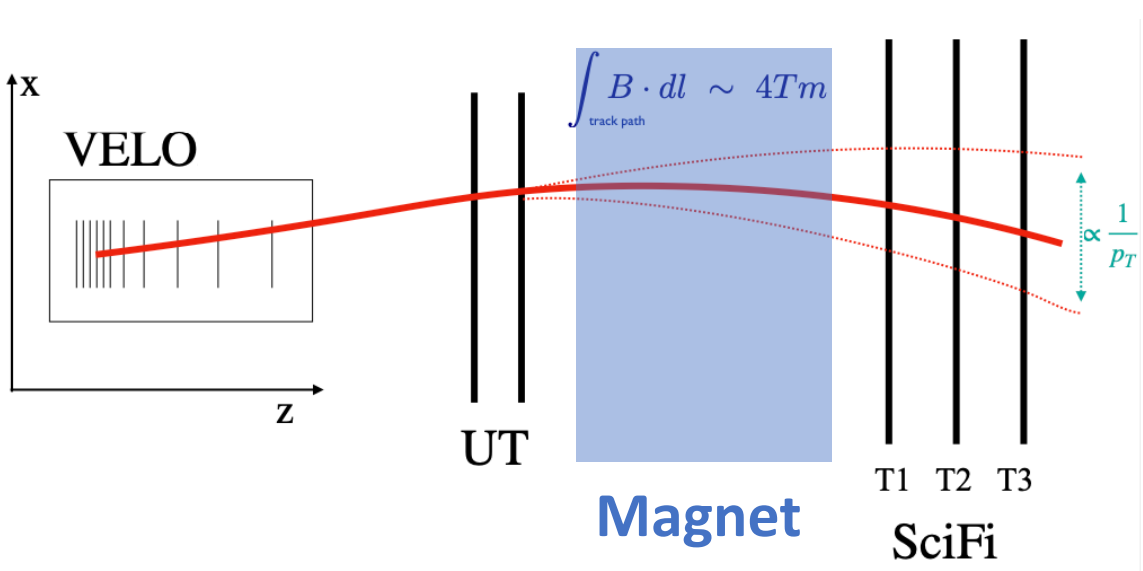


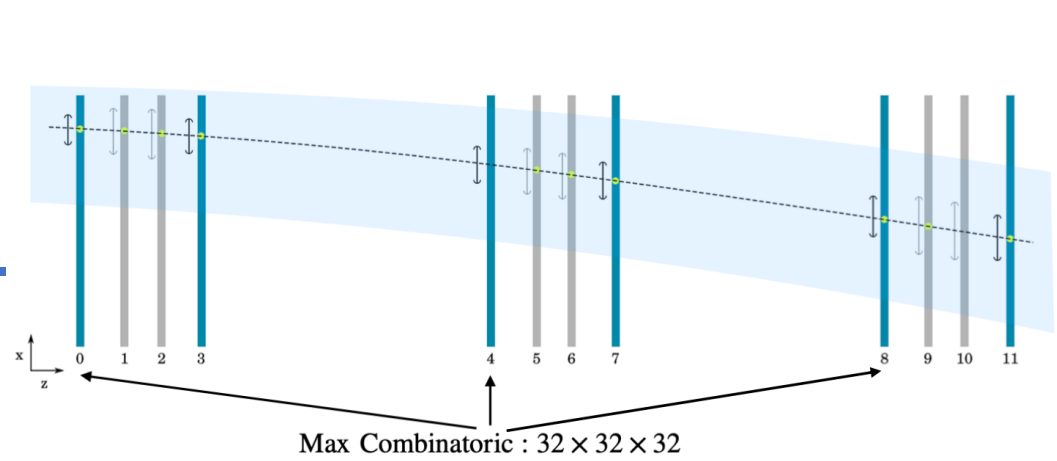
Application to LHCb algorithms

- Assigning each block to a single pp collision (event)
- Use threads within one block for extra parallelisation
- Take into account the warp size (= 32 threads) → optimise code to take this into account and not leave threads in idle state
- Use optimally memory:
  - Hit data can be stored in “shared memory” to speed up accesses when building combinations
  - Store geometry information in “constant memory” which can be accessed by all threads
  - Use “registers” for temporary intermediate variables and simple computations
  - Reducing unnecessary accesses to slower “global memory”

Example: the Forward tracking

- How to fully exploit parallelization power of GPUs?
- Parallelization levels when reconstructing tracks traversing the whole LHCb detector:
  1. Over events, independent p-p collisions
  2. Over input tracks, extrapolate straight tracks in VELO+UT into the magnetic field reaching the SciFi
  3. Over hits in SciFi, meaning possible extrapolations segments → 32 x 32 x 32





IEEE Access 12 (2024) 114198-114211

Coding example

- Example of reconstruction algorithm
- Separate and define device and host input/output in header file

```
22 namespace lf_search_initial_windows {
23   struct Parameters {
24     HOST_INPUT(host_number_of_events_t, unsigned) host_number_of_events;
25     HOST_INPUT(host_number_of_reconstructed_input_tracks_t, unsigned) host_number_of_reconstructed_input_tracks;
26     MASK_INPUT(dev_event_list_t) dev_event_list;
27     DEVICE_INPUT(dev_number_of_events_t, unsigned) dev_number_of_events;
28     DEVICE_INPUT(dev_scifi_hits_t, char) dev_scifi_hits;
29     DEVICE_INPUT(dev_scifi_hit_offsets_t, unsigned) dev_scifi_hit_offsets;
30     DEVICE_INPUT(dev_tracks_view_t, Allen::IMultiEventContainer*) dev_tracks_view;
31     HOST_INPUT(host_track_type_id_t, Allen::TypeIDs) host_track_type_id;
32     DEVICE_INPUT(dev_velo_states_view_t, Allen::Views::Physics::KalmanStates) dev_velo_states_view;
33     DEVICE_INPUT(dev_ut_number_of_selected_velo_tracks_t, unsigned) dev_ut_number_of_selected_velo_tracks;
34     DEVICE_INPUT(dev_ut_selected_velo_tracks_t, unsigned) dev_ut_selected_velo_tracks;
35     DEVICE_OUTPUT(dev_scifi_lf_initial_windows_t, int) dev_scifi_lf_initial_windows;
36     DEVICE_OUTPUT(dev_input_states_t, MiniState) dev_input_states;
37     DEVICE_OUTPUT(dev_scifi_lf_number_of_tracks_t, unsigned) dev_scifi_lf_number_of_tracks;
38     DEVICE_OUTPUT(dev_scifi_lf_tracks_indices_t, unsigned) dev_scifi_lf_tracks_indices;
39   };
```

reconstruction algorithm example

Coding example

- Important:
  - Handling of memory (global, shared, constant, …)
  - Parallelization, best use of blocks and threads

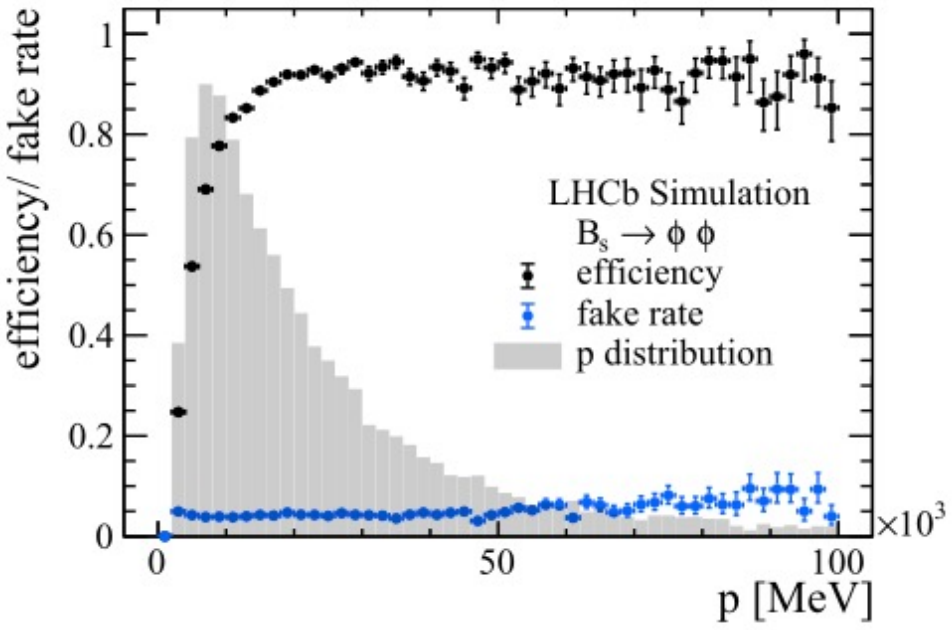
```
18  void lf_search_initial_windows::lf_search_initial_windows_t::set_arguments_size(
19    ArgumentReferences<Parameters> arguments,
20    const RuntimeOptions&,
21    const Constants&) const
22  {
23    const bool with_ut = first<host_track_type_id_t>(arguments) == Allen::TypeIDs::VeloUTTracks;
24    set_size<dev_scifi_lf_initial_windows_t>(
25      arguments,
26      (with_ut ? LookingForward::InputUT::number_of_elements_initial_window :
27                 LookingForward::InputVelo::number_of_elements_initial_window) *
28        first<host_number_of_reconstructed_input_tracks_t>(arguments) * LookingForward::number_of_x_layers);
29    set_size<dev_input_states_t>(arguments, first<host_number_of_reconstructed_input_tracks_t>(arguments));
30    set_size<dev_scifi_lf_number_of_tracks_t>(arguments, 2 * first<host_number_of_events_t>(arguments));
31    set_size<dev_scifi_lf_tracks_indices_t>(arguments, 2 * first<host_number_of_reconstructed_input_tracks_t>(arguments));
32  }
```

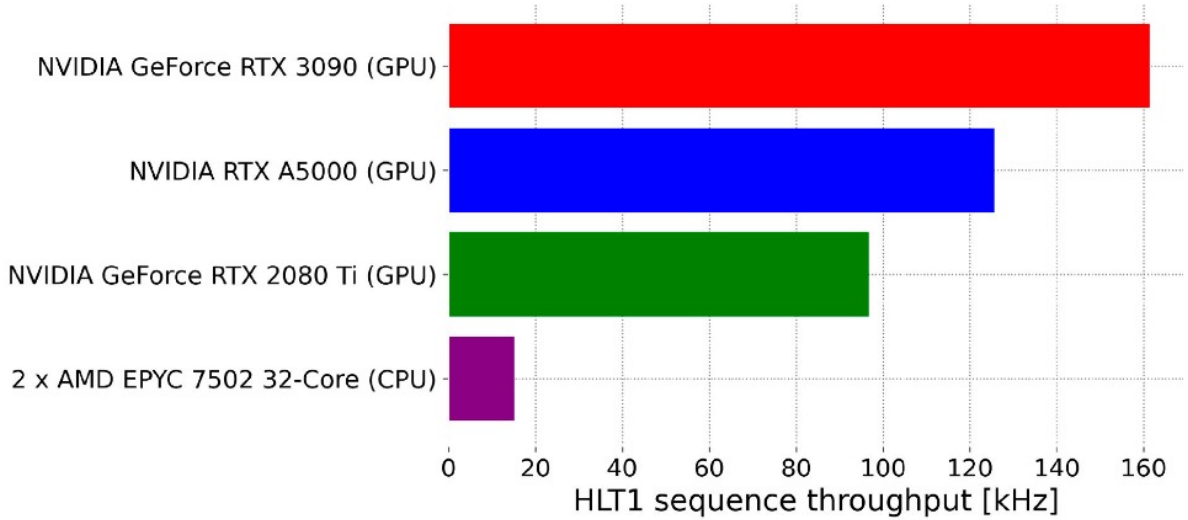
```
137   for (unsigned i_selected_tracks = threadIdx.x; i_selected_tracks < get_number_of_tracks();
138        i_selected_tracks += blockDim.x) {
139     // index for the input track (for velo input only accepted tracks)
140     const int track_index = get_track_index(i_selected_tracks);
141     const auto input_track = input_tracks_view.track(track_index);
```

reconstruction algorithm example

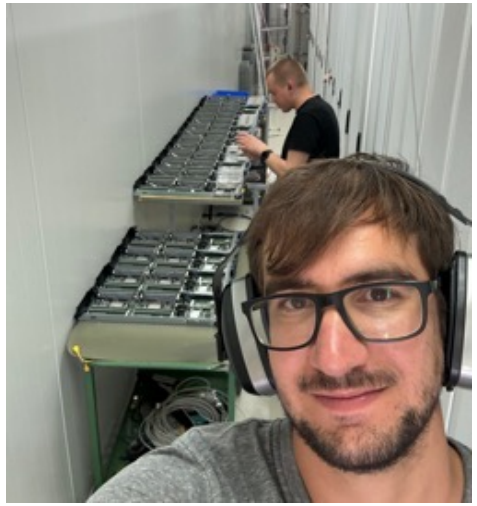
The Forward tracking

- Achieving tracking efficiencies with plateau 90% for a large range of momenta with fake rate ~4-5%
- Most importantly a throughput of 130 kHz per GPU for the whole HLT1 sequence → 300 GPUs are sufficient to handle 40 MHz input

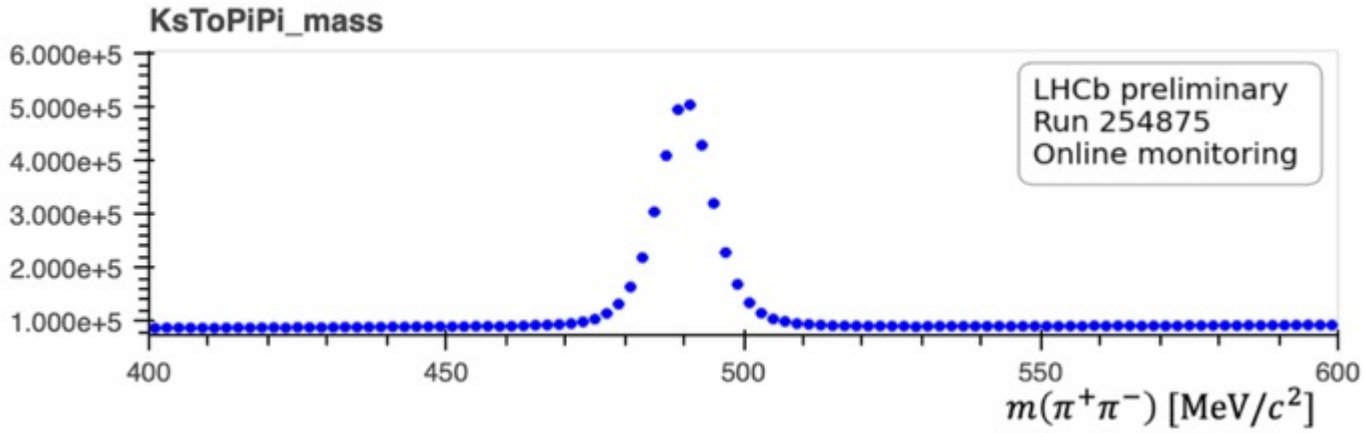


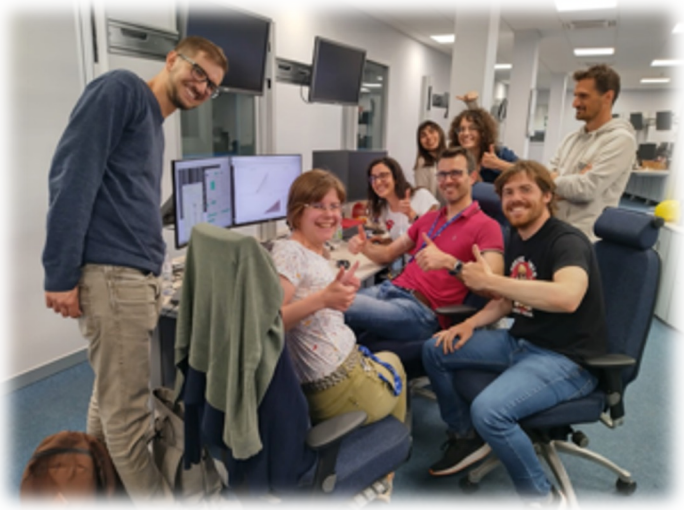


From simulation to …. data taking

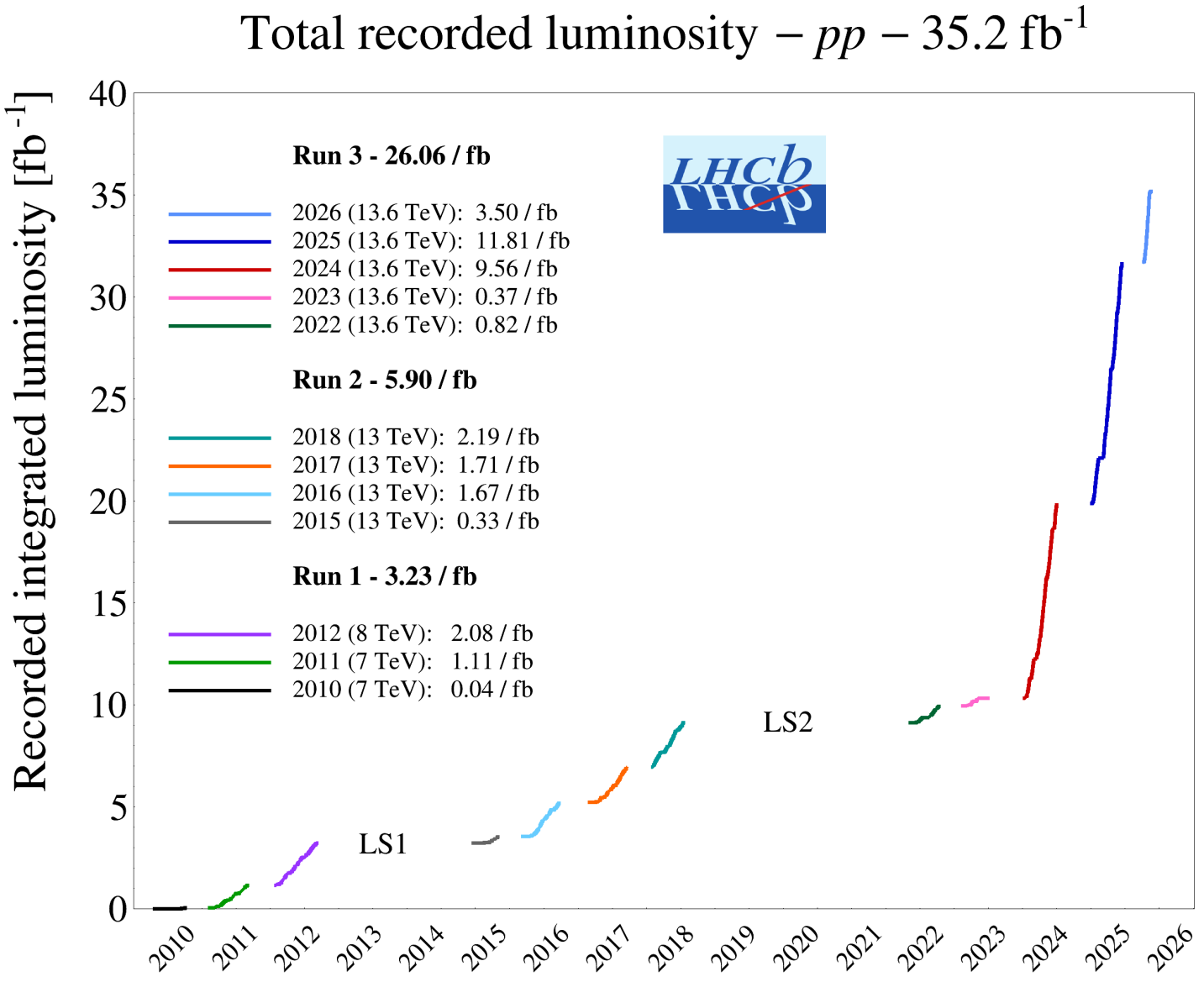


First 2022 online mass peak!





Very successful!

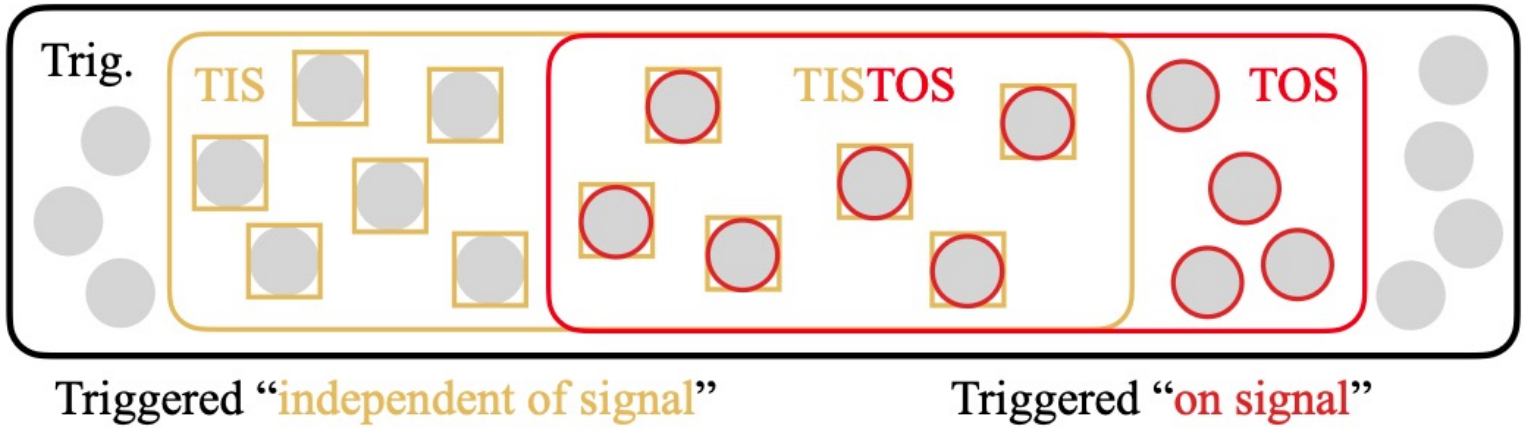


HLT1 performance on data: TISTOS method

- Do not have access to ”total number of events” in data $\varepsilon_{\text{Trig.}} = \frac{N_{\text{Trig.}}}{N_{\text{Tot.}}}$
- Defining Trigger On Signal (TOS) and Triggered Independent of Signal (TIS)
- Correlations between the TOS and TIS categories can lead to biases → reduced if measure in bins of kinematic (momentum, eta, …)

$$\varepsilon_{\text{Trig.}} = \frac{N_{\text{Trig.}}}{N_{\text{Tot.}}} = \frac{N_{\text{Trig.}}}{N_{\text{TIS}}} \times \frac{N_{\text{TIS}}}{N_{\text{Tot.}}} = \frac{N_{\text{Trig.}}}{N_{\text{TIS}}} \times \varepsilon_{\text{TIS}};$$

$$\varepsilon_{\text{TIS}} \equiv \varepsilon_{\text{TIS|TOS}} = \frac{N_{\text{TISTOS}}}{N_{\text{TOS}}}$$

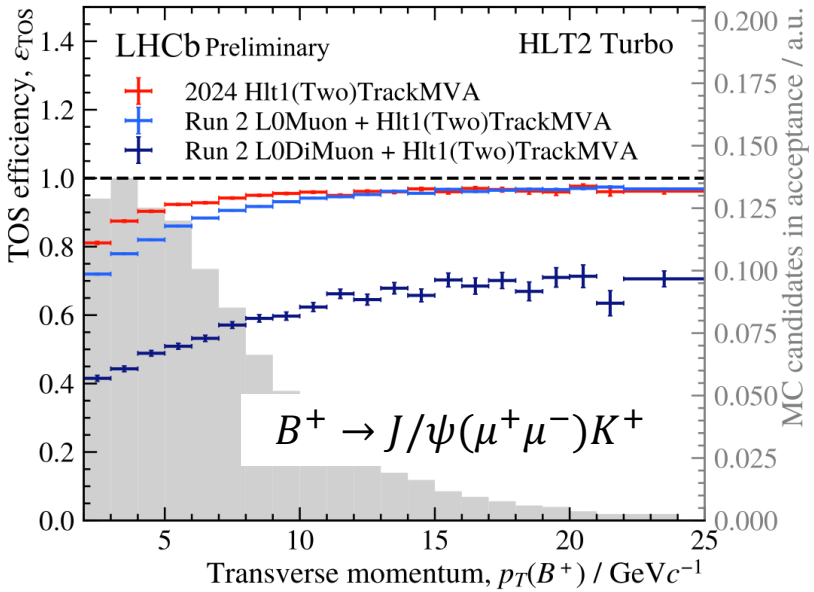


arXiv:2505.15951

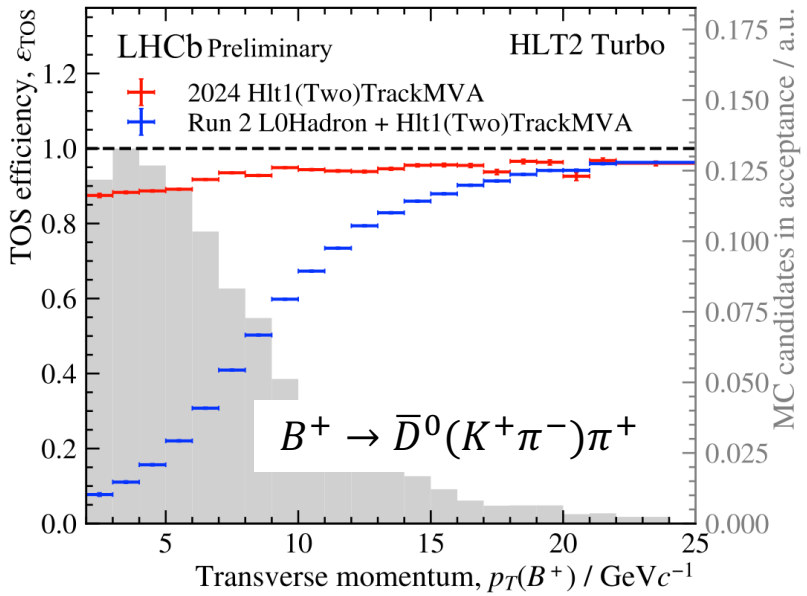
HLT1 performance on data

- Exploiting reconstructed objects to select decays of interest
- Output rate must be around 1MHz
- Comparison with Run2 trigger efficiencies, limited by L0 selections
- Clear gain at low momentum for hadronic and electronic B-mesons modes

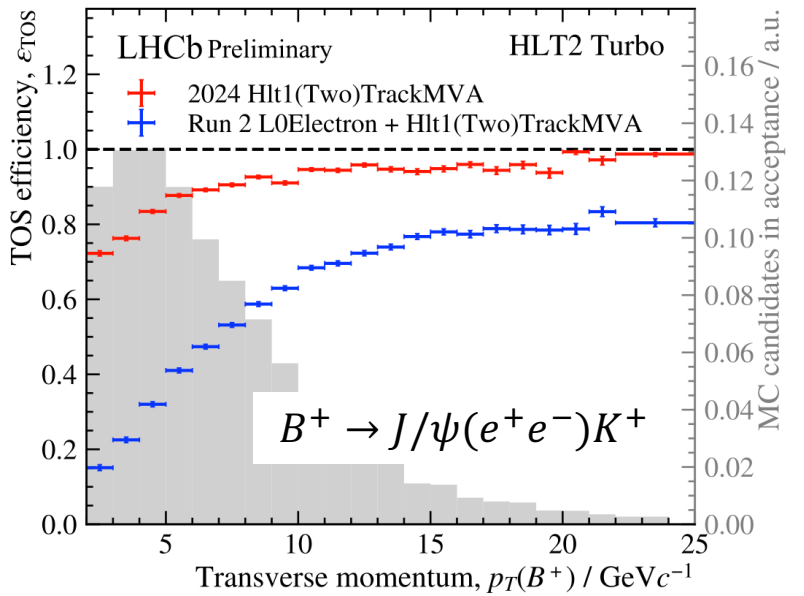
Muonic



Hadronic

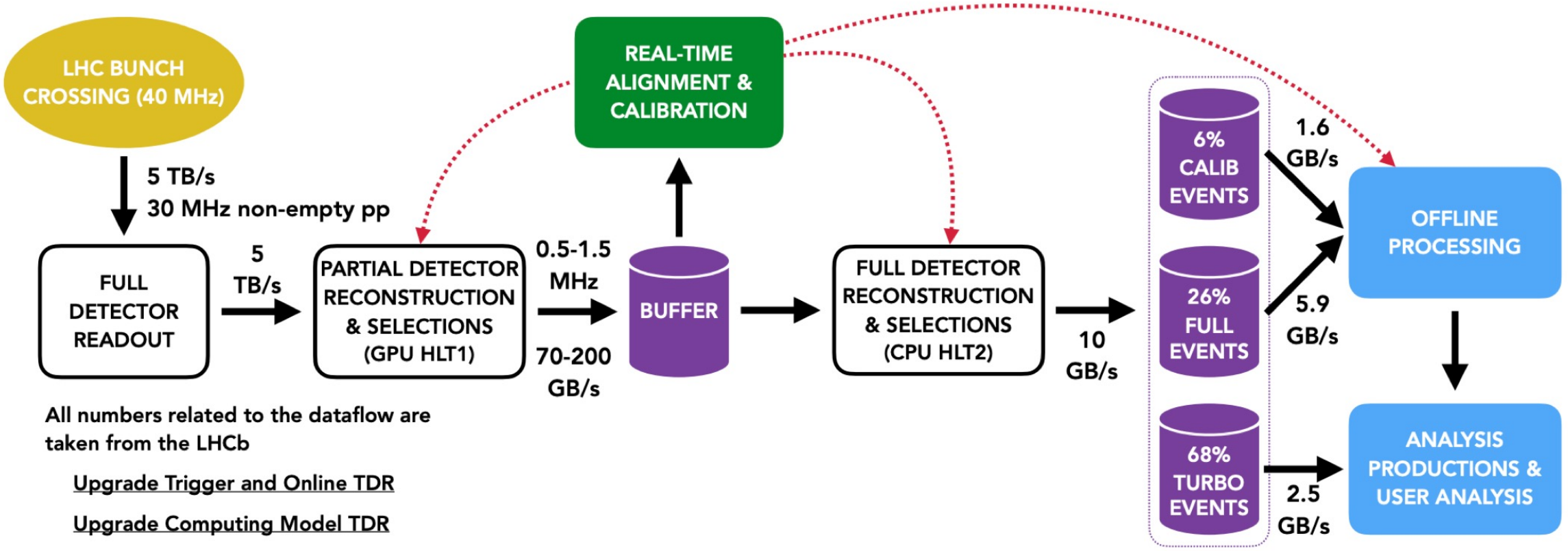


Electronic



LHCB-FIGURE-2024-030

What about the rest of the trigger?



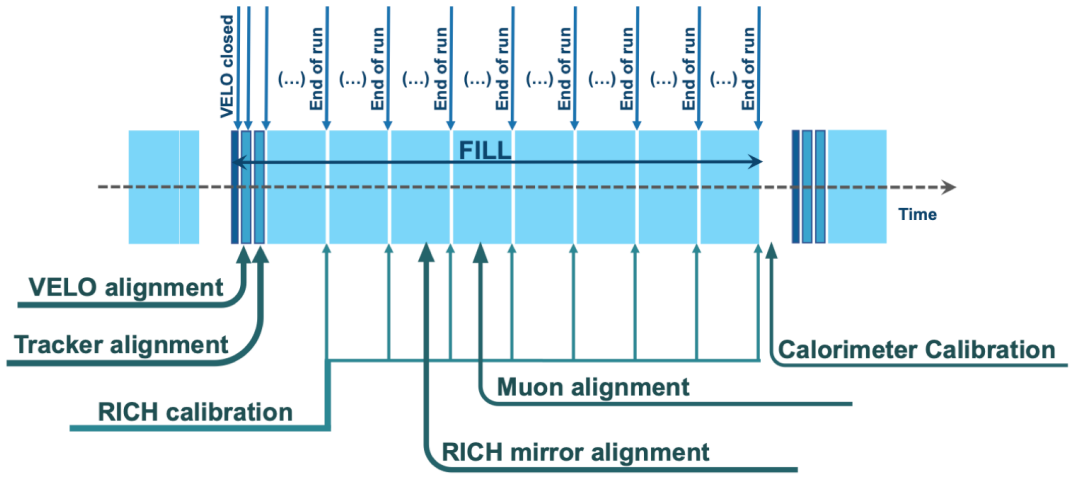
All numbers related to the dataflow are taken from the LHCb

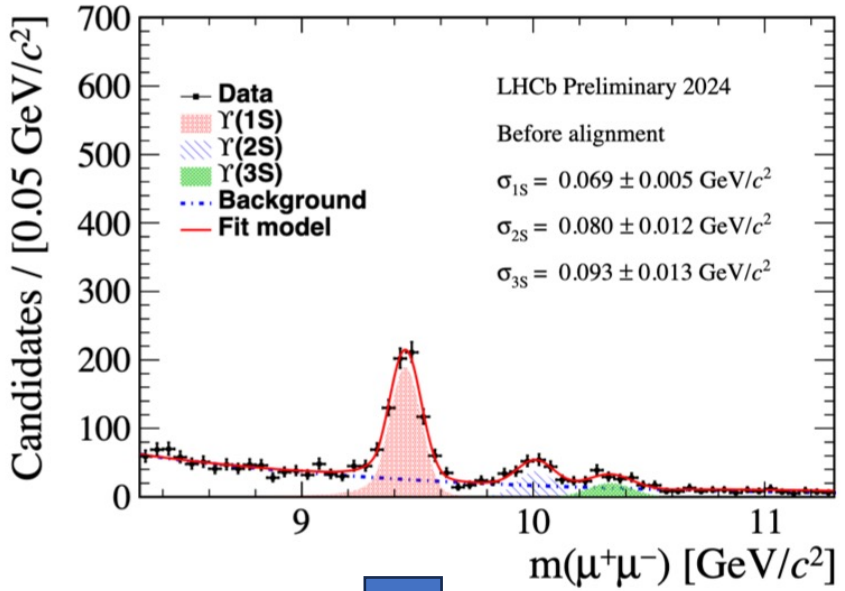
Upgrade Trigger and Online TDR

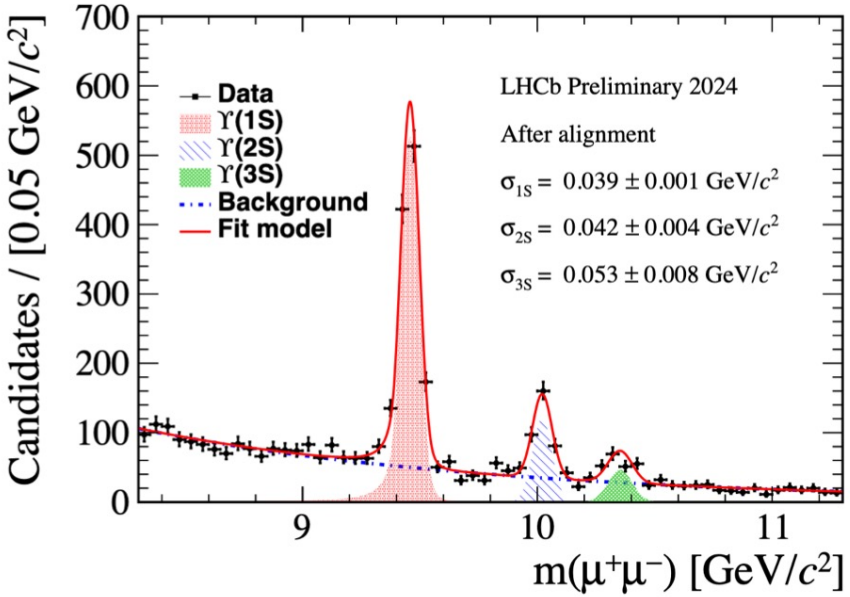
Upgrade Computing Model TDR

Alignment & Calibration

- Output of HLT1 stored in buffer while alignment of the tracking detectors, muon chambers and RICH mirrors
- Performed at each LHC fill or more frequently
- Critical to ensure offline-like quality of HLT2 reconstruction

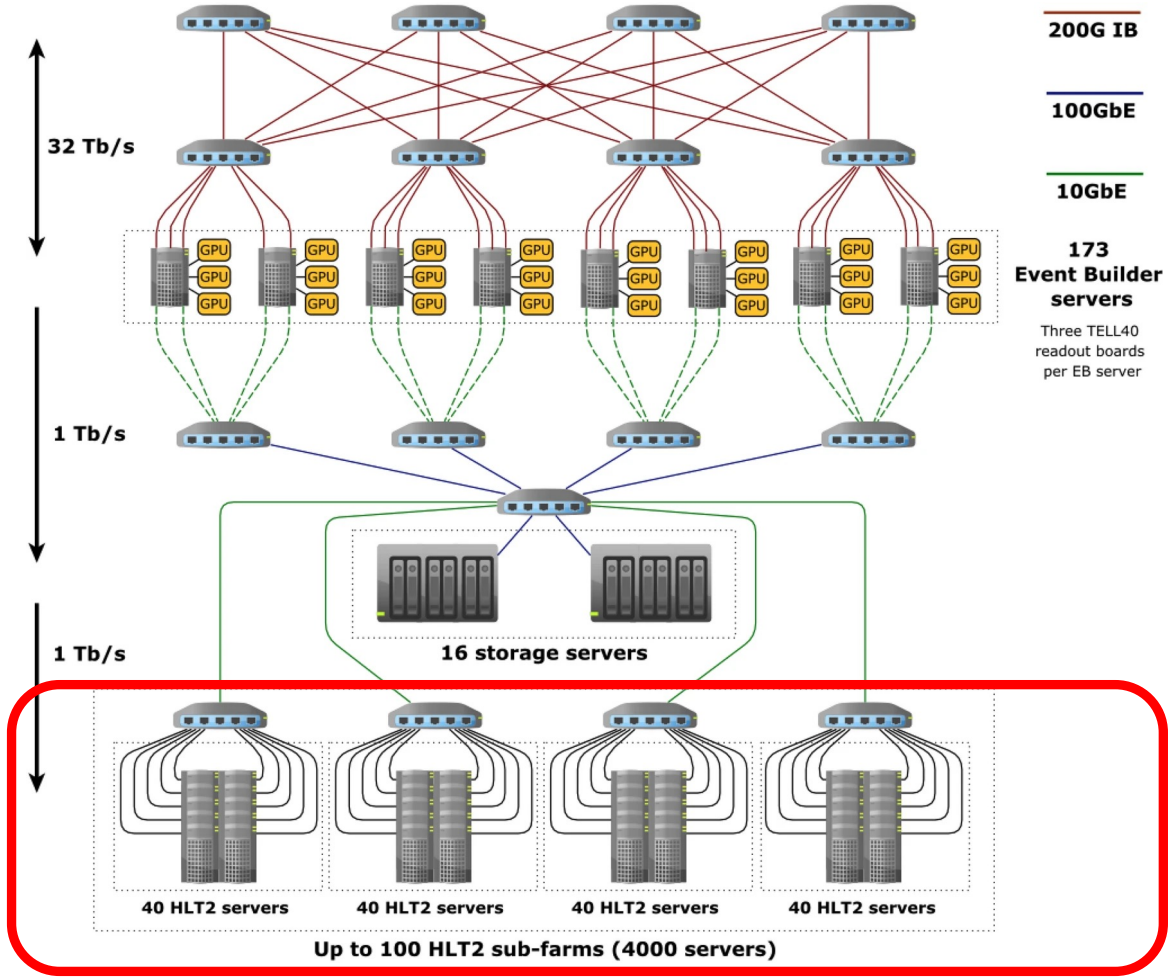






The second high-level trigger: HLT2

- HLT2 can be run asynchronously to HLT1 once the full alignment&calibration is performed
- HLT2 needs to process data at rate greater than half of the HLT1 output (1 MHz): minimum 500 kHz
- Dedicated trigger selections O(3000) to cover broad LHCb physics program
- Limited bandwidth of 10 GB/s of data saved in memory
- Exploiting around 5000 CPUs (now even reaching 900 kHz)

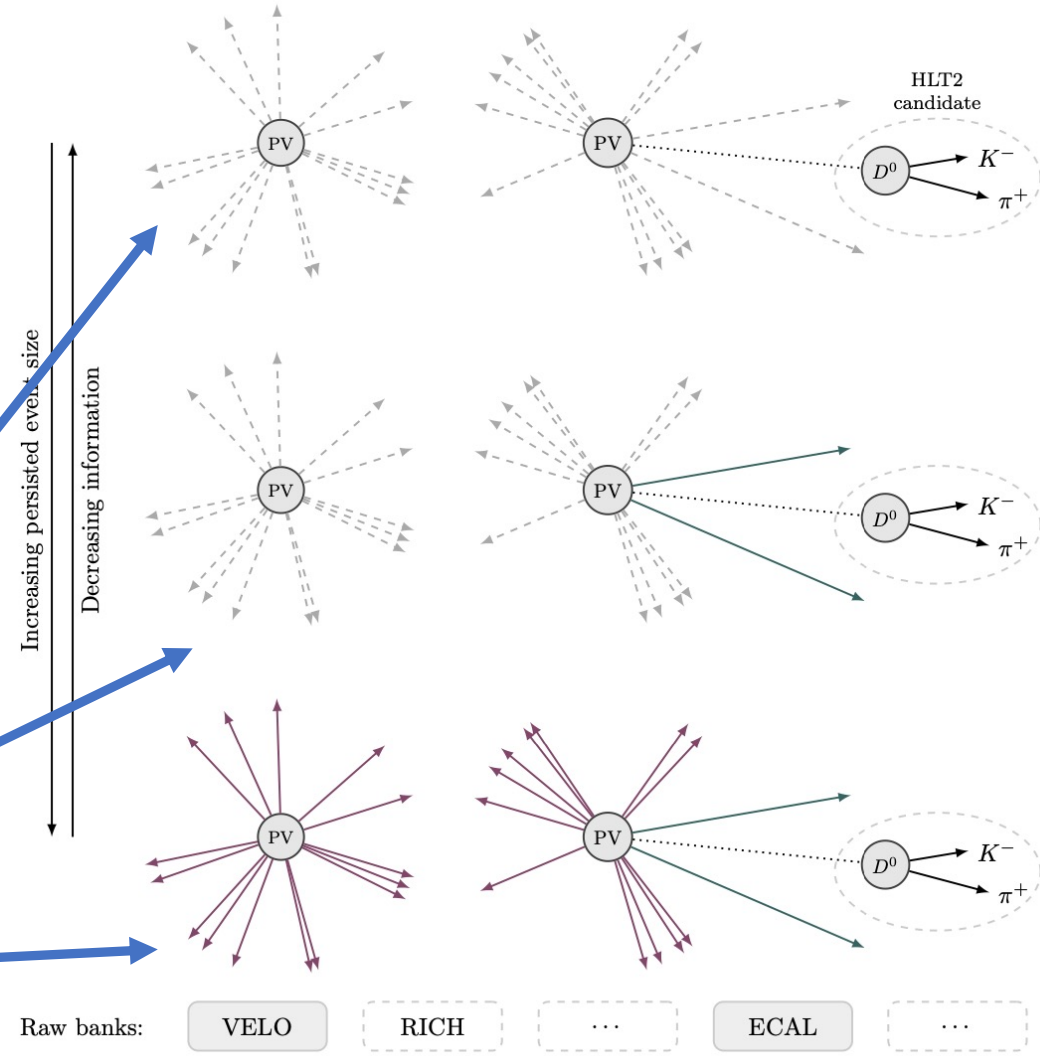


[Comput.Softw.Big Sci. 6 (2022) 1, 1]

HLT2 throughput and bandwidth

- Throughput (minimum 500 kHz):
  - Structure-of-Arrays collections to exploit vectorisation and multi-threading
  - Throughput-Oriented selections (Thor functors): built at compile time into cache memory and agnostic on I/O type
- Bandwidth (maximum 10 GB/s):
  - Turbo: saving only info related to the signal candidate reducing the event size by a factor 10 (about 70% of all LHCb selections)
  - Selective persistency: save additional objects relative to the signal
  - Full persistency

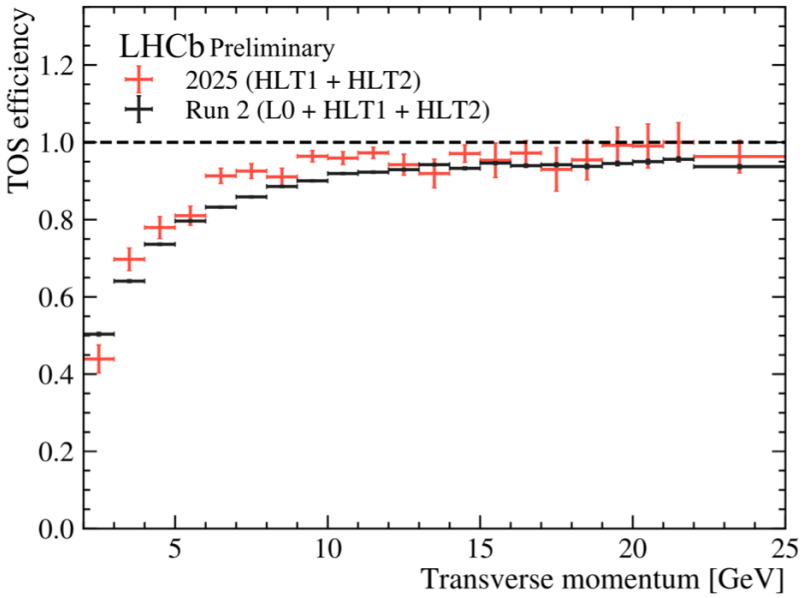
JINST 14 (2019) P04006



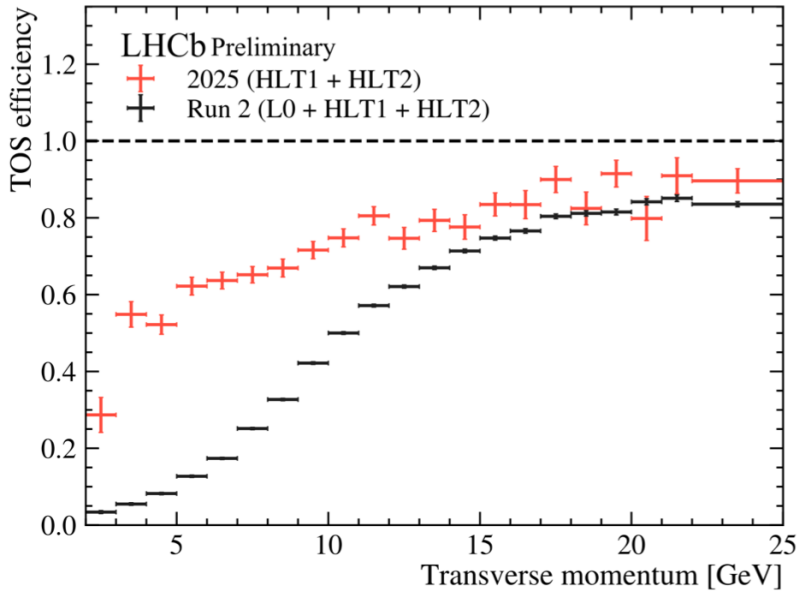
HLT2 performance

- Combined efficiency HLT1+HLT2 vs Run2 trigger (L0+HLT1+HLT2)

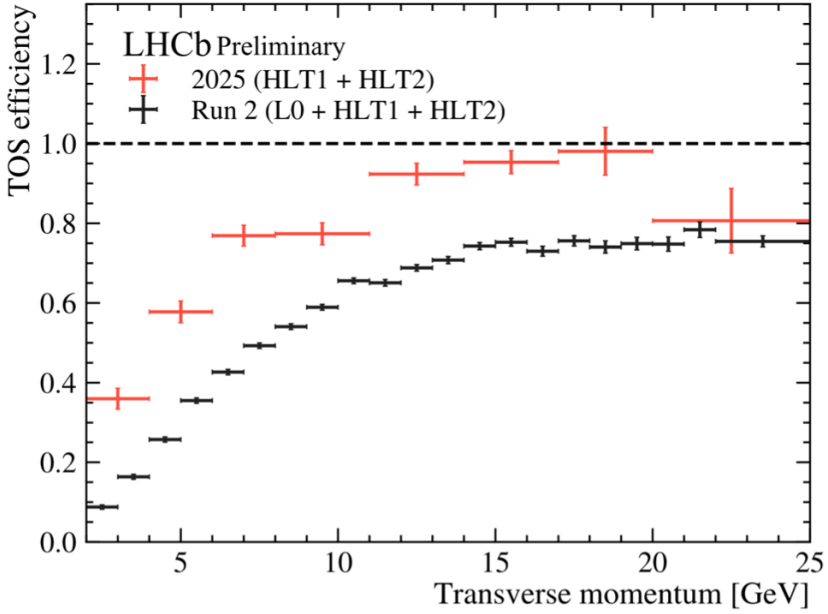
$B^+ \to J/\psi\,(\mu^+\mu^-)\,K^+$



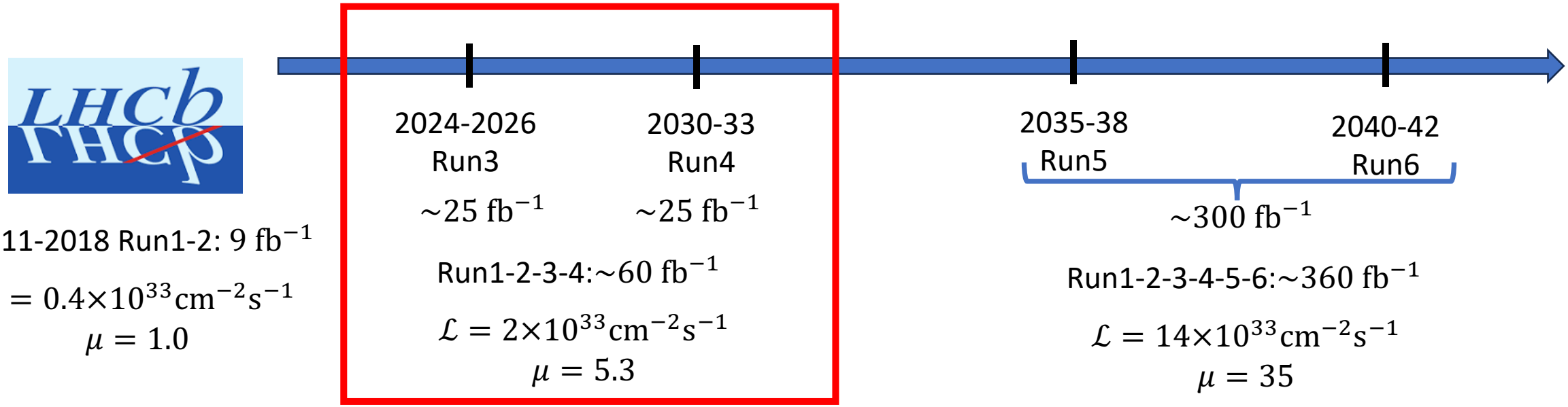
$B^0 \to D^-\,(K^+\pi^-\pi^-)\,\pi^+$



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



The future of LHCb and trigger system

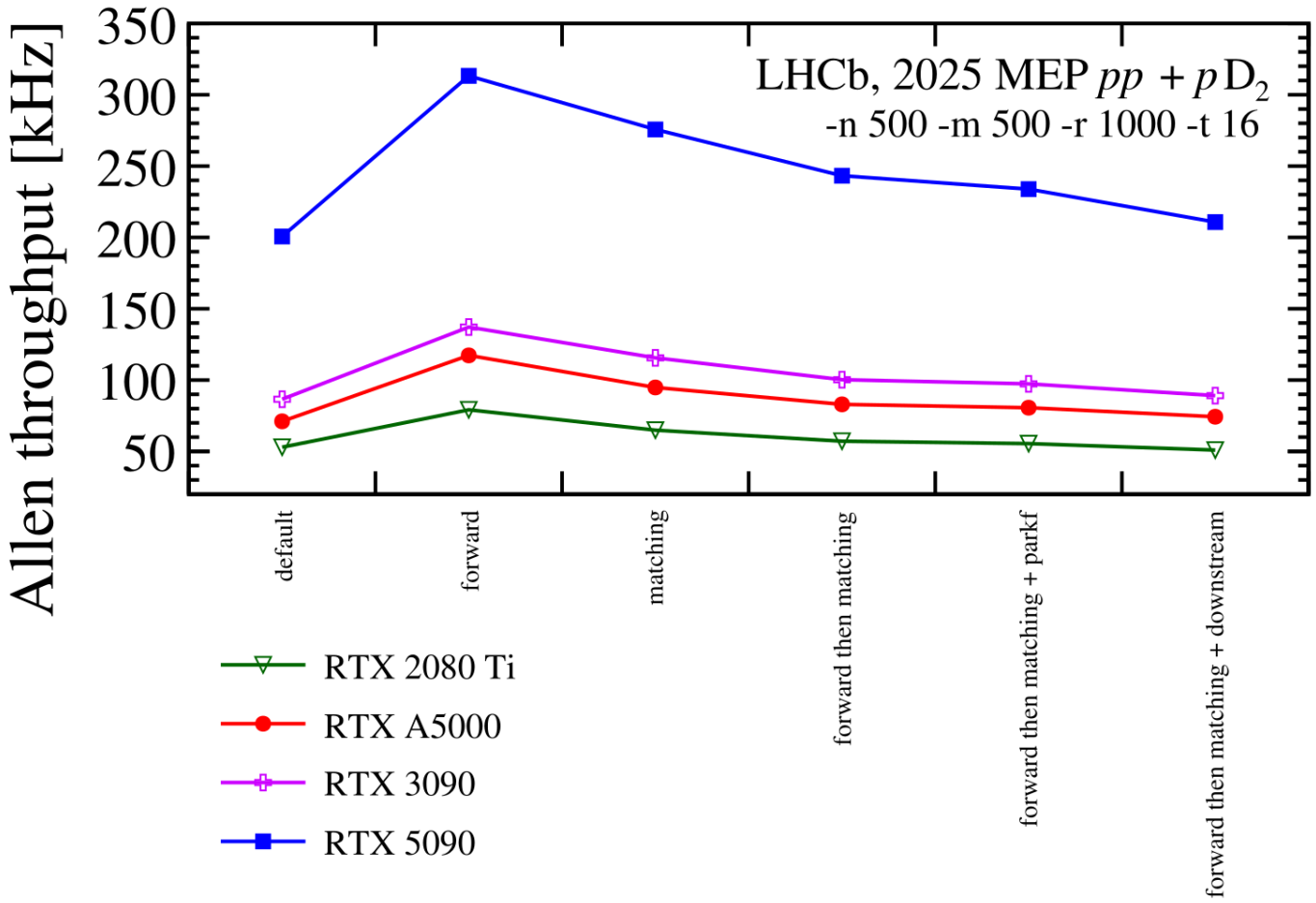


To respect high-throughput constraints HLT1 algorithms reach almost always lower physics efficiency compared to HLT2 ones (low momentum tracking, simpler calo reconstruction, …)

Can we optimise HLT1 efficiency also exploiting modern GPUs?

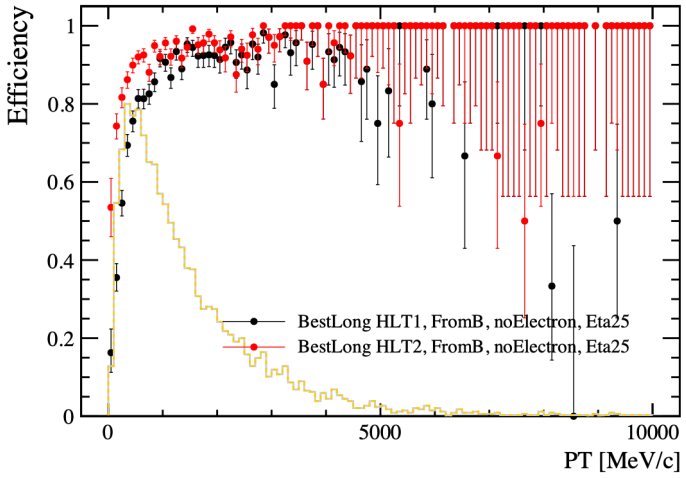
Can modern GPUs help?

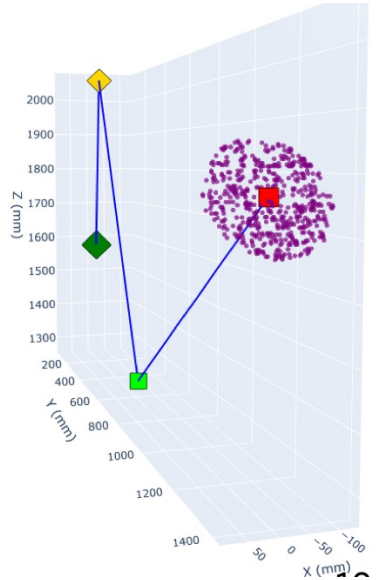
- RTX 5090 has ~3 times more CUDA cores and ~4 times TFLOPS compared to our current A5000 (but roughly twice the price)
- Reaches almost ~2.5 higher HLT1 throughput with no optimisations
- We can exploit also larger memories in modern GPUs



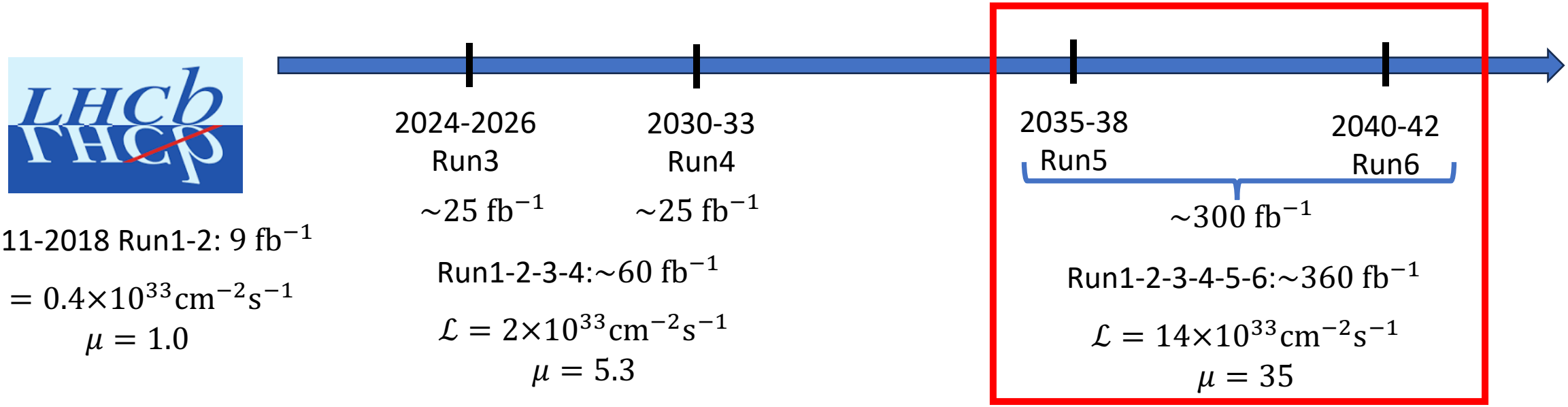
Physics efficiency

- Exploring opportunities to improve reconstruction and selection at low momentum:
  - Challenge: HLT2 uses a very sequential approach going from higher to lower momenta tracks, flagging at each step the used hits
- Adding PID via RICH reconstruction:
  - Challenge: photon bouncing-off spherical mirrors needs a compute-intesive solving of 4th order polynomial equation but can parallelize over input tracks vs photons vs particle hypothesis



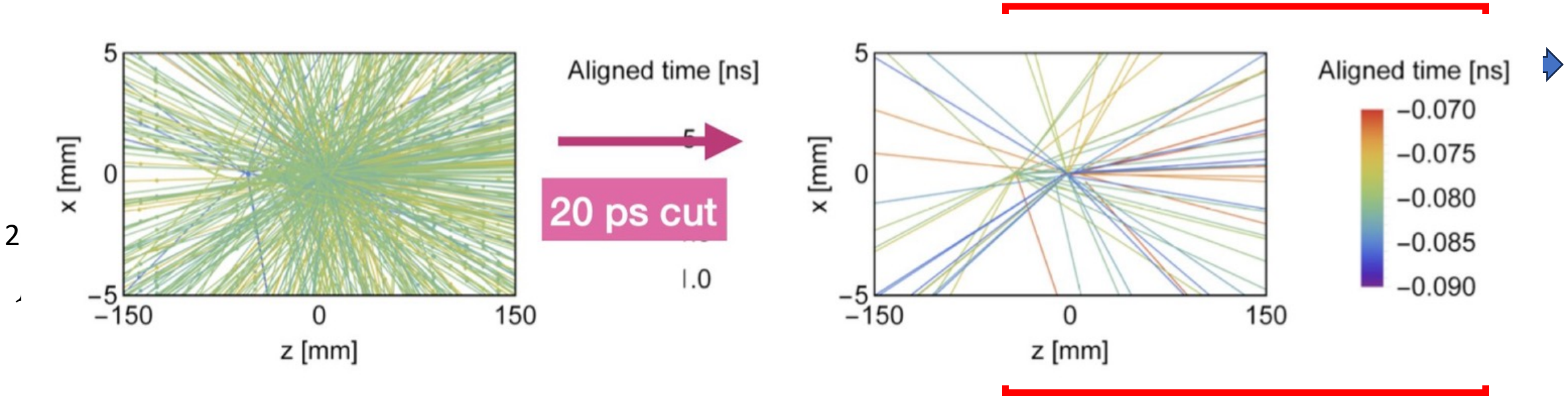


The future of LHCb and trigger system



- Increasing the instataneous luminosity by a factor 7 reaching a pileup of 35-40
- Need timing information to distinguish each pp collisions → upgrade of LHCb subdetectors reaching ~20 ps resolution

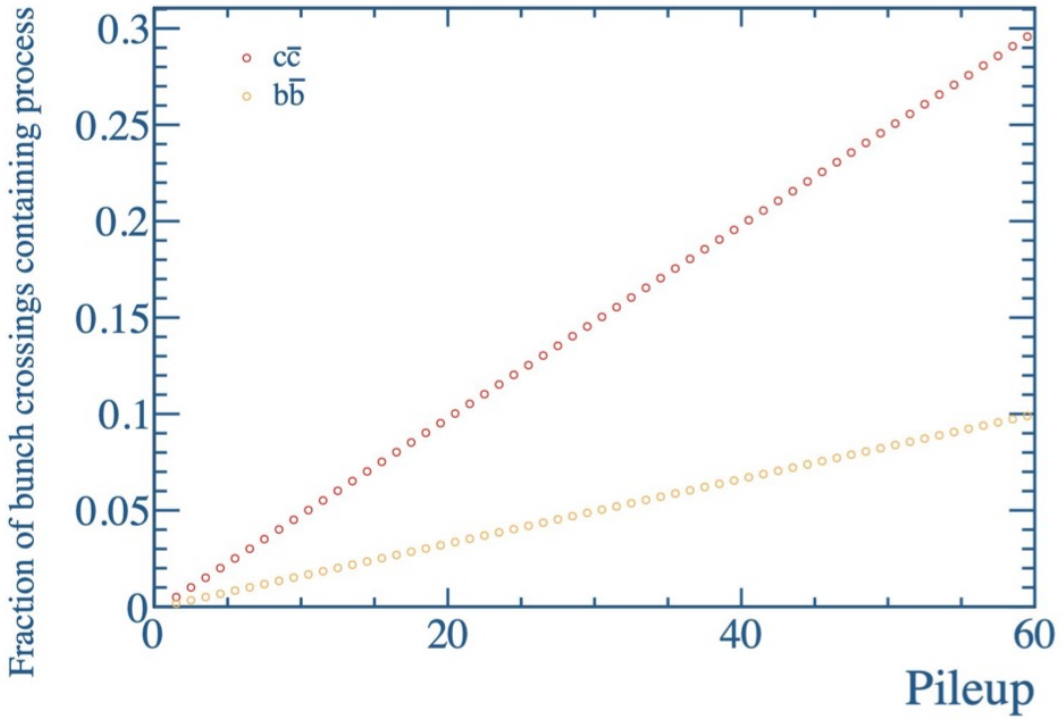
The future of LHCb and trigger system



- Increasing the instataneous luminosity by a factor 7 reaching a pileup of 35-40
- Need timing information to distinguish each pp collisions → upgrade of LHCb subdetectors reaching ~20 ps resolution

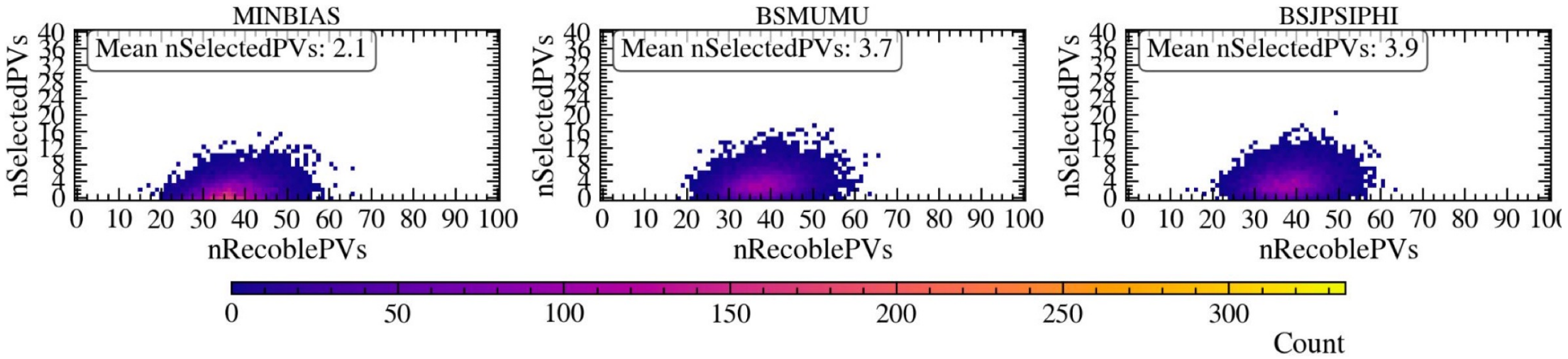
The future of the trigger: Run5-6

- Problem: too much signal
- At pileup = 40 we have, 20% of charm and 5% of beauty production
- Running with the current trigger would imply → 11 MHz or 5 TB/s to be saved to disk (or 3 Exabyte /week)
- Not realistic, or at least very expensive



Pileup suppression in the trigger

- Between ~40 primary vertices (pp collisions) only 3-4 contain interesting events
- If this decision separation can be done quickly exploiting ~ps resolution information, the majority of the event data can be discarded
- Great challenge and opportunity to study triggering in extreme conditions



Conclusions

- LHCb trigger system fully on software with a heterogenous system (CPU-GPU)
- First stage optimised for GPUs has the crucial task to reconstrut and select events at 40 MHz (one event every 25 ns)
- Full exploitation of GPUs is needed: from parallelisation to memory handling
- Operation was success reaching the goal of $25\ \mathrm{fb}^{-1}$ in 3 years (2024-25-26)
- The knowledge acquired until now will allow us to continue improving the LHCb trigger system un Run4 (2030-33)
- Run5-6 after 2035 will be a completely new challenge with timing information to be exploited at best

Backup