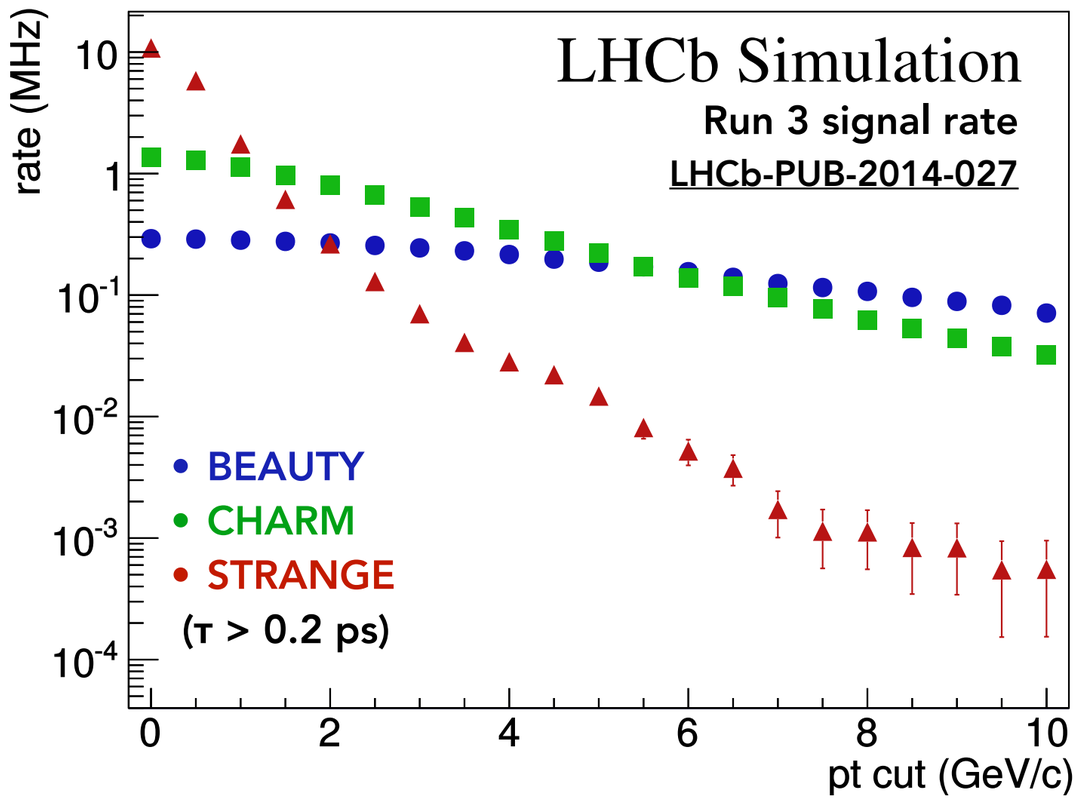
Designing a 30 MHz GPU trigger, the LHCb experience



Vladimir V. Gligorov, CNRS/LPNHE

On behalf of the LHCb Real Time Analysis Project

CPAD 2021, cyberspace, 18.03.2021

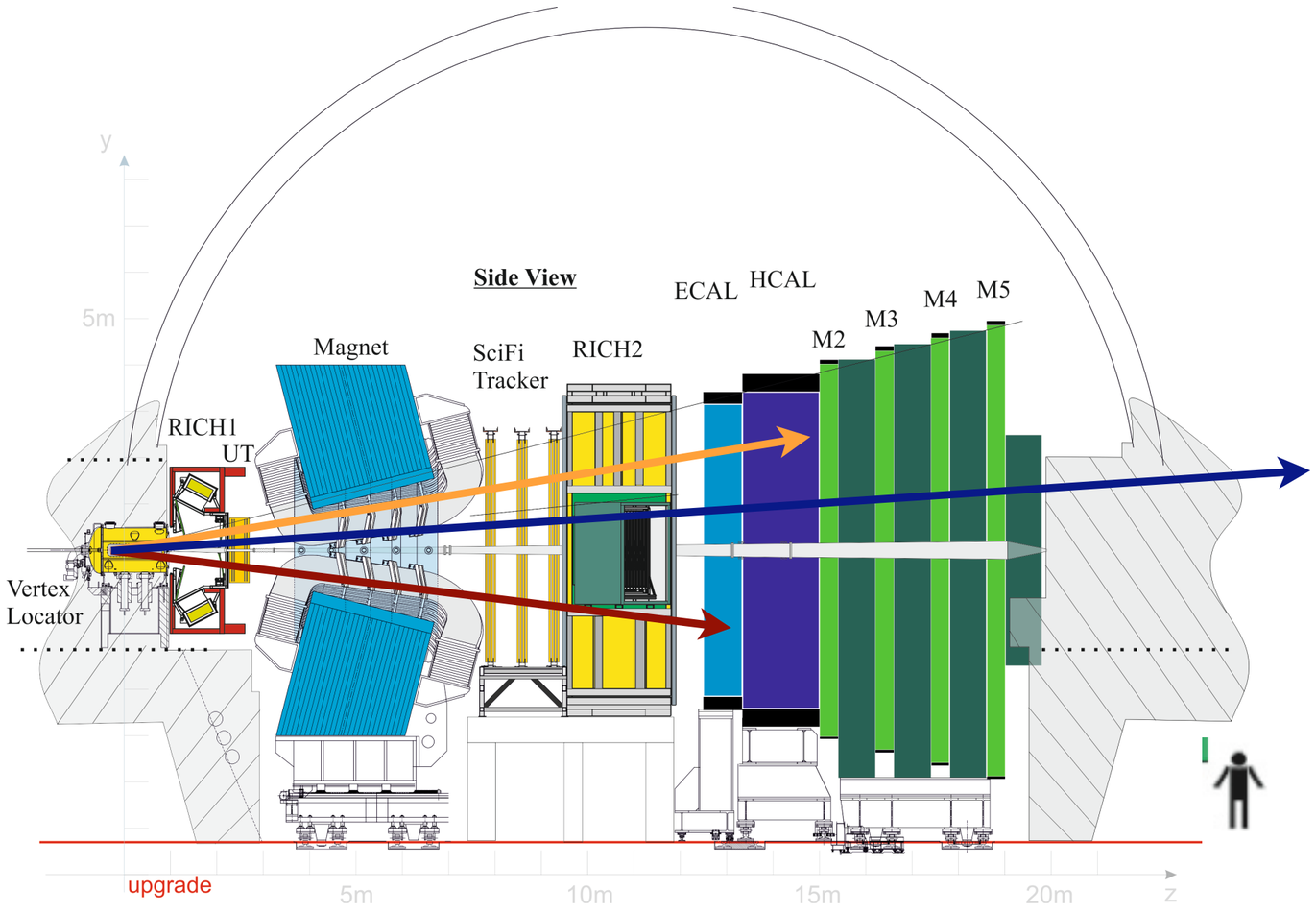
Introduction to LHCb

0.5 - 1% momentum resolution

15 + 29/p_T[GeV] micron impact parameter resolution

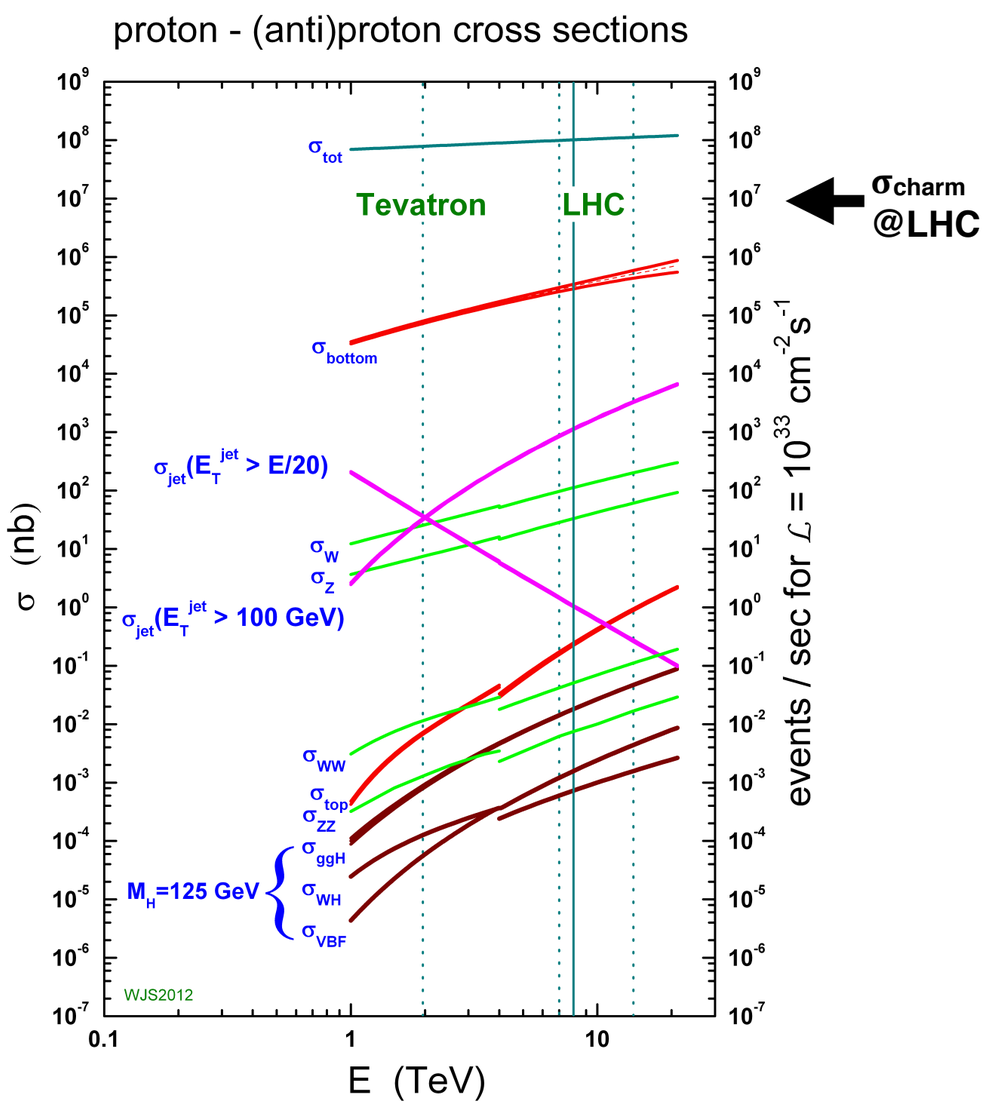
(1+10/√E[GeV])% ECAL resolution

Efficient and high-purity identification of all five stable charged particle types (pion, kaon, proton, electron, muon) over the momentum range 2-100 GeV

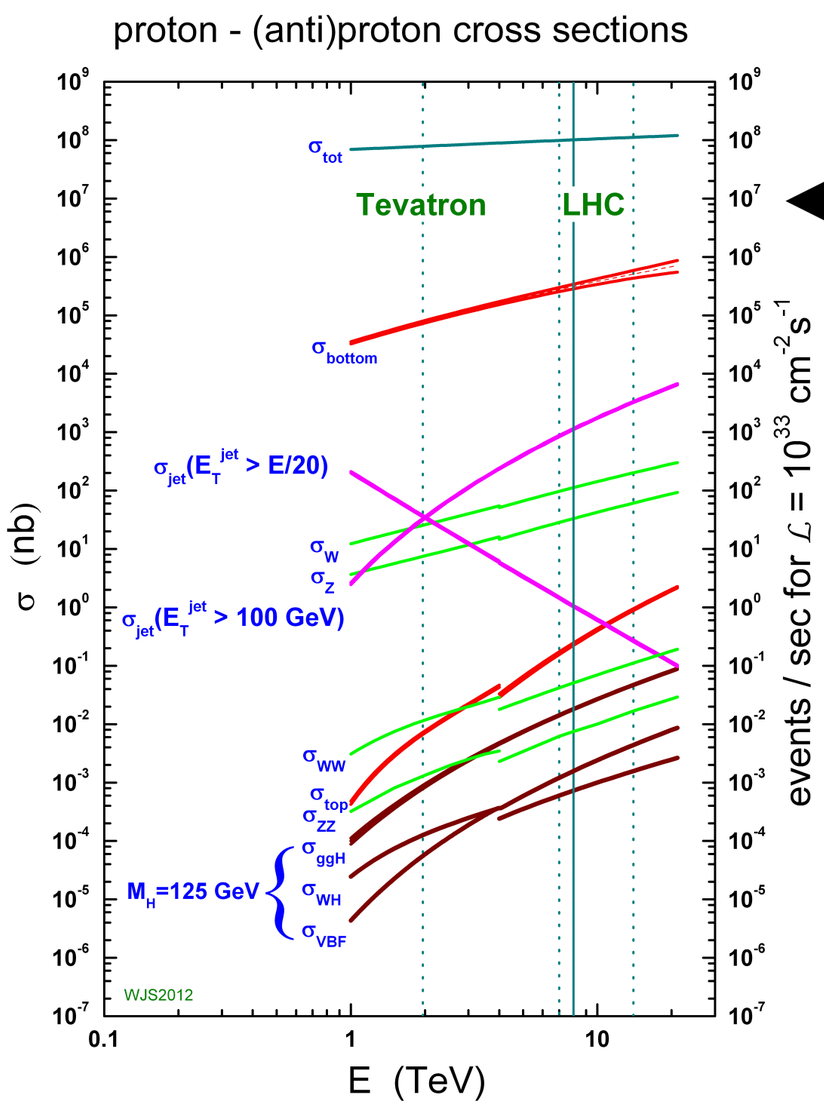


➡ ELECTRONS ➡ HADRONS
➡ PHOTONS ➡ MUONS

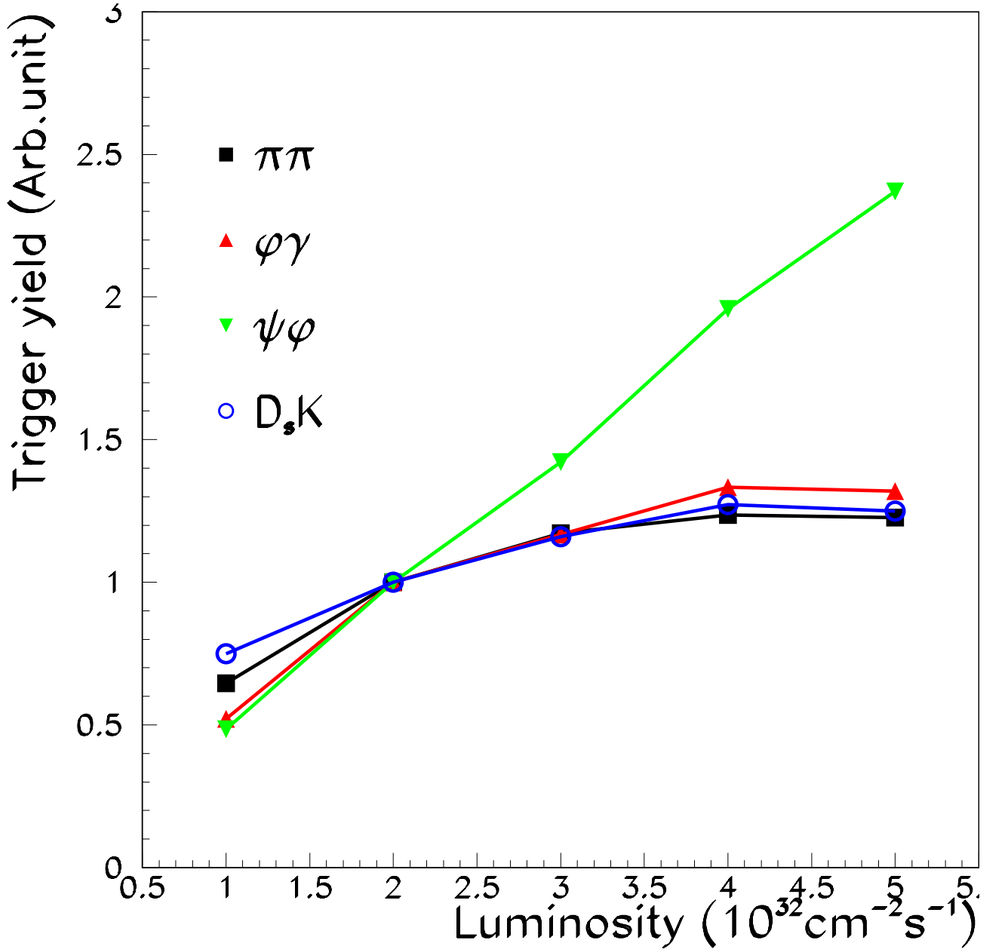
The challenge of triggering @ LHCb



The challenge of triggering @ LHCb



σ_{charm} @LHC

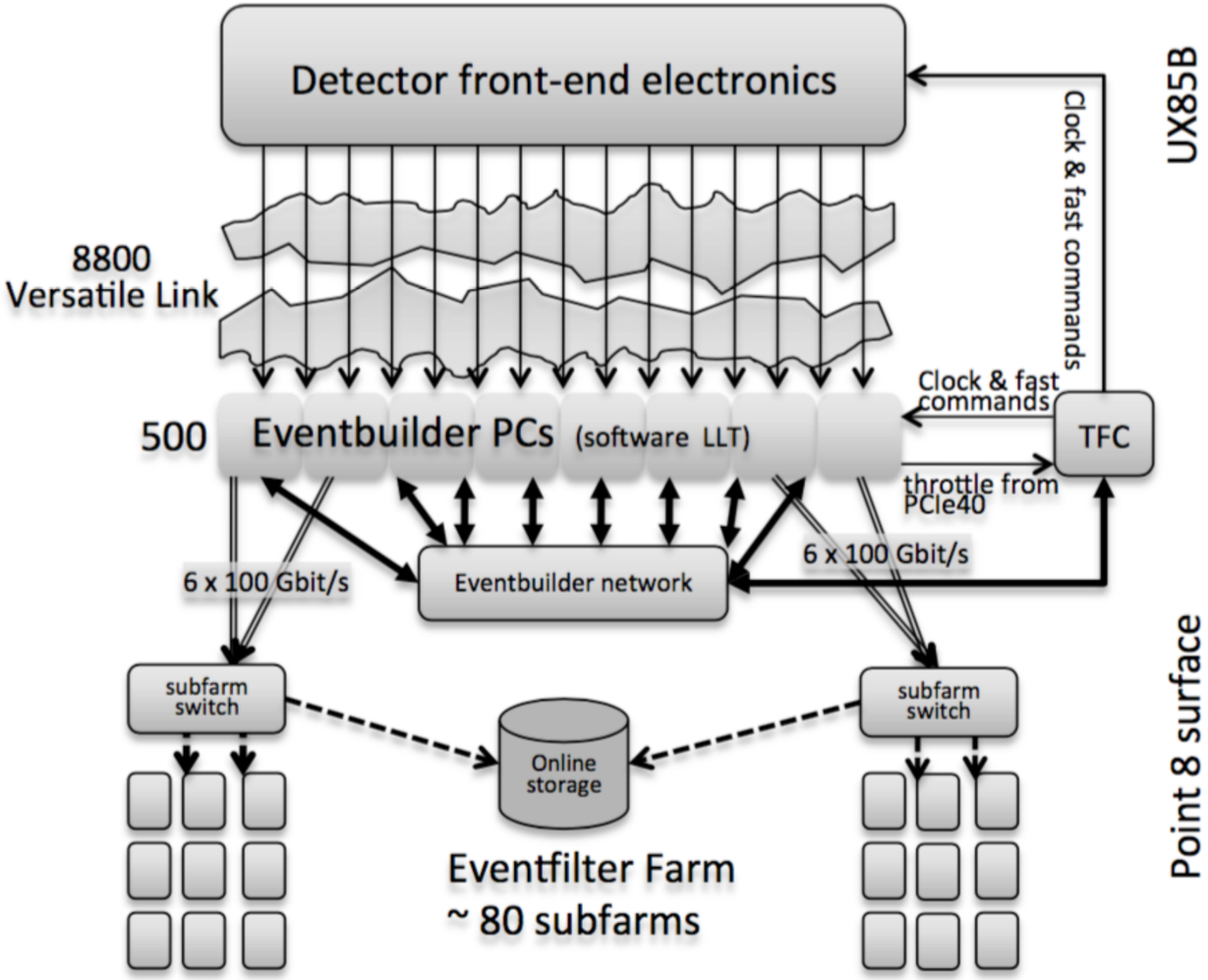


Cross-section for processes of interest to LHCb saturates a traditional CALO/Muon trigger
The LHCb upgrade has to run at $2 \cdot 10^{33}$ cm^{-2}s^{-1} — total rethink necessary

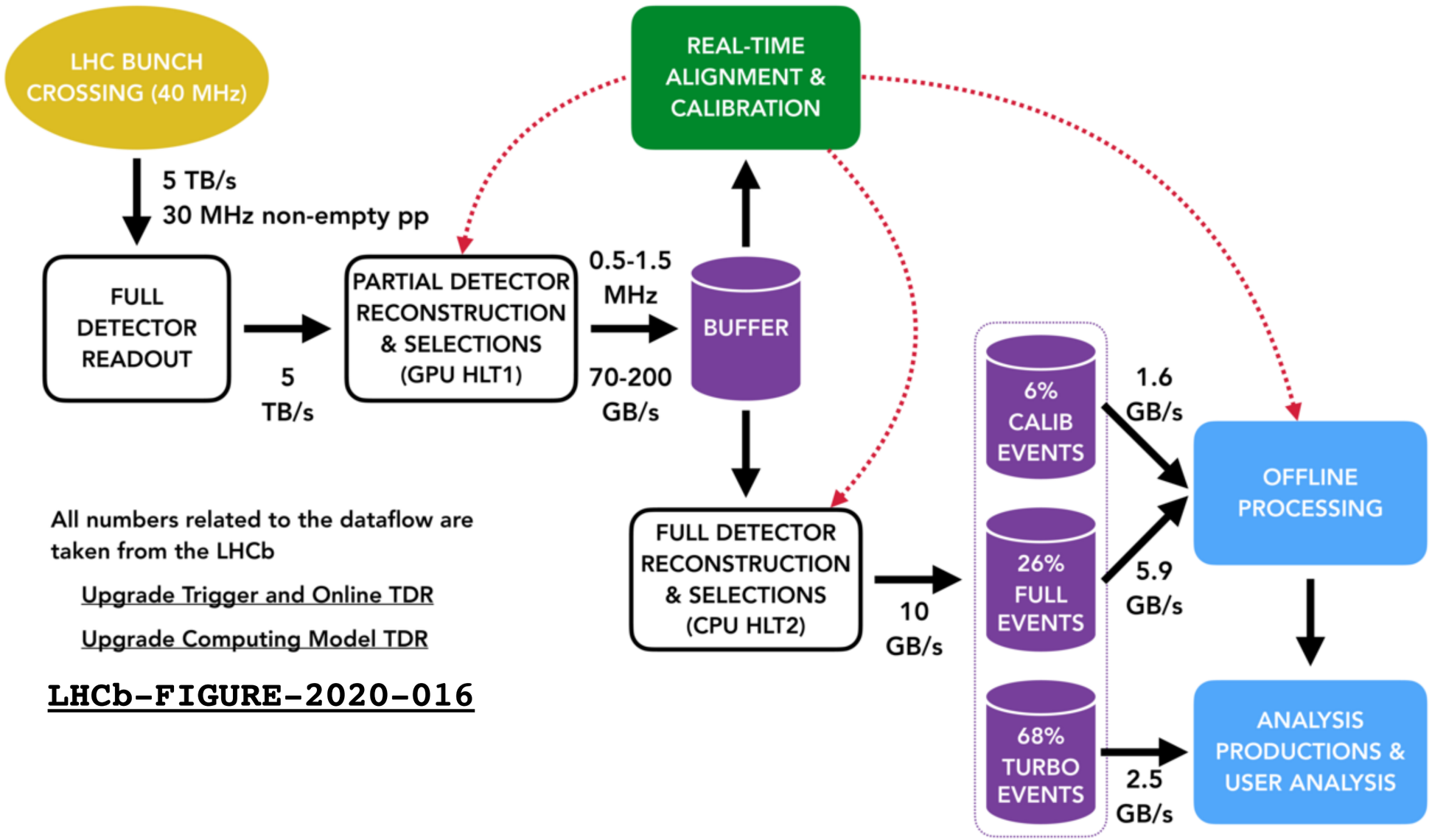
Or in a picture…

Typical triggers select signal needles in Standard Model haystacks
LHCb needs to sort and compresses haystacks of needles — Real Time Analysis!

From this follows the LHCb DAQ design for the upgrade



LHCb upgrade dataflow in more detail



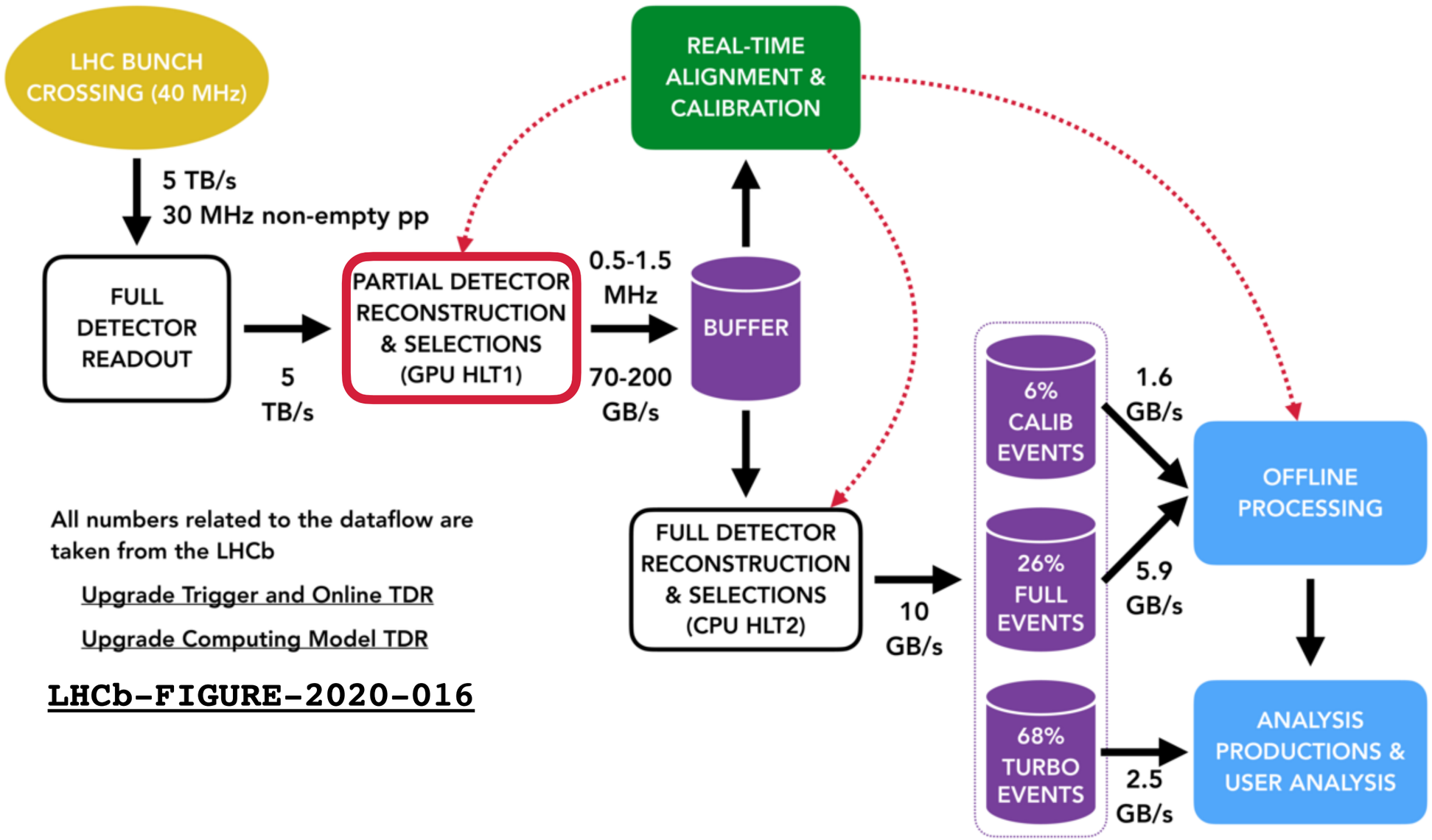
All numbers related to the dataflow are taken from the LHCb

Upgrade Trigger and Online TDR

Upgrade Computing Model TDR

LHCb-FIGURE-2020-016

LHCb upgrade dataflow in more detail



All numbers related to the dataflow are taken from the LHCb

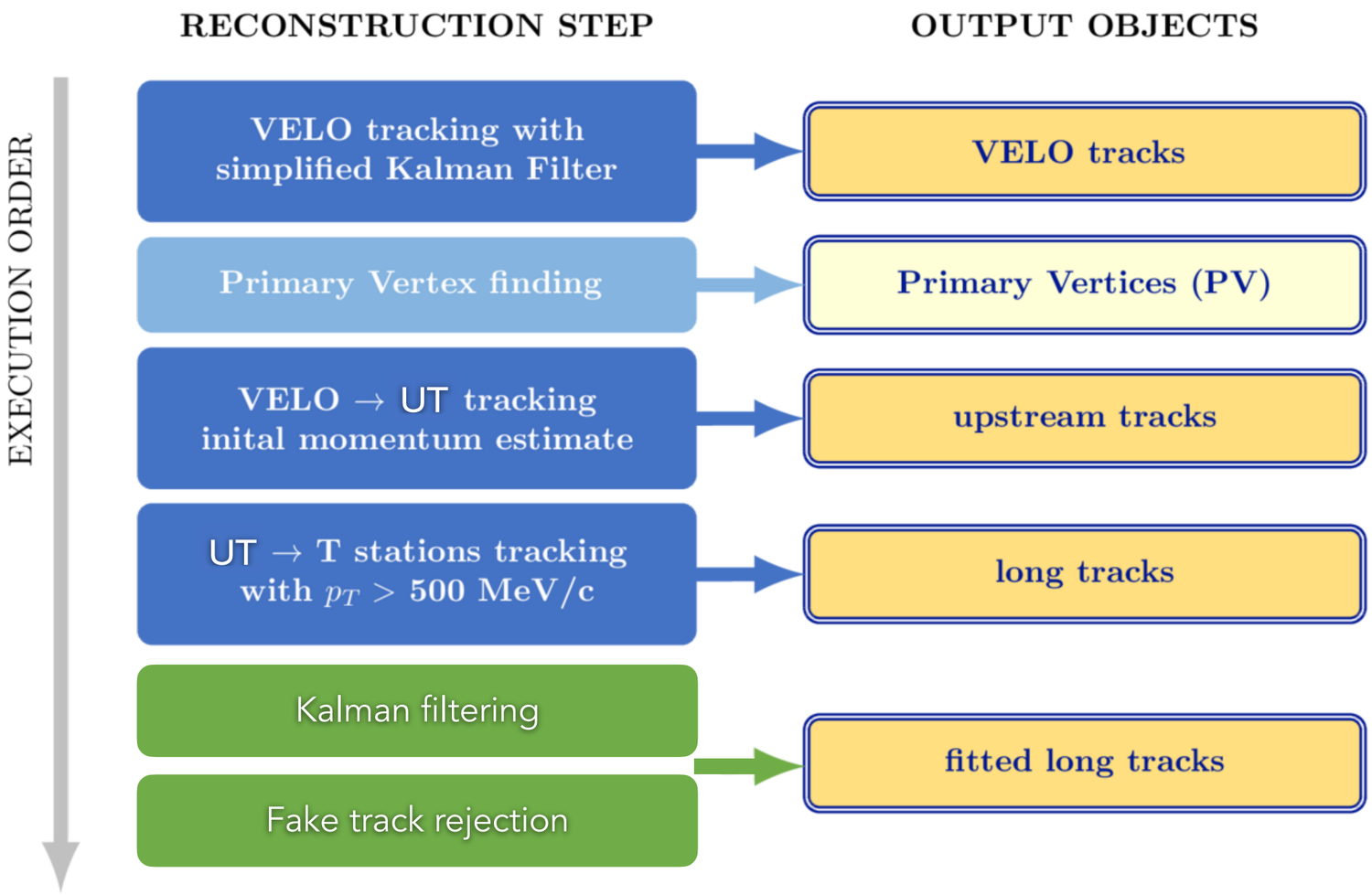
Upgrade Trigger and Online TDR

Upgrade Computing Model TDR

LHCb-FIGURE-2020-016

Need to preselect events before running full reconstruction, but based on which criteria?

Physics content of HLT1 which runs @ 30 MHz



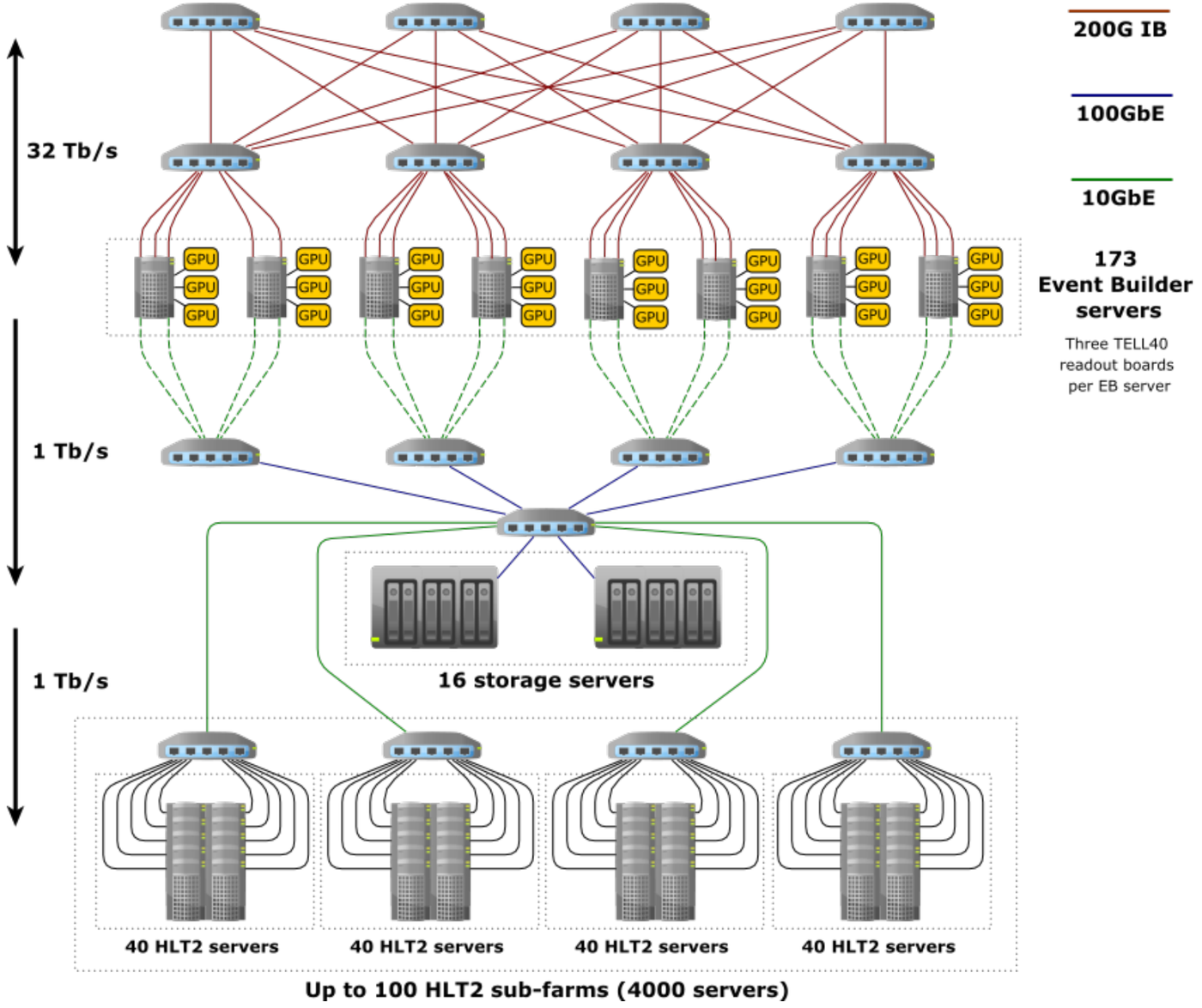
"Traditional" inclusive selections but based on both transverse momentum and displacement
This of course requires charged particles, so we require 30 MHz tracking at $2 \cdot 10^{33}$!

Pause and compare this to ATLAS/CMS HL-LHC processing

CMS detector	LHC Run-2	HL-LHC Phase-2	
Peak ⟨PU⟩	60	140	200
L1 accept rate (maximum)	100 kHz	500 kHz	750 kHz
Event Size	2.0 MB [a]	5.7 MB [b]	7.4 MB
Event Network throughput	1.6 Tb/s	23 Tb/s	44 Tb/s
Event Network buffer (60 seconds)	12 TB	171 TB	333 TB
HLT accept rate	1 kHz	5 kHz	7.5 kHz
HLT computing power [c]	0.5 MHS06	4.5 MHS06	9.2 MHS06
Storage throughput	2.5 GB/s	31 GB/s	61 GB/s
Storage capacity needed (1 day)	0.2 PB	2.7 PB	5.3 PB

The LHCb upgrade HLT must handle the same real-time data volume as ATLAS/CMS HL-LHC HLTs

Exploiting the full detector readout



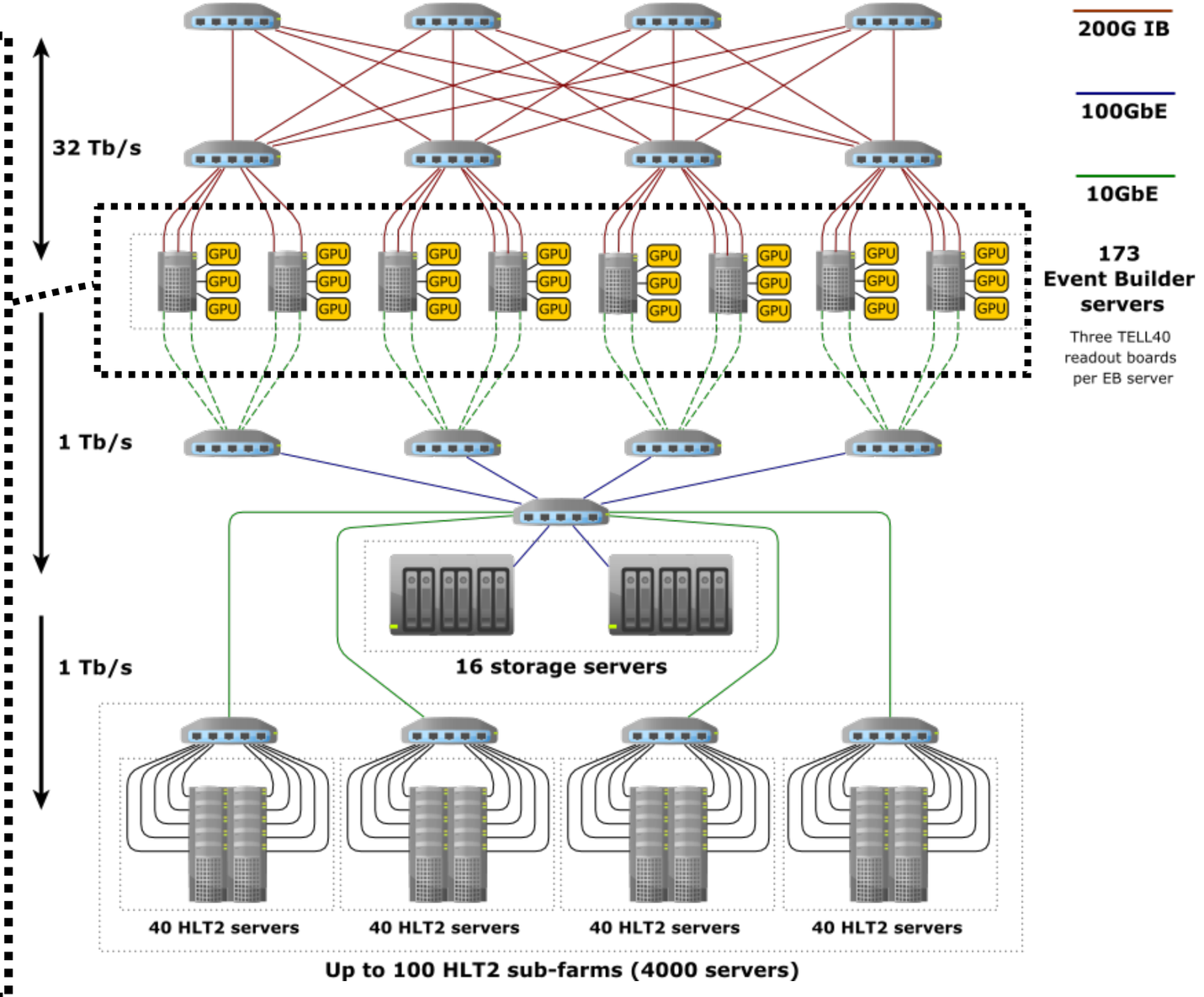
Exploiting the full detector readout

Data from detectors received by O(500) FPGA readout boards and built into events by a farm of 173 servers

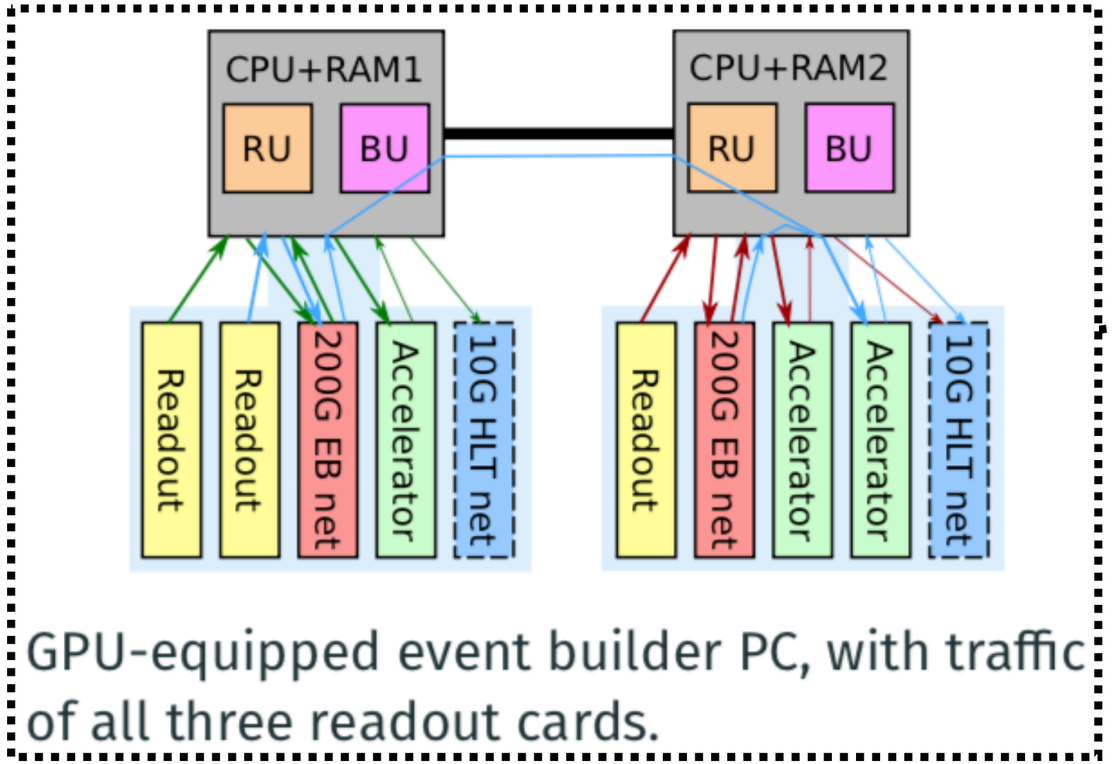
Leaves three PCIe slots free per server

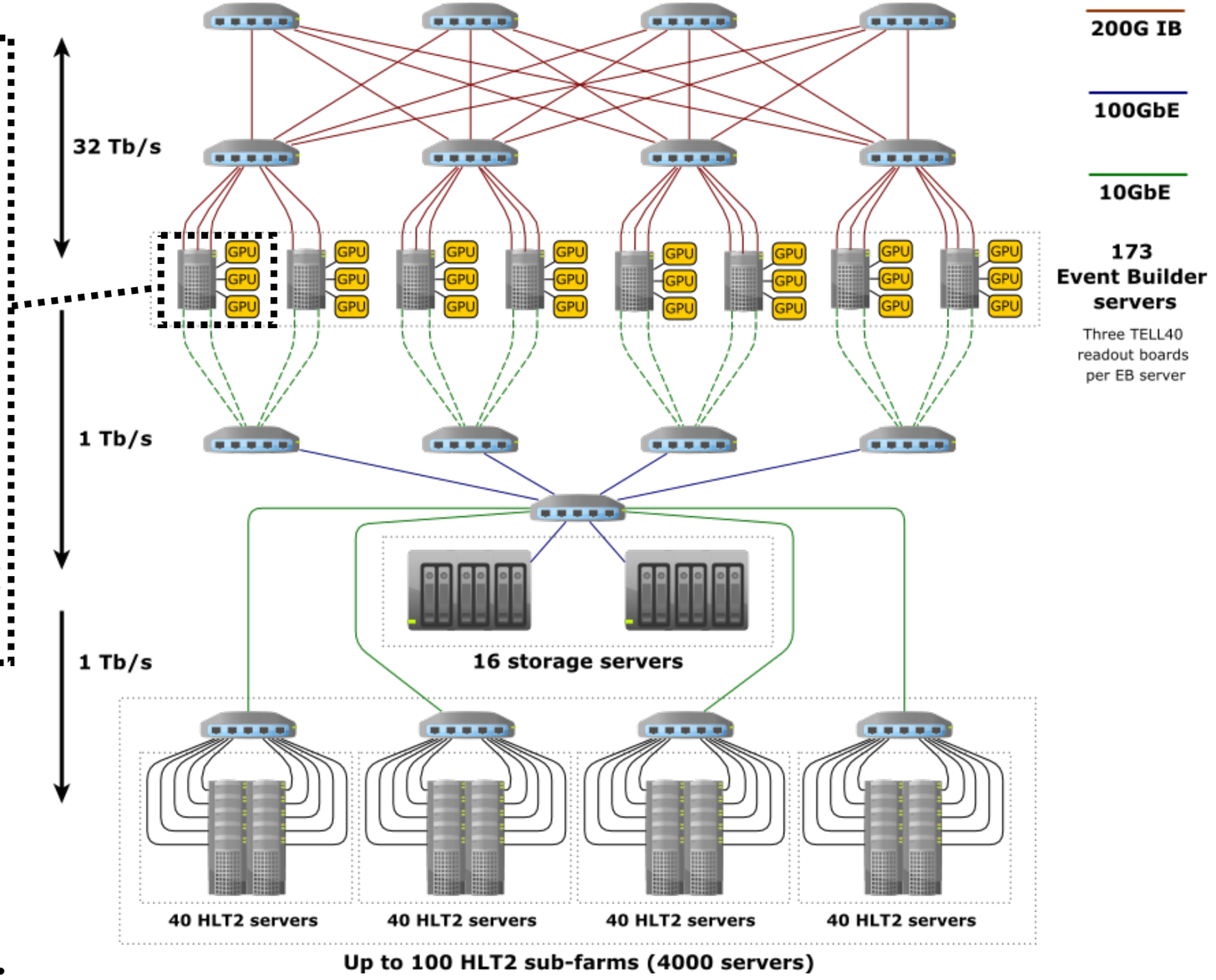
Can now choose: send the full 32 Tb/s to a CPU server farm for processing (requires two extra network cards per server) or fill these slots with co-processors (e.g. GPUs) and reduce the data rate locally to 1 Tb/s

Both options developed and viable — we finally chose the GPU option and I'll talk about it in more detail now.



Architecture of one Event builder node





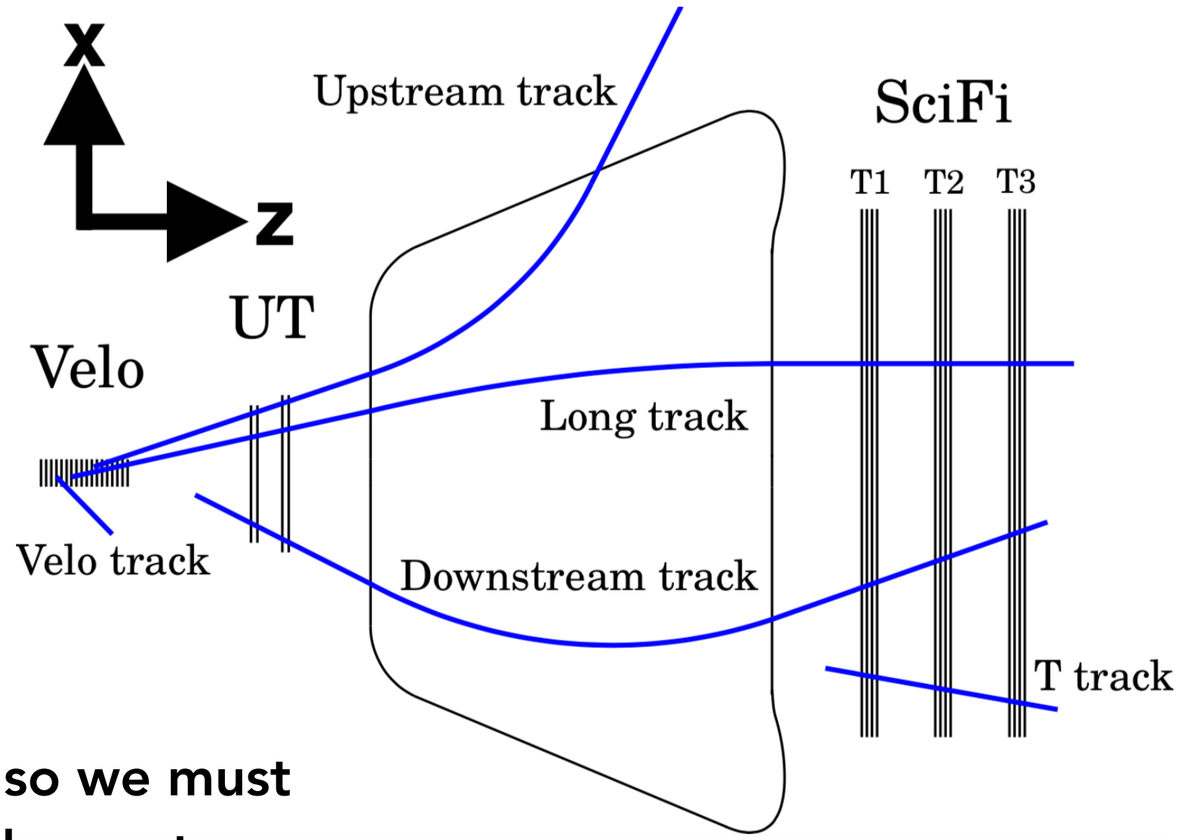
If we can put the whole trigger software on the GPU, then a GPU accelerator is "transparent" to the Event Builder — like sending data to a network card.
Architecture should "automagically" work.

Up to three cards possible — servers have the cooling capacity to deal with it.

So the architecture should work, will the software?

Essential ingredients for HLT1:

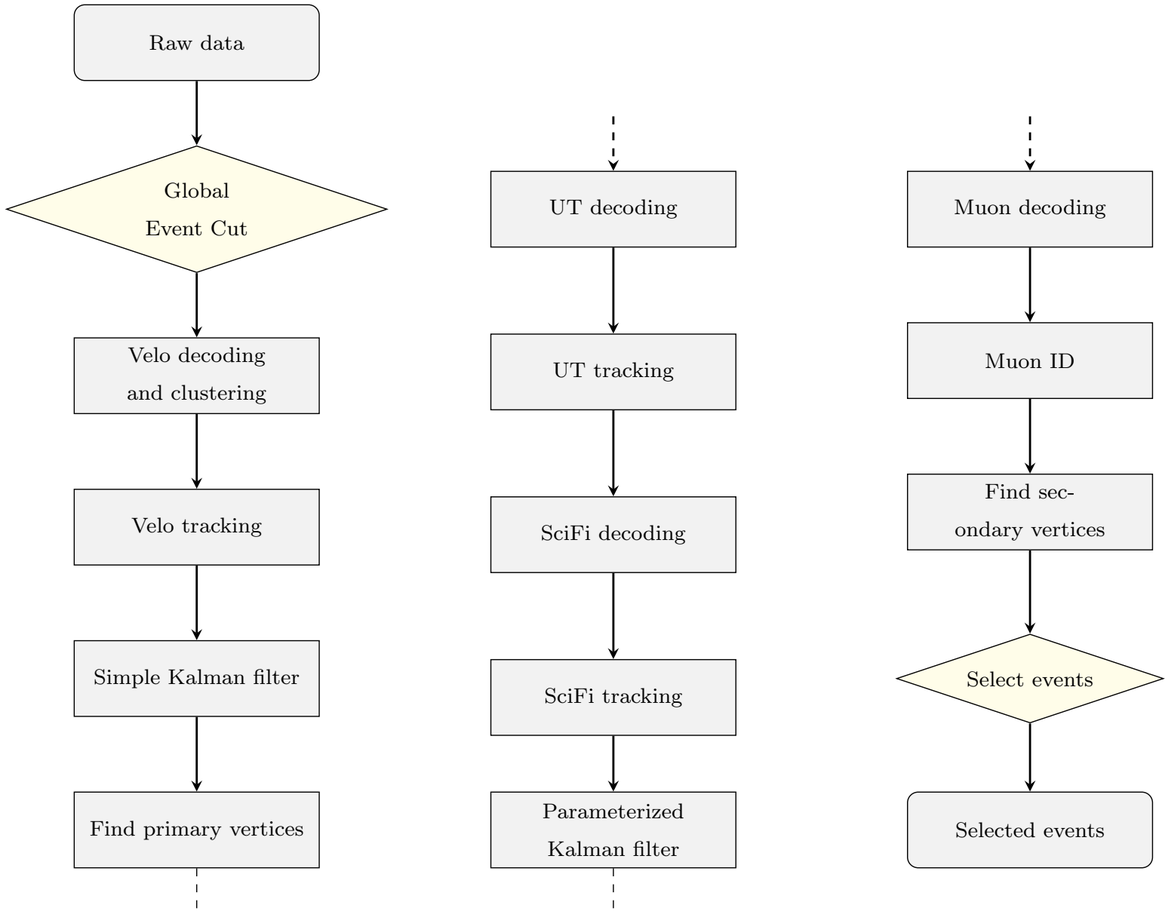
1. Find tracks in the vertex detector
2. Find pp collision points and measure track displacement to them
3. Extrapolate tracks to the UT, and then to the SciFi trackers
4. Perform muon identification and fake track rejection



LHCb events are relatively small, O(100 kB), so we must parallelize both across events and inside each event.

Can we achieve enough parallelism to do the job?

Spoiler alert: we indeed can



The full sequence contains dozens of components and can accomodate up to O(100) selections
Most algorithms written from scratch for GPU with a logic optimized for SIMT — not ported
Cross-architecture by construction, compiles for CPU to enable "for free" simulation

Interlude on managing software: this began as pure R&D

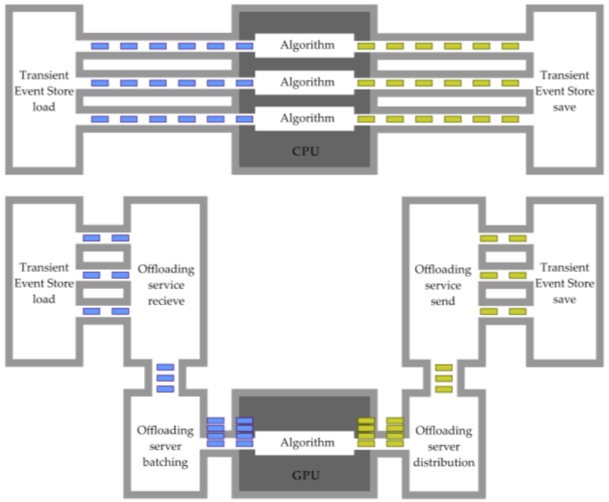
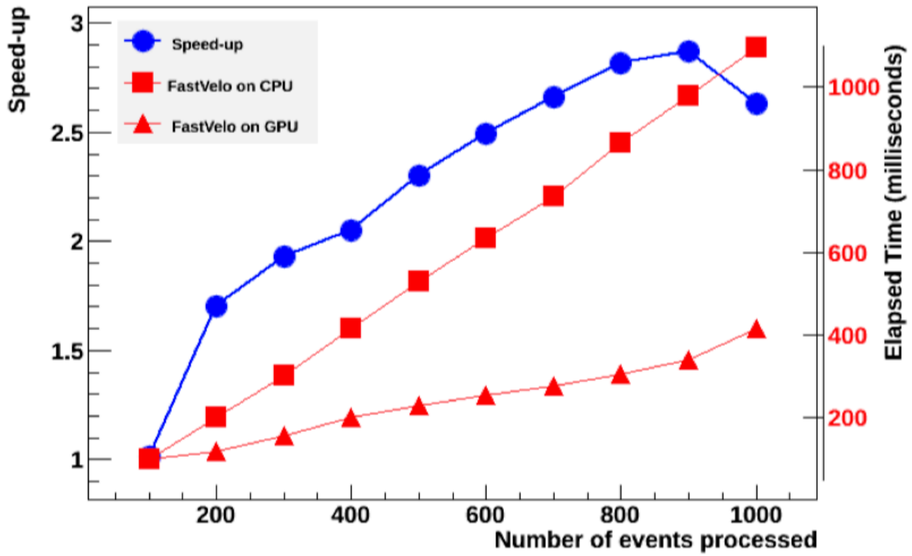
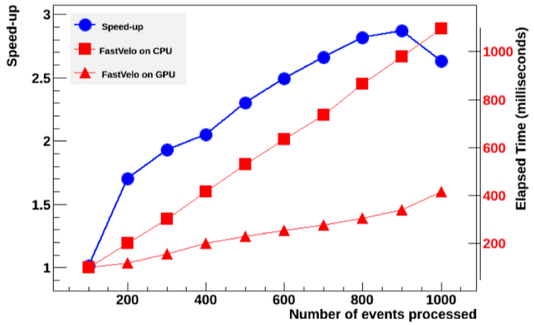


Figure 2 Execution of multiple Gaudi pipelines concurrently using a classical CPU algorithm vs. offloading to a GPU.



LHCb had been pursuing individual GPU reconstruction algorithms since 2014, with the most promising work done on the vertex detector reconstruction algorithm and associated infrastructure (see biblio at bottom).

By 2017 we had largely concluded this would never work



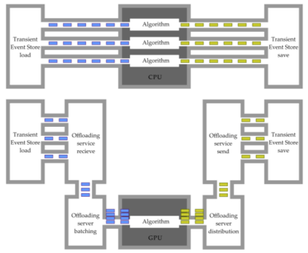


Figure 2 Execution of multiple Gaudi pipelines concurrently using a classical CPU algorithm vs. offloading to a GPU.

Table 1. Comparison of a CPU and a GPU VELO Pixel tracking algorithm.

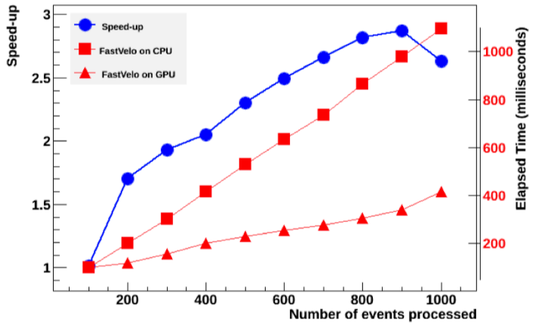
	PrPixel	Track forwarding	
Time per event (ms)	3.6	26.2	batch of 1
		3.5	batch of 40
		2.0	batch of 100
		0.80	batch of 300
Ghost rate	1.7%	0.8%	
Efficiency for long tracks	98.3%	98.0%	
Efficiency for long tracks over 5 GeV	98.8%	98.4%	

Nota bene: compares GPU to a single CPU core!

2014 2015 2016 2017 2018 2019 2020

However porting single algorithms to GPUs was not going to work, mainly because no single algorithm took a large enough piece of the reconstruction sequence to make this cost-effective.

Then we decided to give the architecture a fair chance...



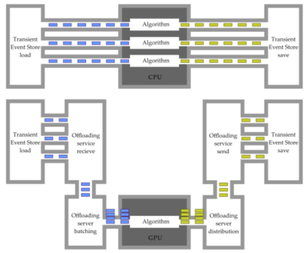
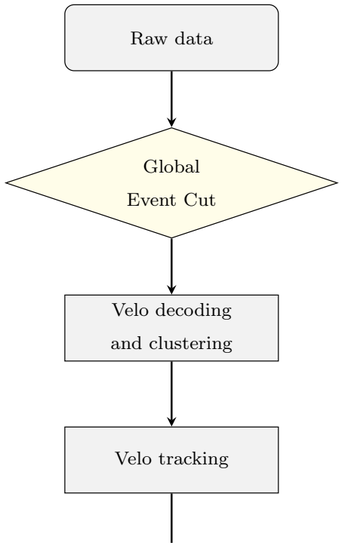


Figure 2 Execution of multiple Gaudi pipelines concurrently using a classical CPU algorithm vs. offloading to a GPU.

Table 1. Comparison of a CPU and a GPU VELO Pixel tracking algorithm.

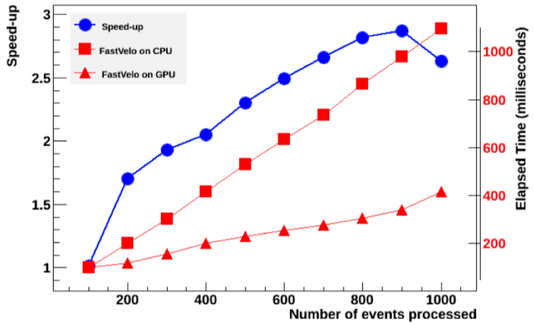
	PrPixel	**Track forwarding**	
Time per event (ms)	3.6	26.2	batch of 1
		3.5	batch of 40
		2.0	batch of 100
		0.80	batch of 300
Ghost rate	1.7%	0.8%	
Efficiency for long tracks	98.3%	98.0%	
Efficiency for long tracks over 5 GeV	98.8%	98.4%	



2014 2015 2016 2017 2018 2019 2020

At the start of 2018 we decided to try to put the entire HLT1 on GPUs, despite only having a functioning vertex detector reconstruction and two years to get the job done. We hedged our bets, which seemed expensive from the point of view of developer time but in fact made optimal use of people's diverse skills.

And learned that it can be easier to achieve the harder goal



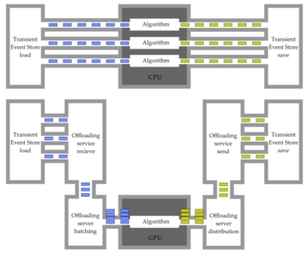
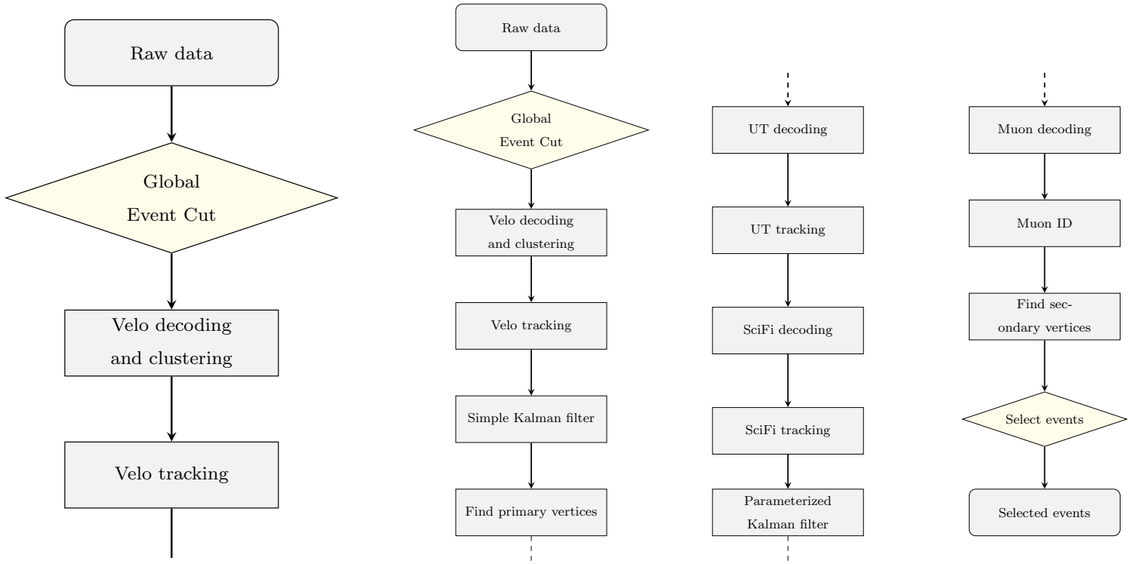


Figure 2 Execution of multiple Gaudi pipelines concurrently using a classical CPU algorithm vs. offloading to a GPU.

Table 1. Comparison of a CPU and a GPU VELO Pixel tracking algorithm.

	PrPixel	Track forwarding	
		batch of 1	batch of 40
Time per event (ms)	3.6	26.2	3.5
		batch of 100	batch of 300
		2.0	0.80
Ghost rate	1.7%	0.8%	
Efficiency for long tracks	98.3%	98.0%	
Efficiency for long tracks over 5 GeV	98.8%	98.4%	

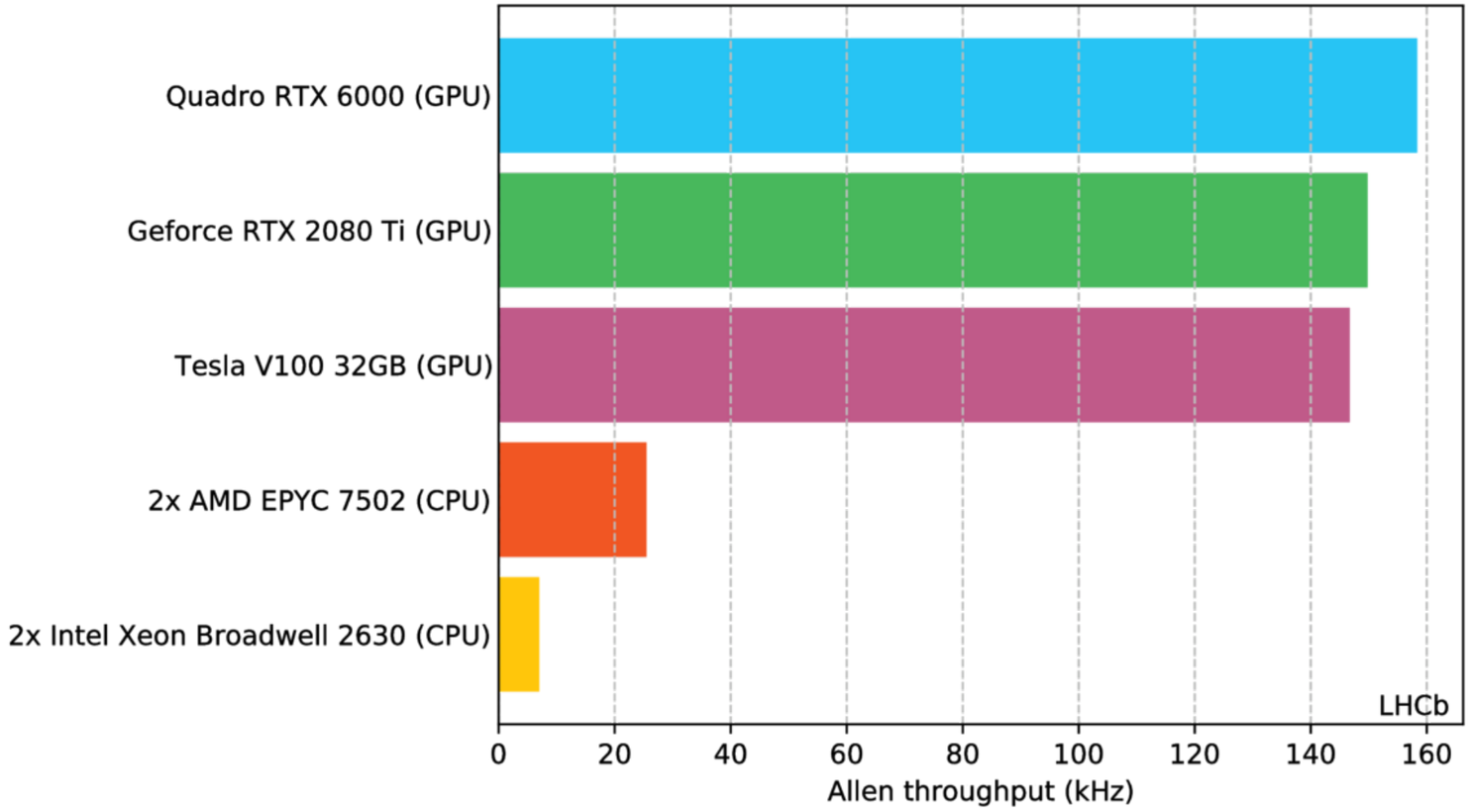


2014 2015 2016 2017 2018 2019 2020

Putting "everything" on the GPU unlocked the power of the architecture and made it cost-effective. Classic accumulation of knowledge on a plateau followed by a phase transition as it came together. Similarly, the vectorization of our CPU reconstruction also came together in parallel to meet the required performance.

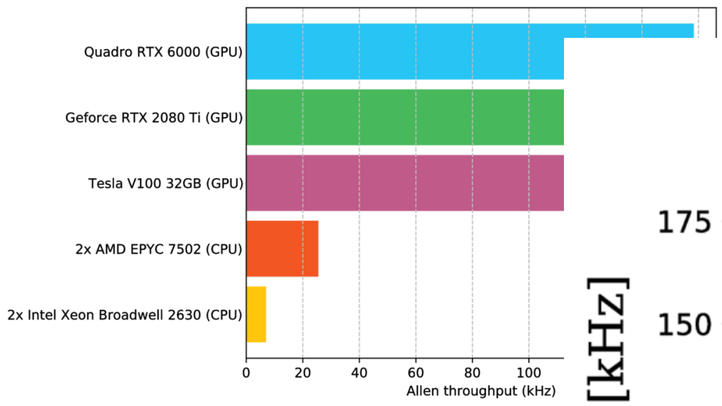
Bibliography: Badalov et al, Badalov et al, Badalov et al, Campora et al, GPU HLT TDR

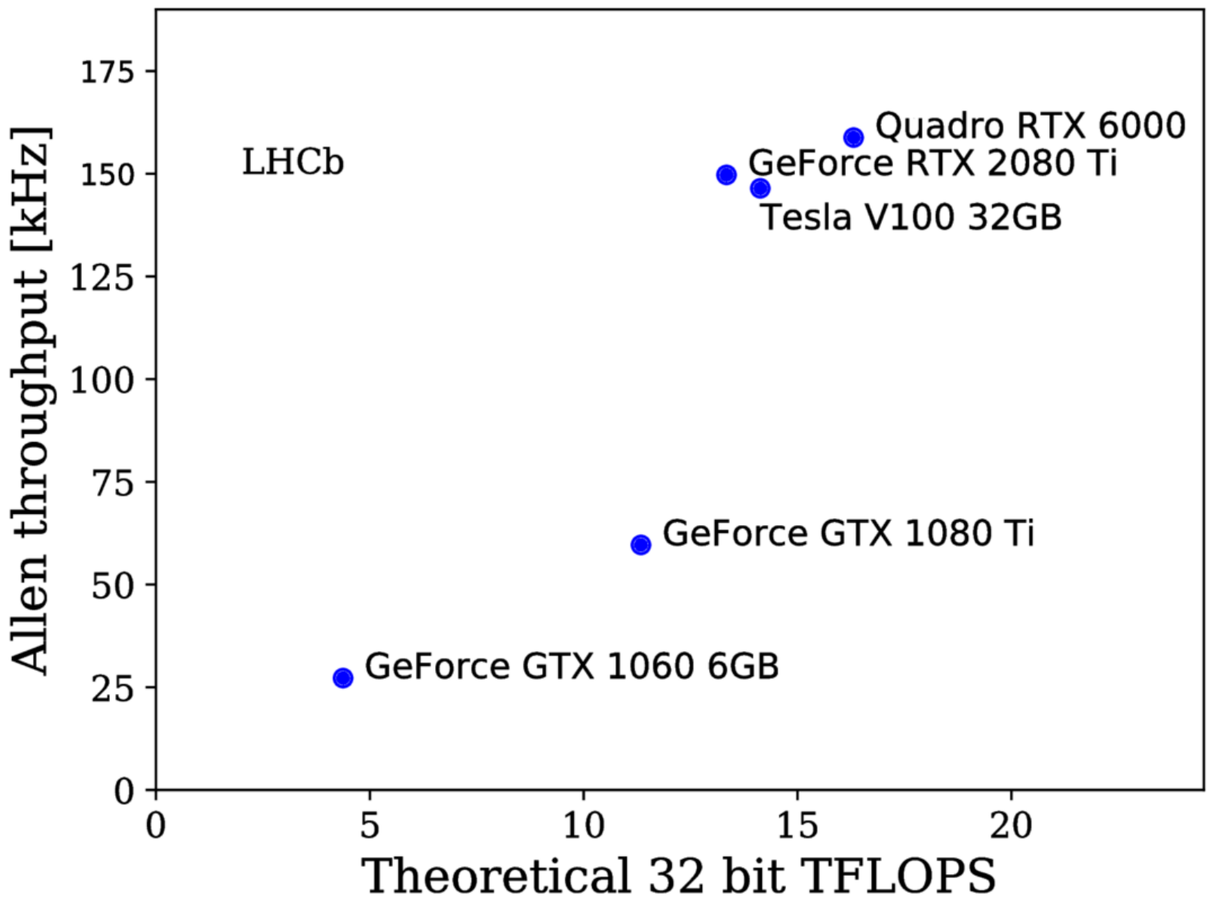
HLT1 throughput performance



O(200) GPUs required to reach 30 MHz so there is plenty of spare capacity!

HLT1 throughput performance

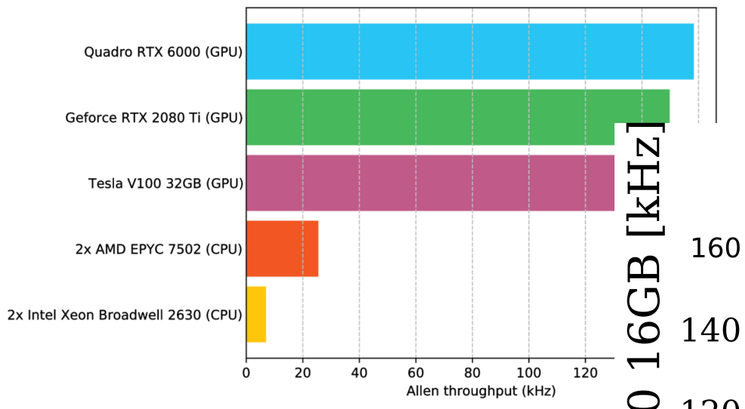


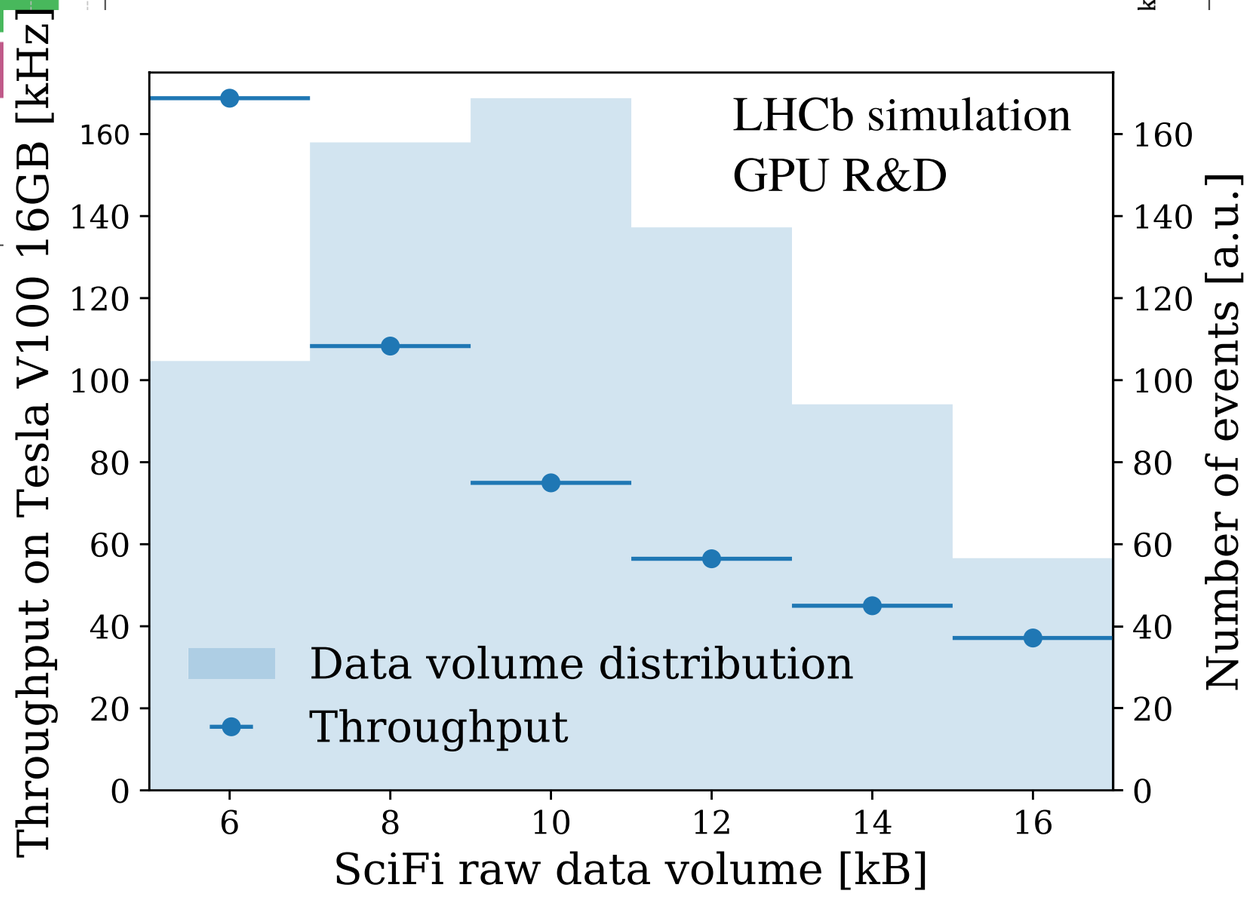


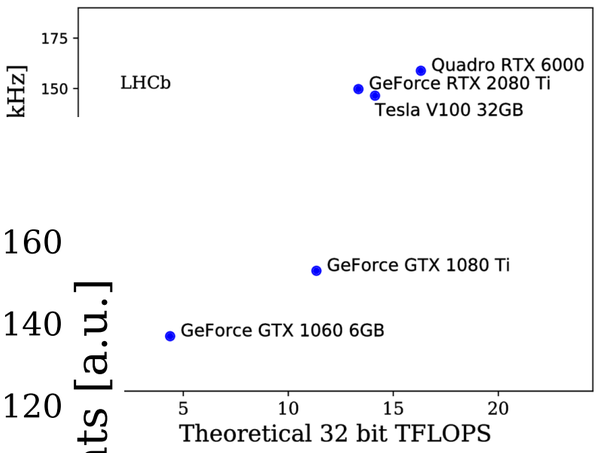
Excellent throughput scaling with theoretical TFLOPS of GPU card

References: LHCb-FIGURE-2020-014

HLT1 throughput performance

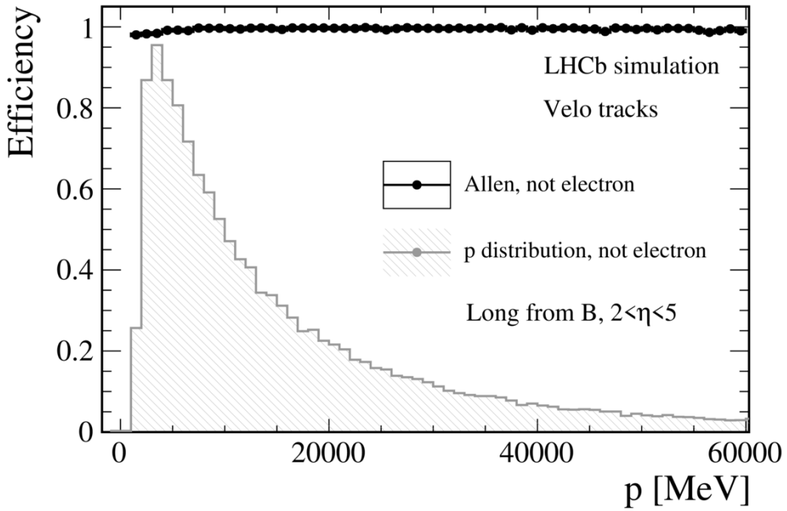


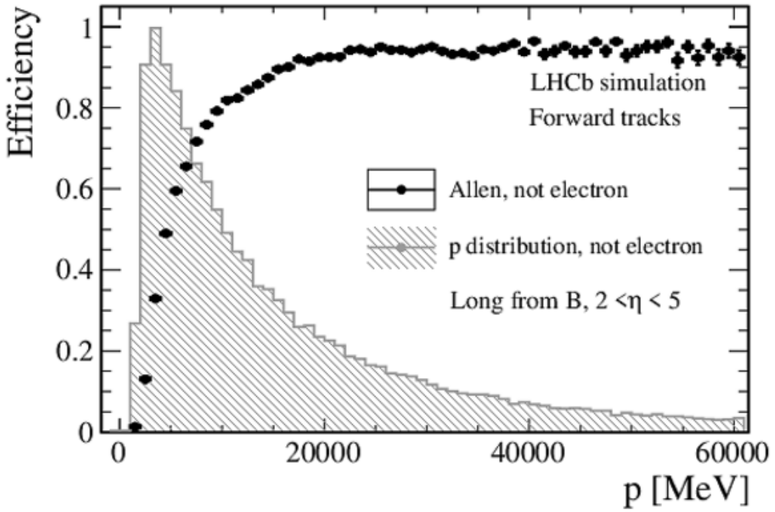


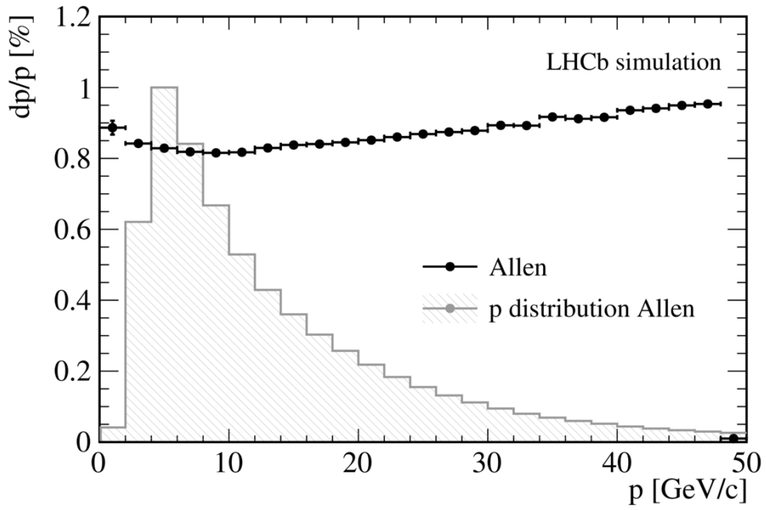


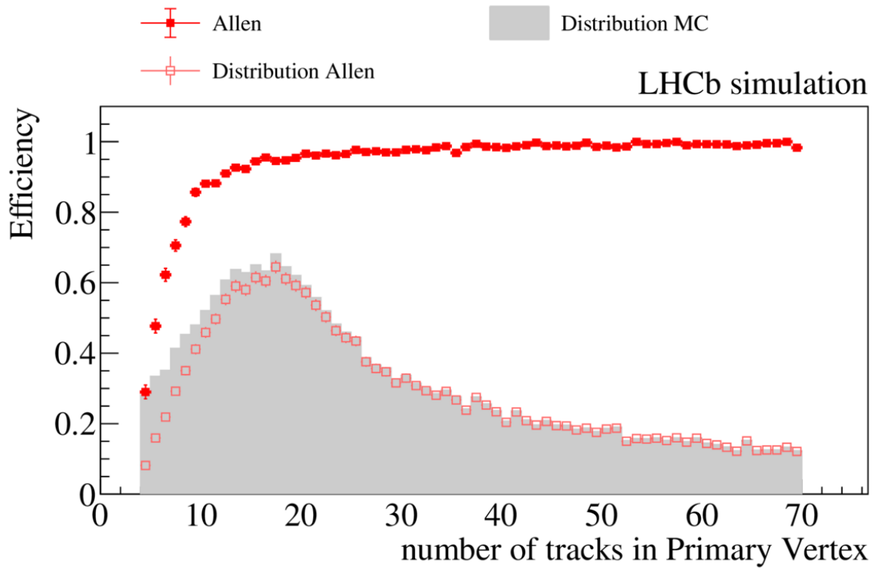
Throughput plateaus with increasing occupancies rather than falling off a cliff

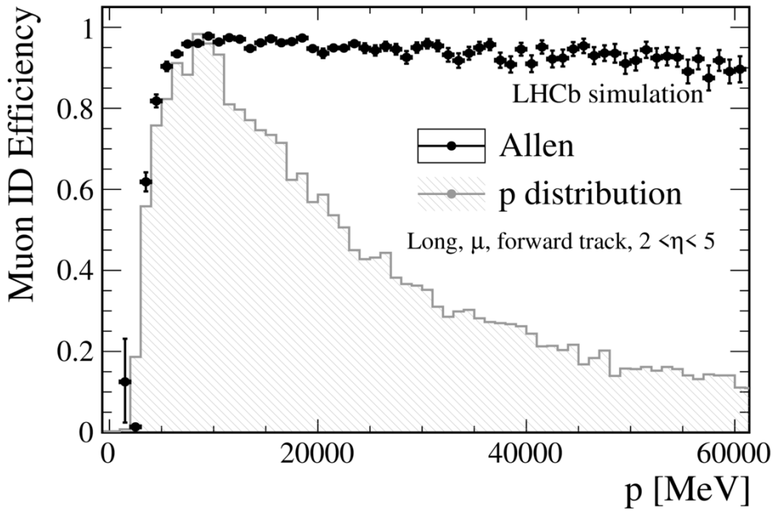
HLT1 reconstruction performance





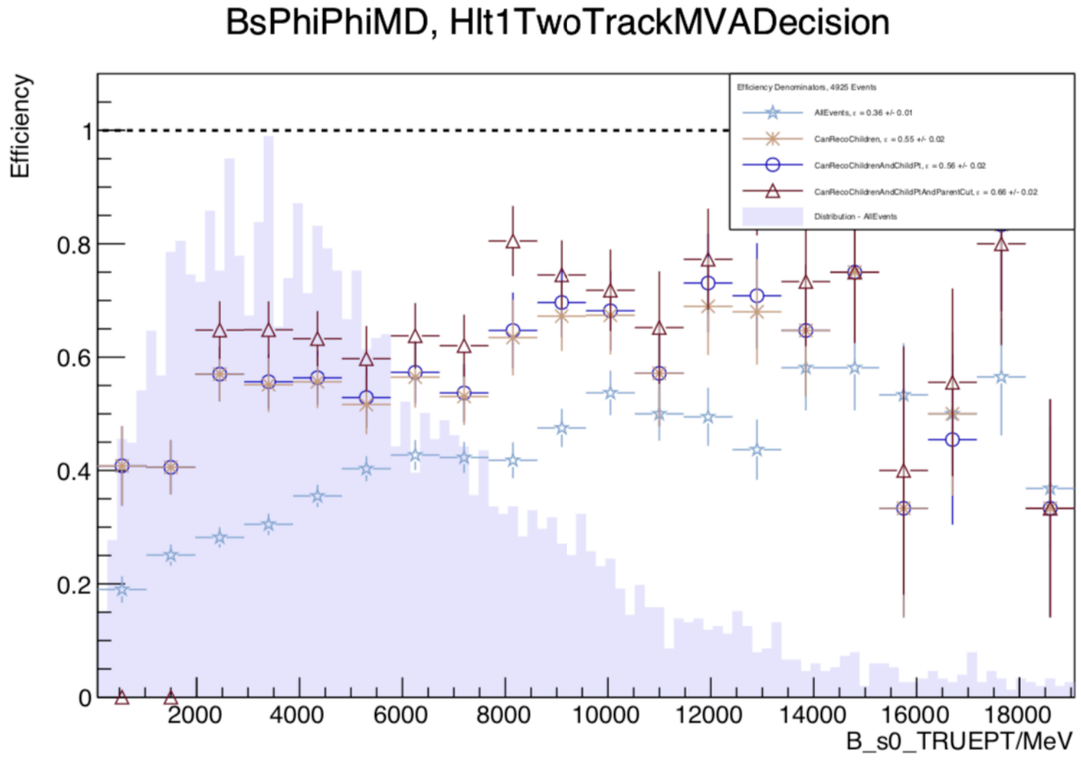


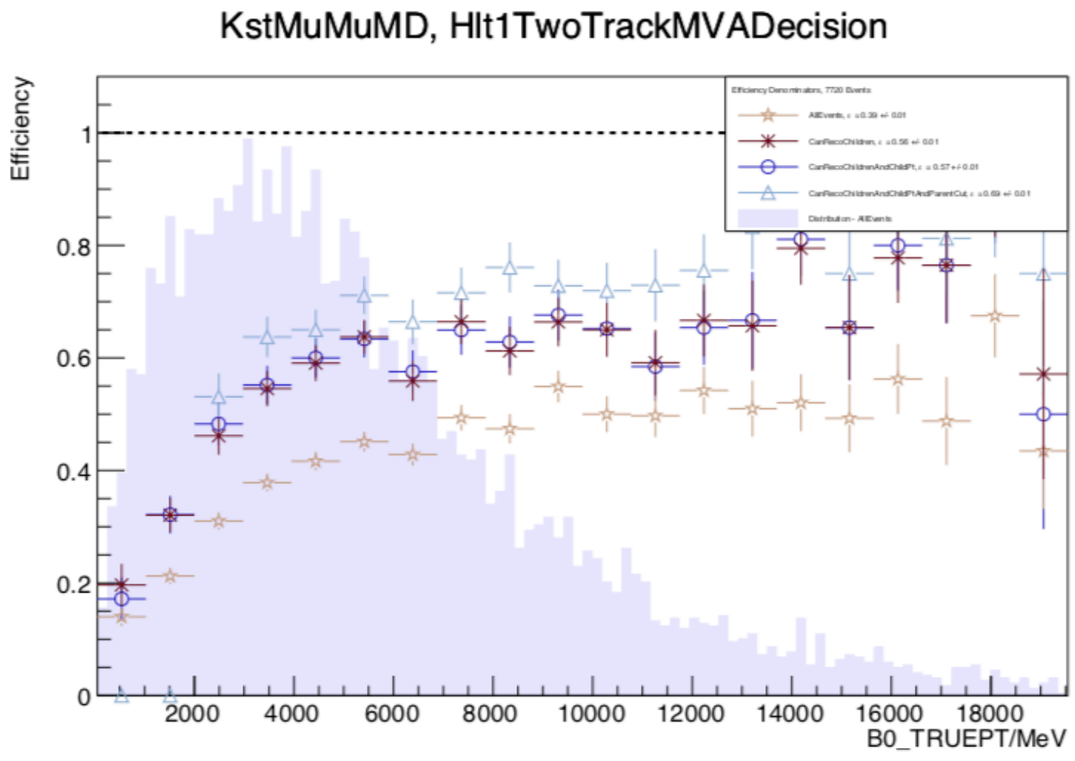




References: LHCb-FIGURE-2020-014

HLT1 selection performance @ ~1 MHz output rate

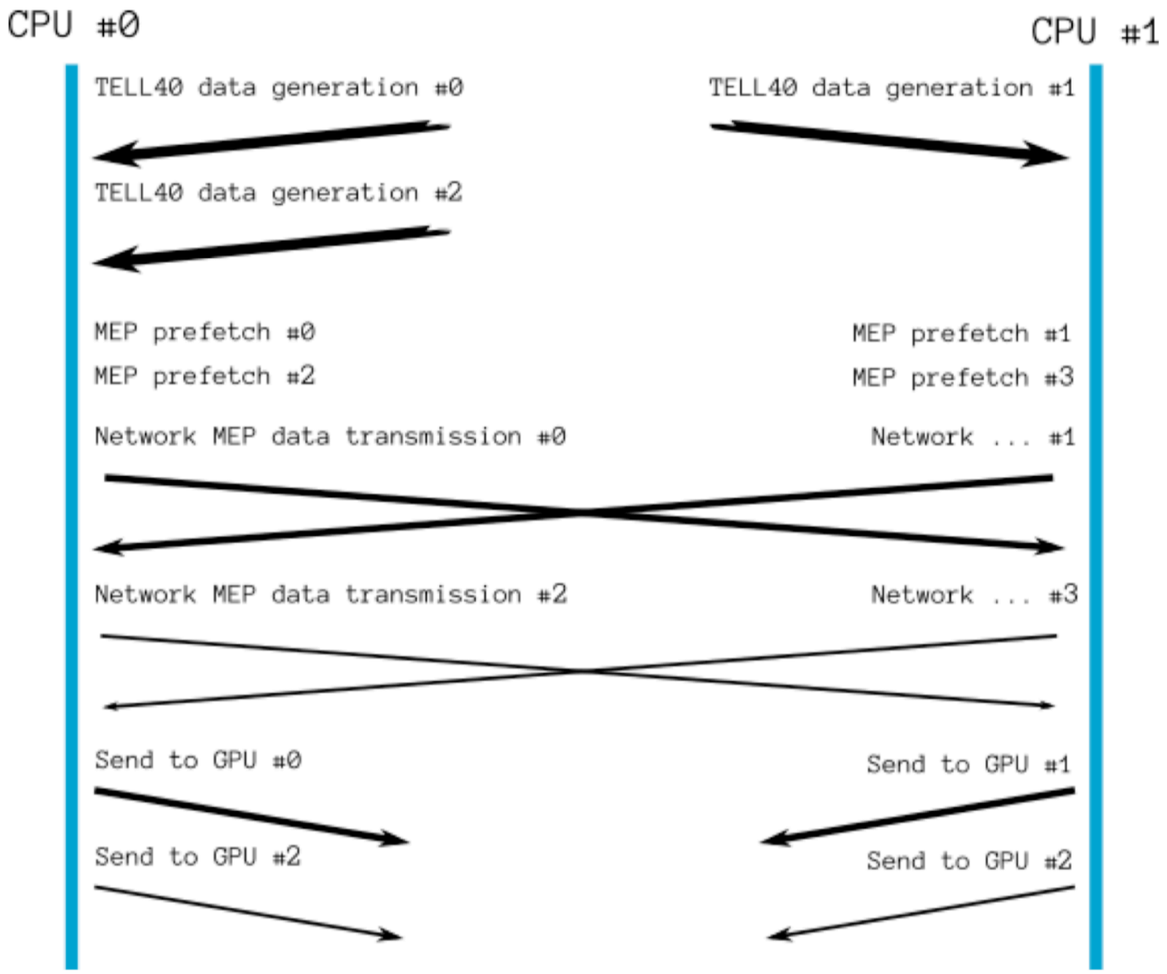




Selections nowhere near tuned — of course can only happen once we've commissioned the all-new detector hardware

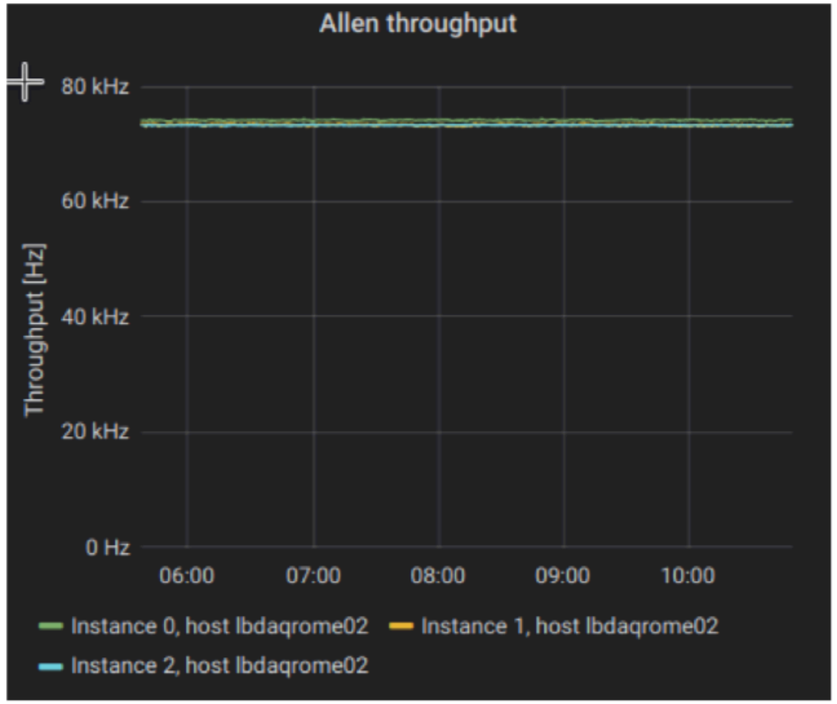
On MC keep > 50% of all reconstructible key B decays with some reasonable parent/child transverse momentum. More than good enough for now!

Integration test — can this be used for stable datataking?

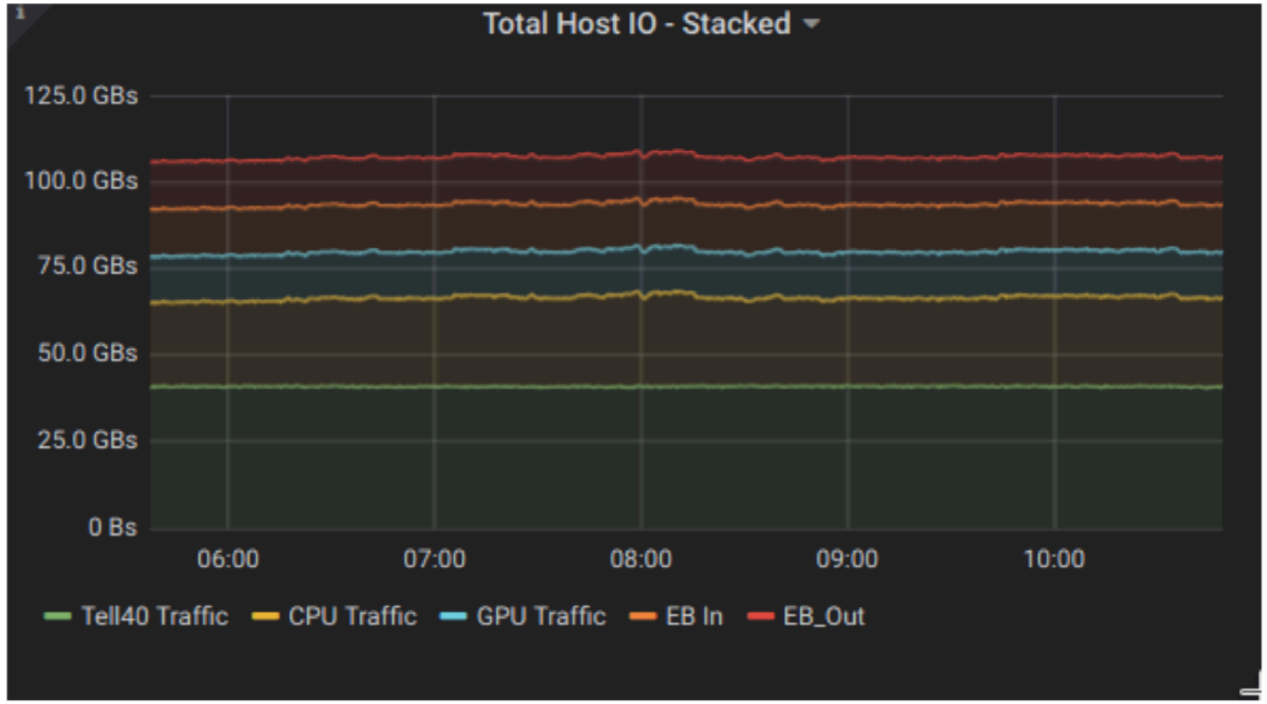


Emulate network traffic & memory pressure by getting FPGA boards to generate data.

Integration test – can this be used for stable datataking? Yes!



(a) Allen throughput



(b) Memory I/O

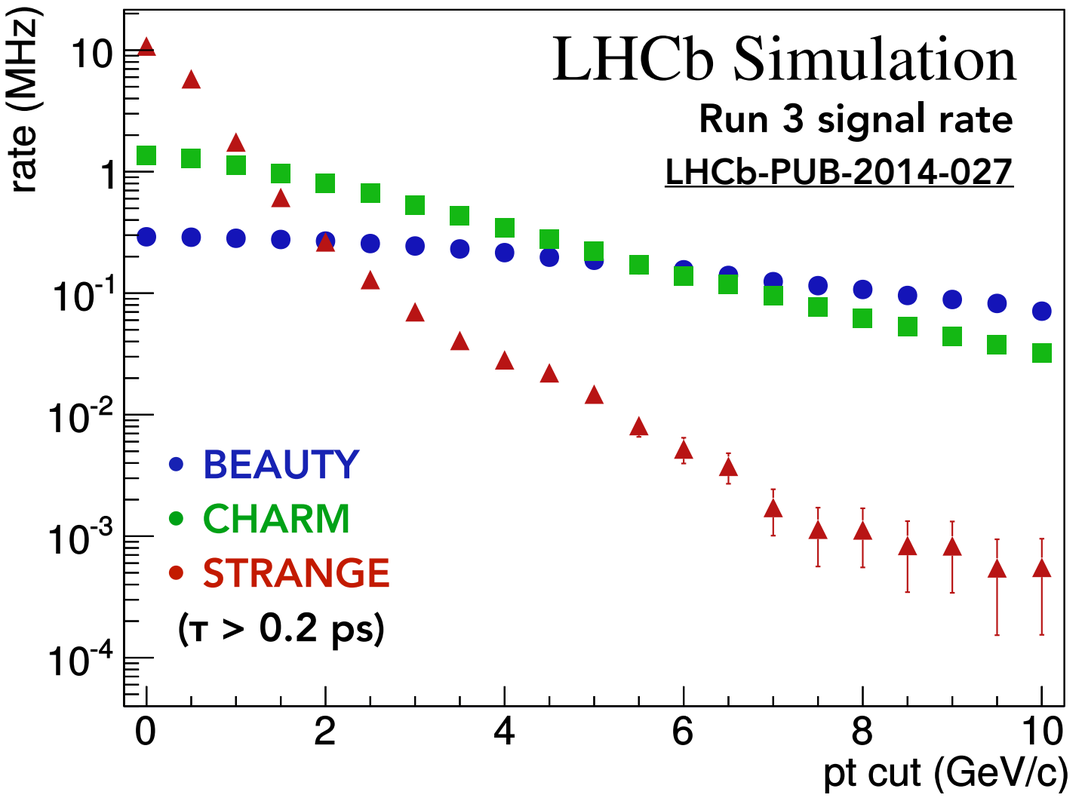
Conclusion and outlook

LHCb is ready to tackle the Run 3 signal deluge with a first-level trigger fully implemented on GPUs

Allen framework is not specific to LHCb, aim to decouple framework and LHCb-specific code in future.

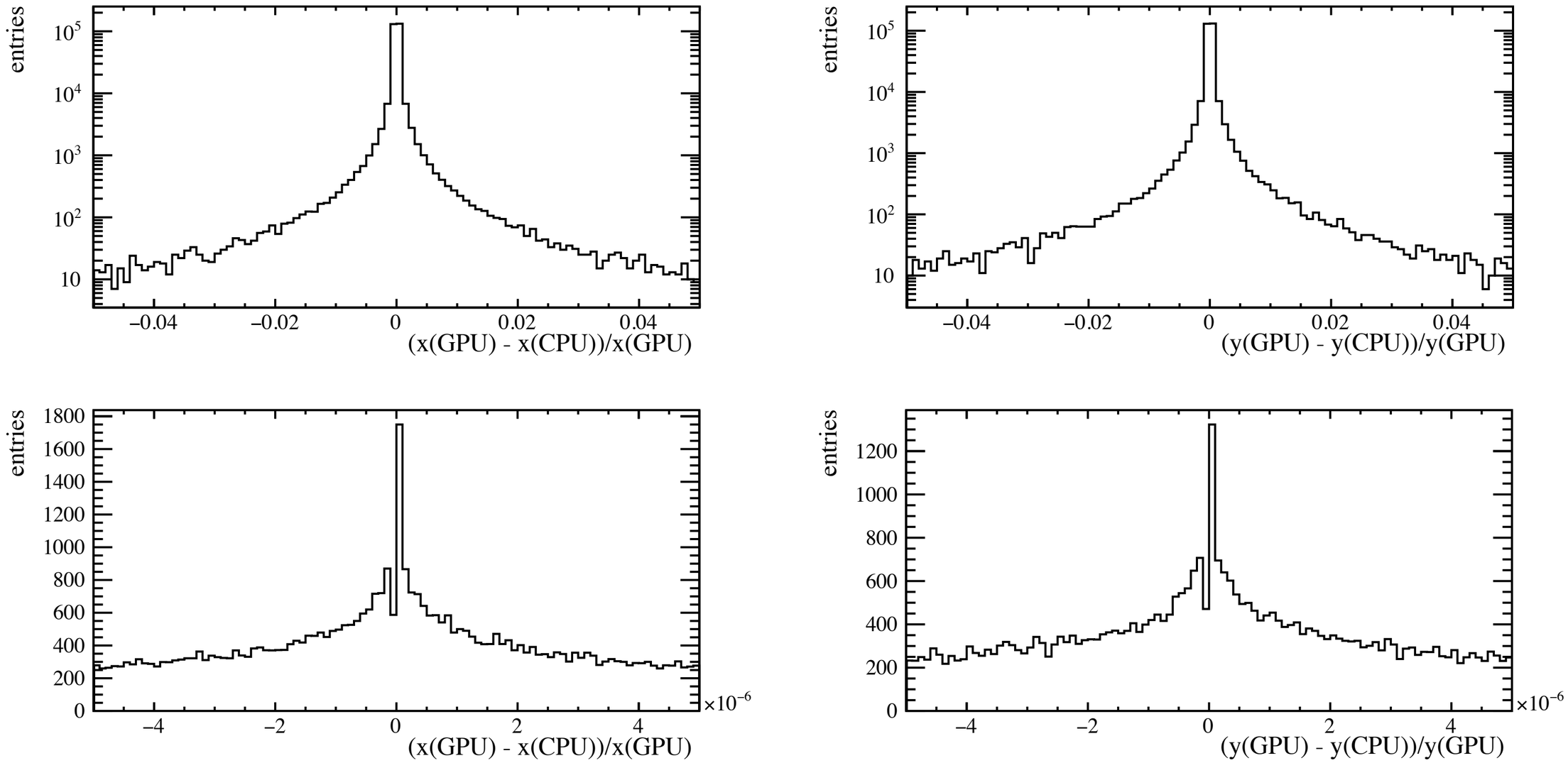
Work ongoing to implement further algorithms on GPU and extend the reach of the HLT for neutral and long-lived particles

Looking forward to commissioning!



Backup

Cross-architecture differences



Comparison of track states executing Allen on GPU and CPU — for vast majority of tracks agreement is at permille level or better. Same is true for most other quantities and we explicitly test for this.

References: LHCb Upgrade GPU High Level Trigger TDR