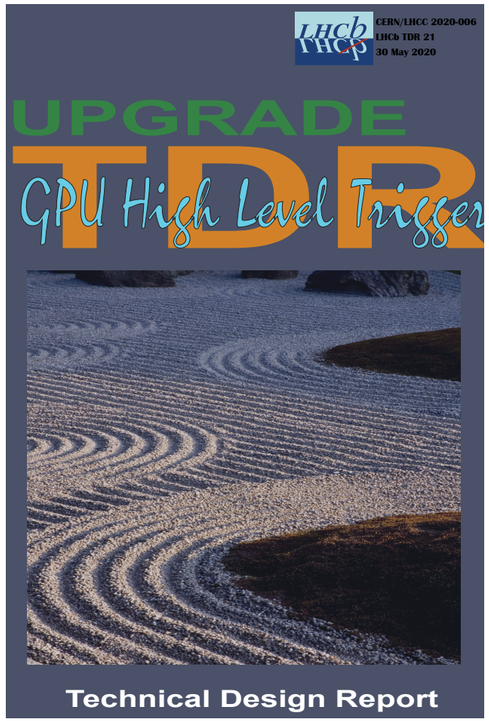
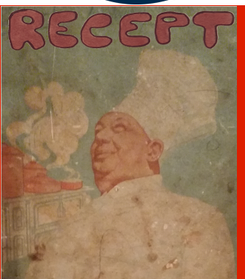
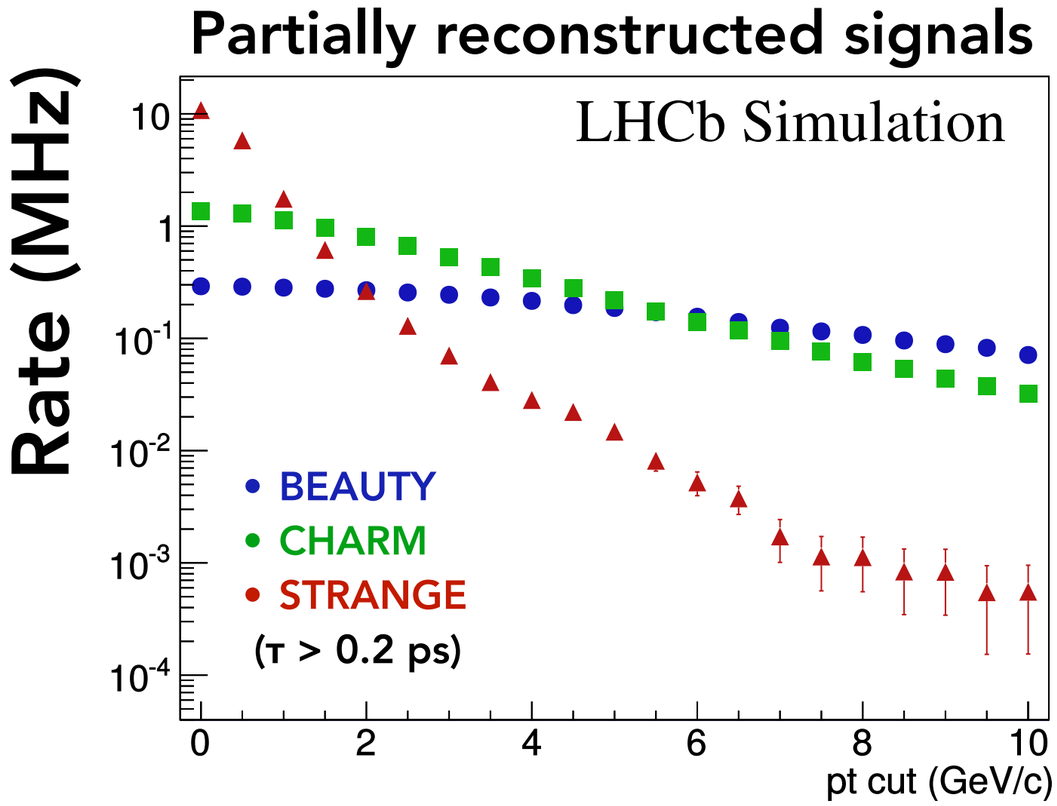
Allen: a complete cross-architecture first-level trigger (& framework)





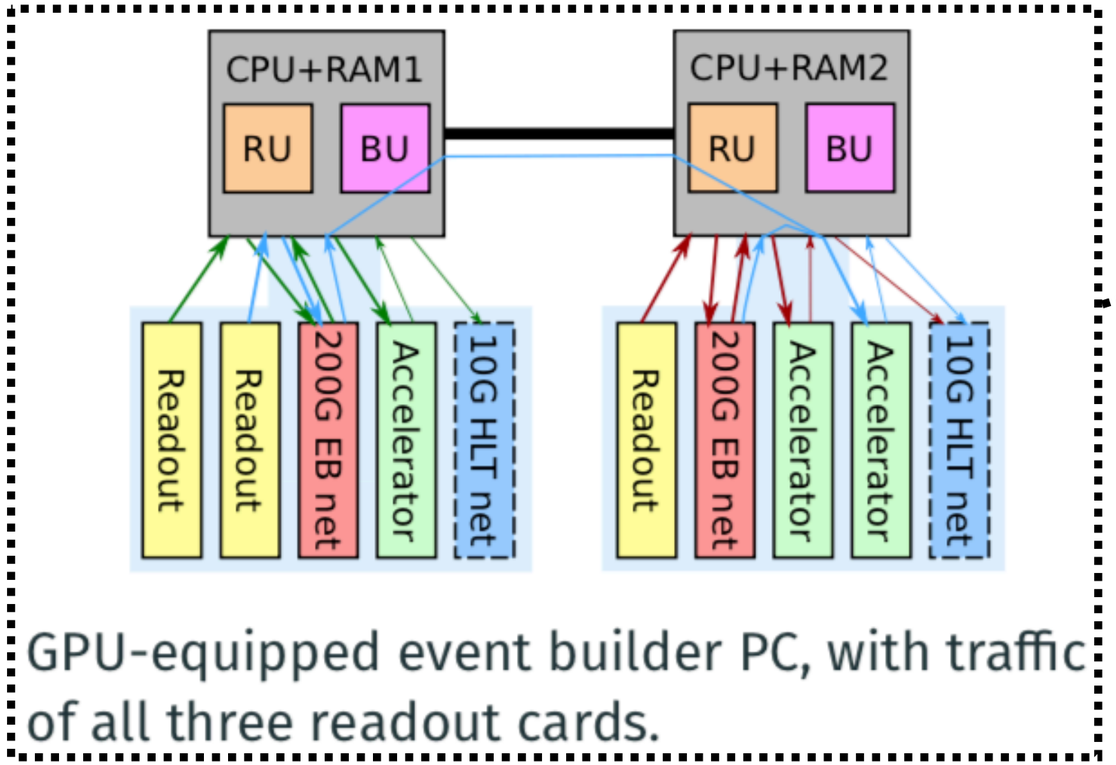
V. V. Gligorov, CNRS/LPNHE
On behalf of the LHCb RTA & Online teams
CERN OpenLab Technical Workshop, 21.03.2022

The challenge of the LHCb upgrade in one slide

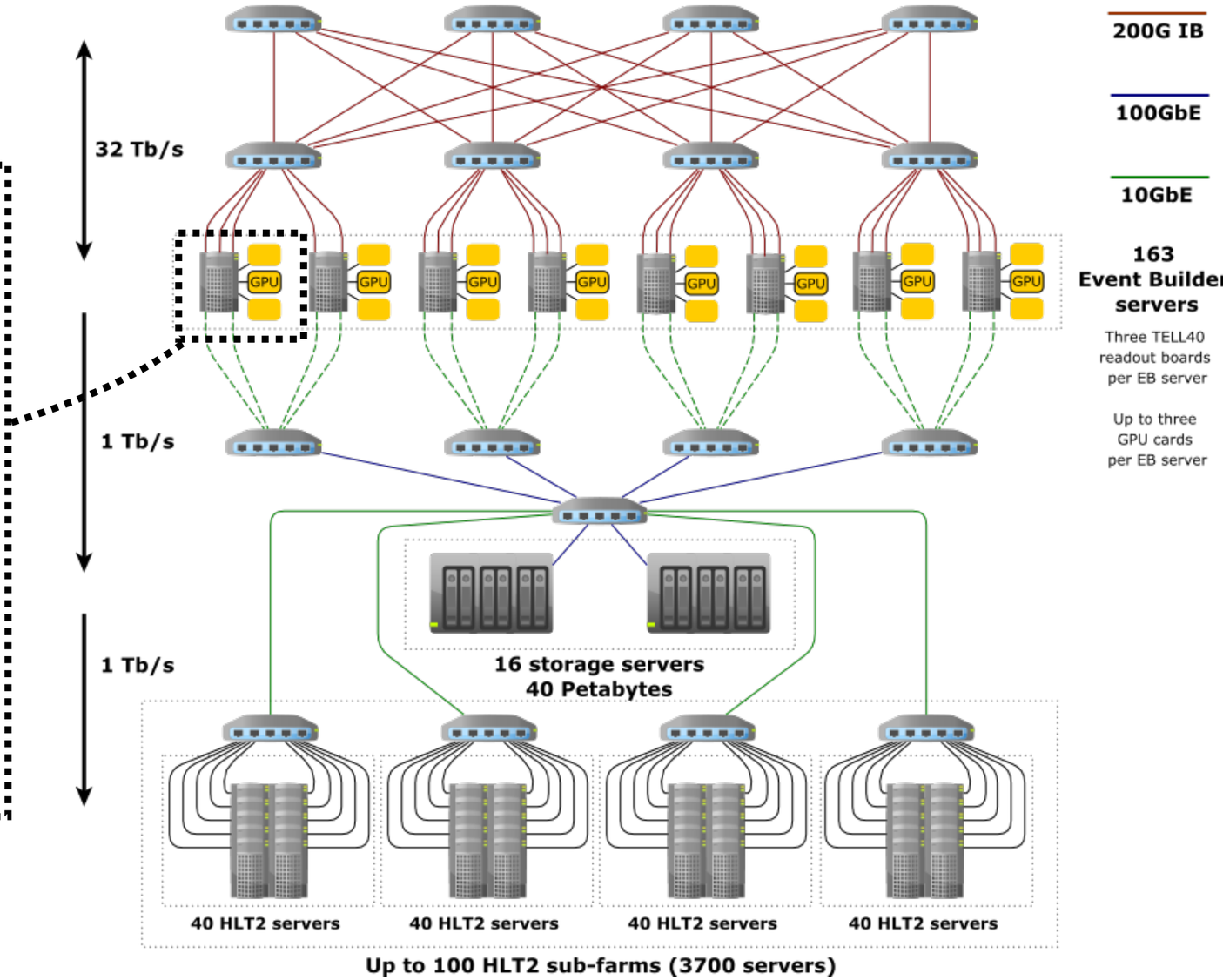


We will have MHz of signals in our acceptance!
We can only afford to fully store 50 kHz of events

From this follows the LHCb DAQ design for the upgrade

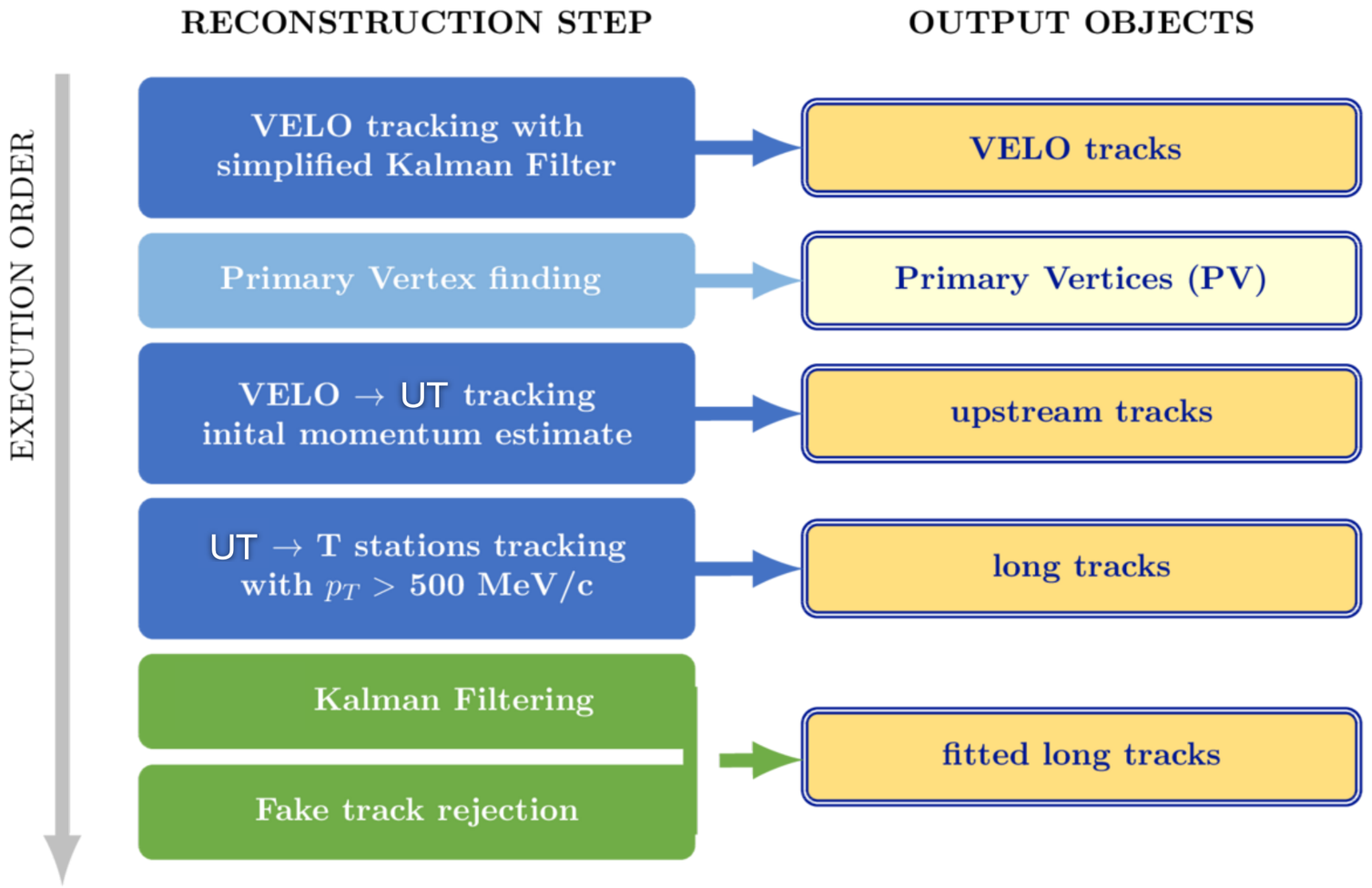


GPU-equipped event builder PC, with traffic of all three readout cards.



32 Tbit/s full event building & processing in a data centre

What is the physics content of HLT1 which runs @30 MHz?



"Traditional" inclusive selections selecting bunch crossings.
Based on charged particles, so 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 has to handle the same data volume in real-time as ATLAS/CMS HL-LHC upgrades! But earlier and for less money...

Allen is not just one or two algorithms

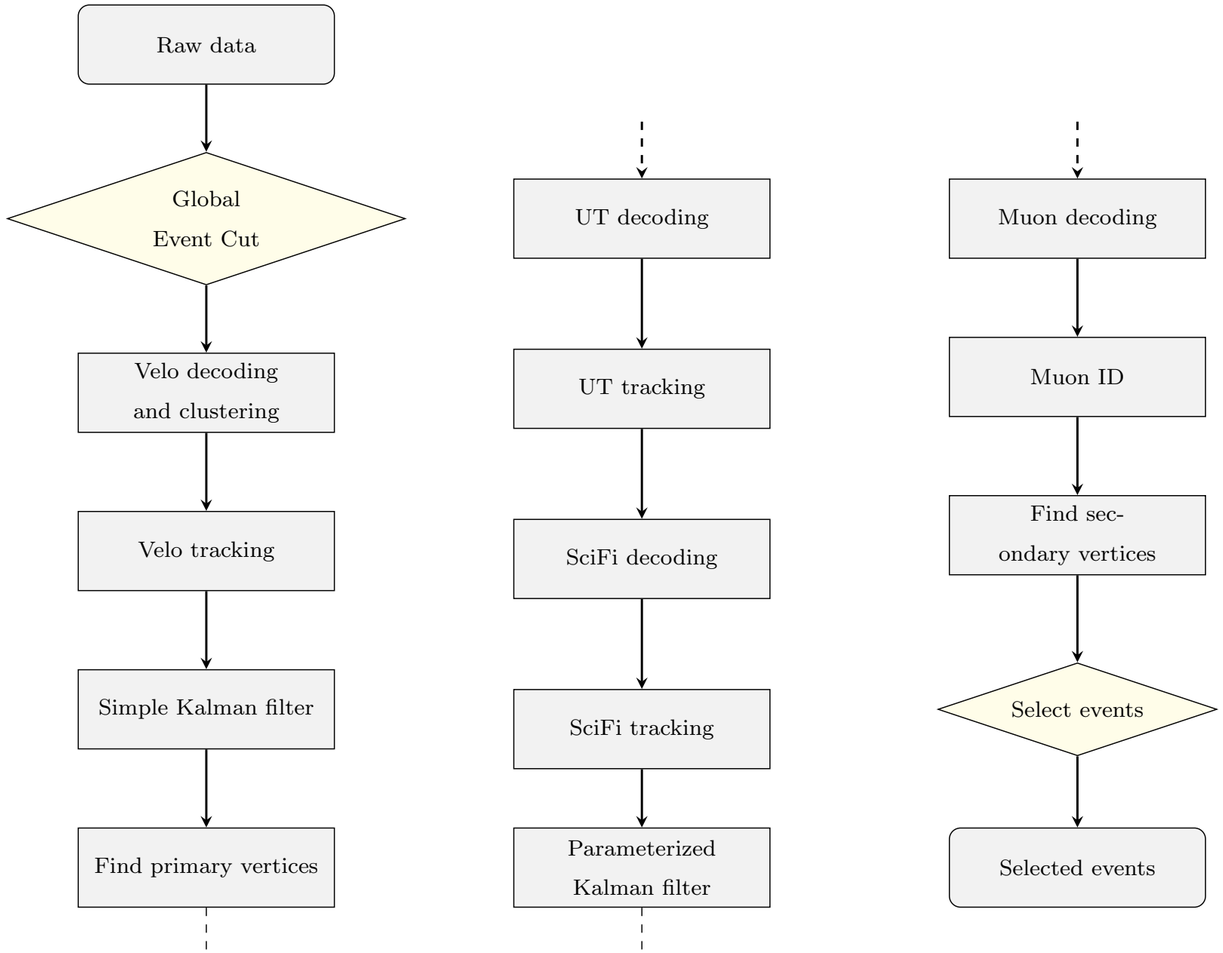
Allen is a complete data processing solution!

A wide range of configurable reconstruction and selection algorithms, monitoring and writing of provenance information.

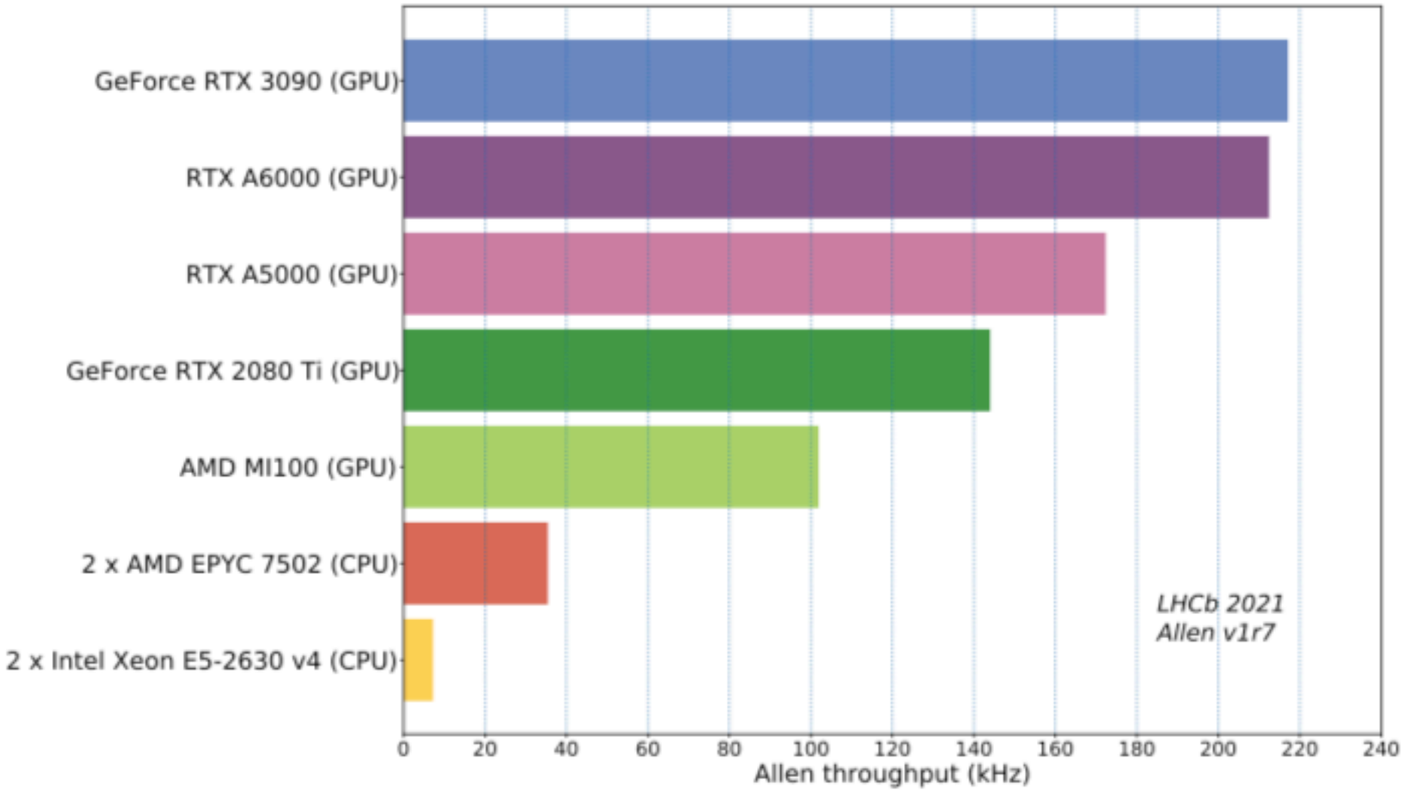
Integration with Gaudi is provided for the LHCb production environment but Allen has no inherent reliance on the LHCb codebase.

I/O is handled with minimal reliance on server CPU.

Cross-architecture by design: write CUDA, compile across a range of GPU and CPU architectures.



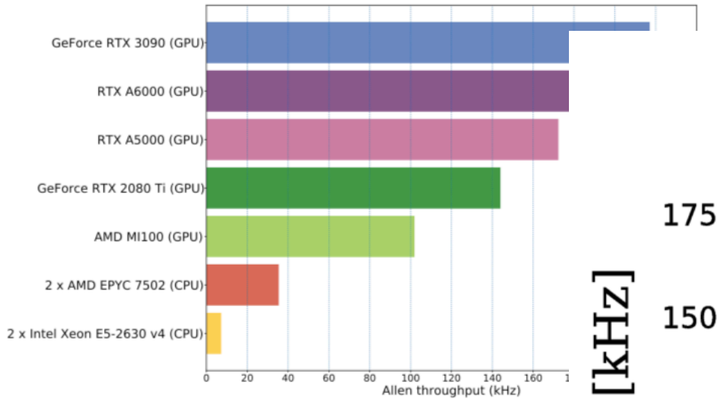
HLT1 throughput performance

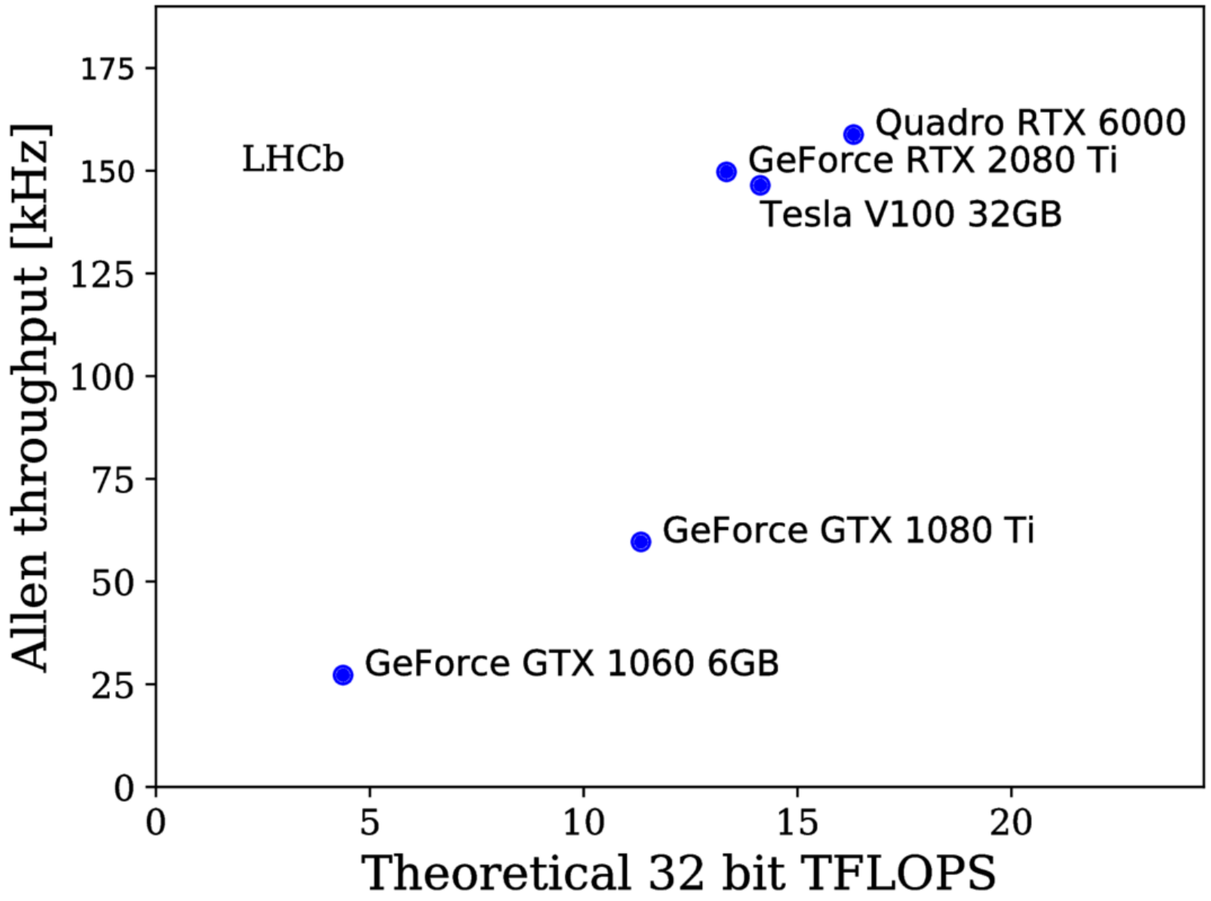


O(200) GPUs required to reach 30 MHz so there is plenty of spare capacity!

References: LHCb-FIGURE-2020-014

HLT1 throughput performance

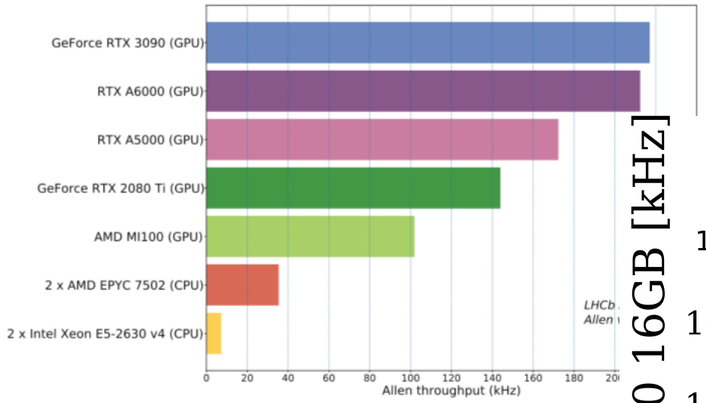


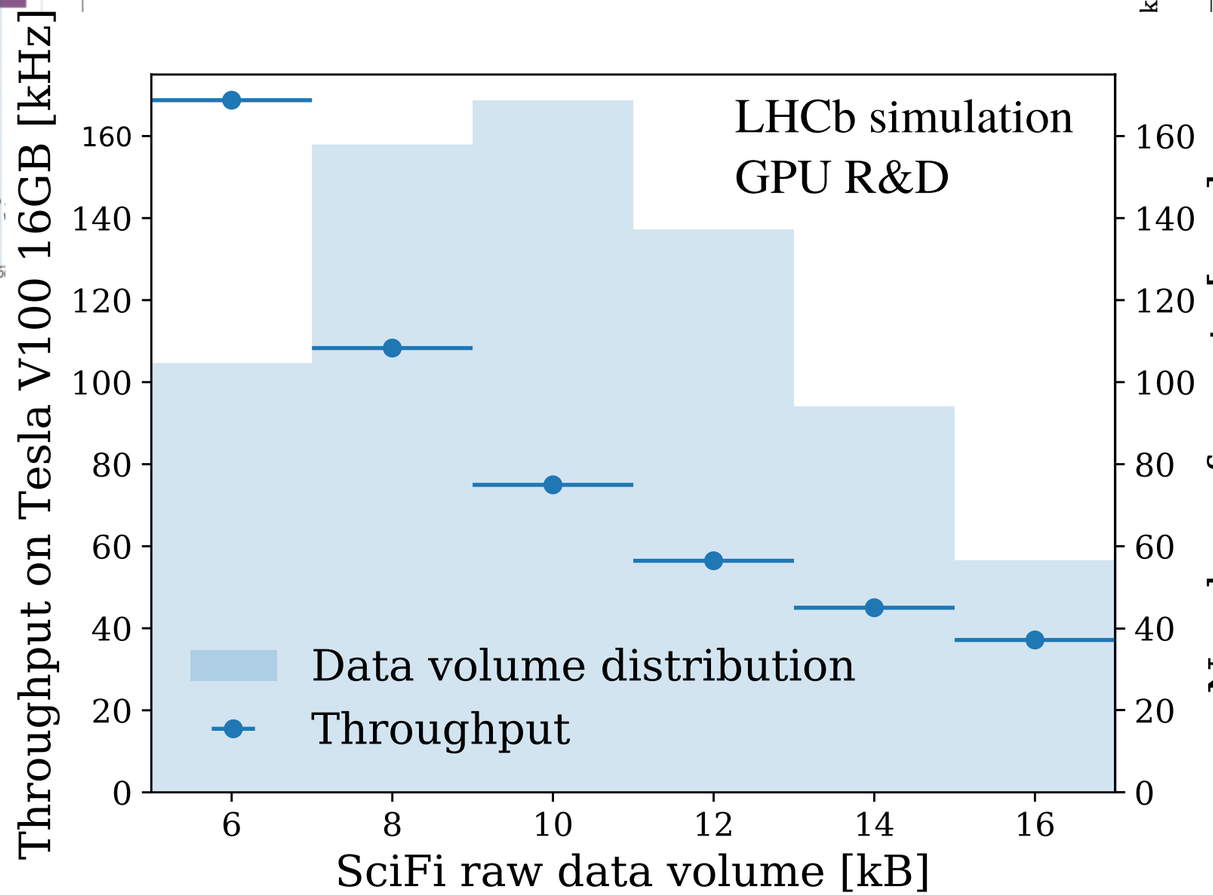


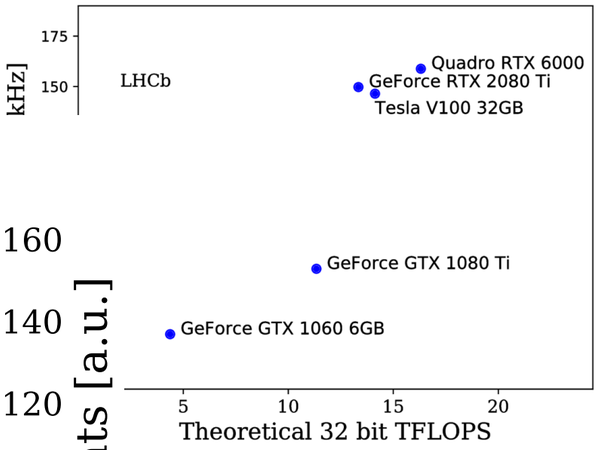
Excellent throughput scaling with theoretical TFLOPS of GPU card

References: LHCb-FIGURE-2020-014

HLT1 throughput performance

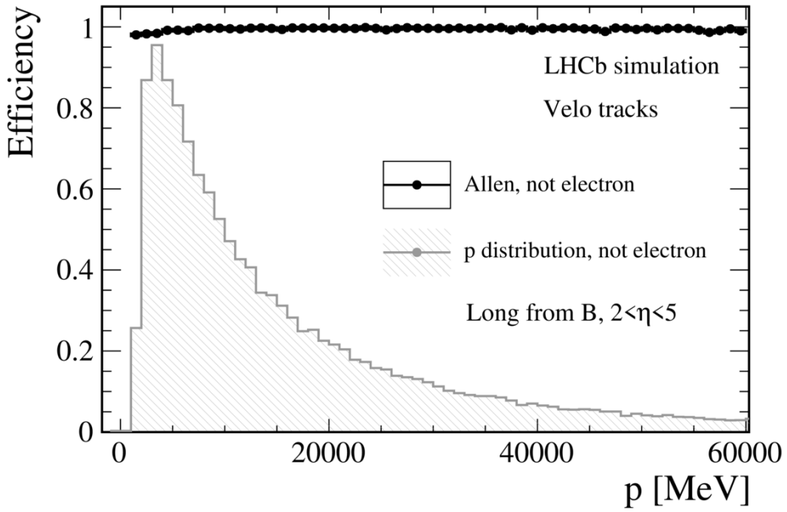


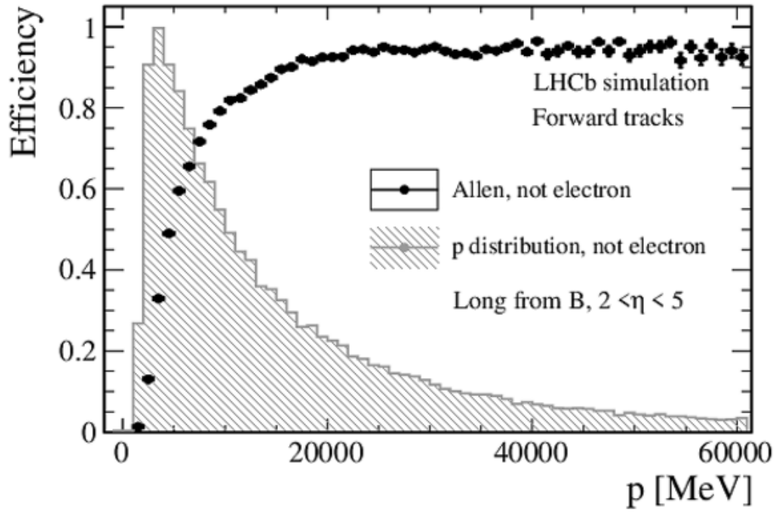


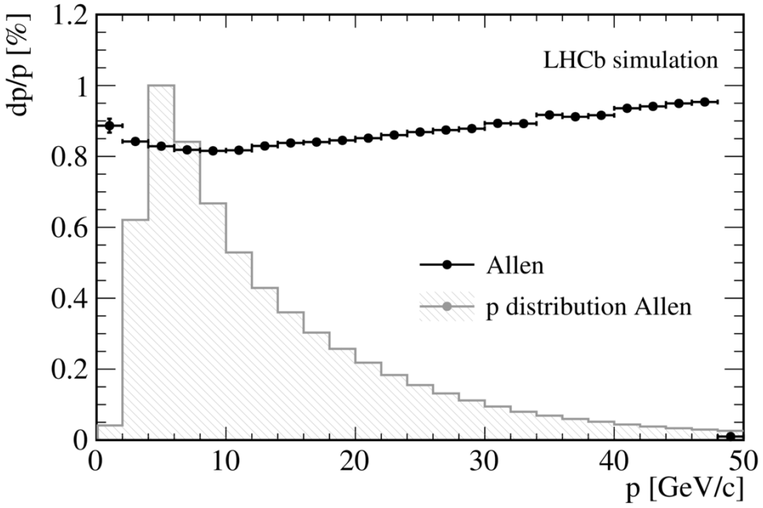


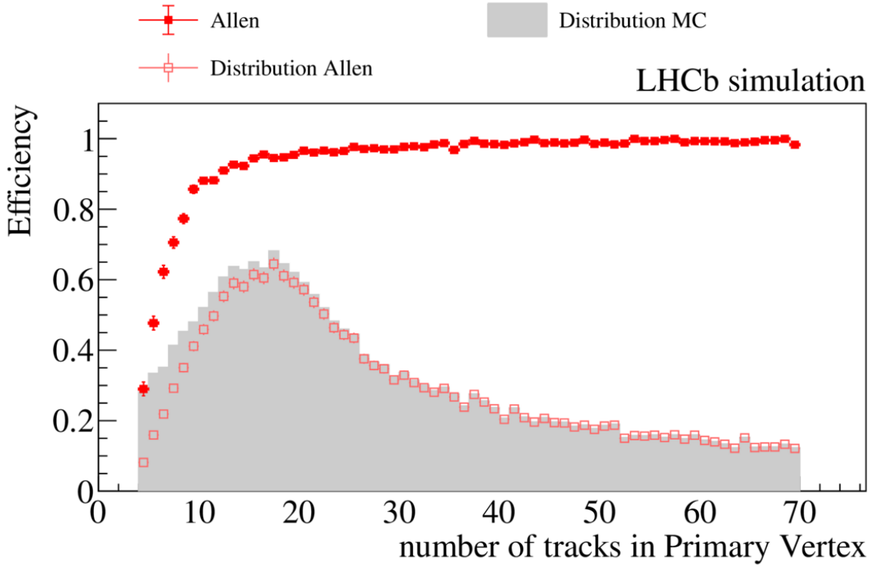
Throughput plateaus with increasing occupancies rather than falling off a cliff

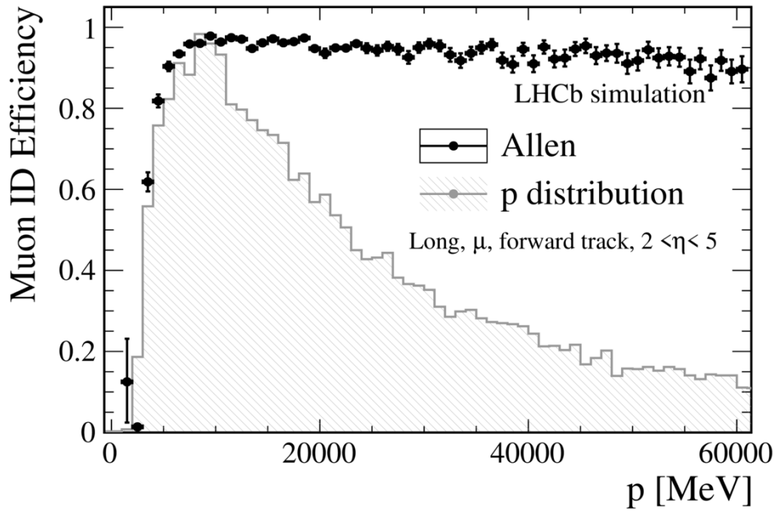
HLT1 reconstruction performance





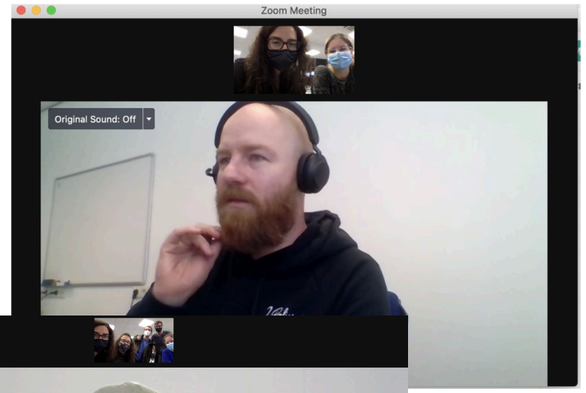


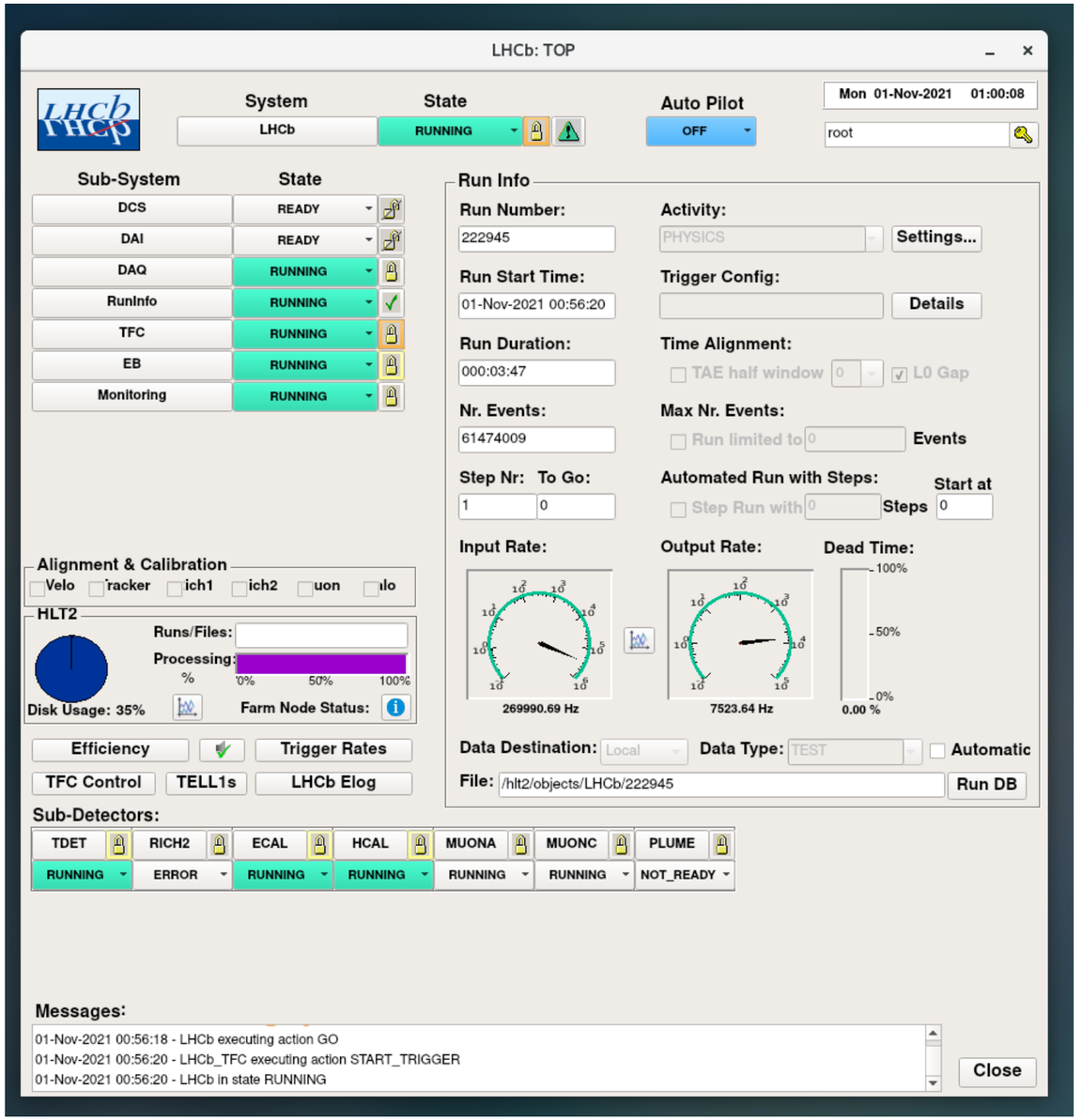




References: LHCb-FIGURE-2020-014

The best integration test possible – real data!





Took some data based on calorimeter activity for the first time during last year's beam test!

SYCL/oneAPI studies

2021 Technical student project: port Allen to run on a prototype Intel DG1

All important parts of the default HLT1 trigger successfully ported and run

Portability verified using physics performance — no major differences found with respect to CUDA, HIP, or CPU builds

No computational performance numbers — but if we can find/fund another student we promise to tell you how fast it is!

Integration into the main development branch is under active discussion.

Personal observations on working in a hybrid world

1. The computing landscape increasingly consists of hybrid architectures. We are developing the skills to thrive in this world!
2. If the basic principles of high throughput computing are respected, a well designed software architecture will perform on x86, GPU, or FPGA systems. Functional design and uniform API helps to achieve this.
3. High-throughput software is far from what universities teach physics students no matter the architecture. Learning CUDA, HLS or C++17 is the same for them. Recognise the importance of new skills in the field.
4. Almost all developer time is spent reliably maximizing performance. Achieving cross-architecture portability is O(10%) of developer time max.
5. Real-time processing is a home for API designers, physicists and selection authors, throughput experts, algorithm designers. It is more work to keep a diverse community coherent, but it's worth it.

Huge thanks to NVIDIA and OpenLab for all the support and encouragement over the last years!

Conclusion and outlook

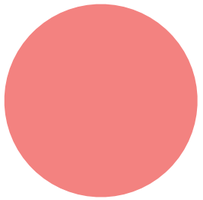
LHCb 2032?

>1000 Eb/year?

Square Kilometre Array (2030s)

Sequence genome of all humans on Earth

Global internet dataflow 2021



2800 Eb/year

ATLAS+CMS 2027

260 Eb/year

LHCb's wonderful real-time adventure continues — perhaps into the 2030s with another upgrade?
An exciting decade of heterogeneous high-performance computing lies ahead of us!

Backup

LHCb analysis methodology and role of calibration samples

Trigger Efficiency

Tag-and-probe calibration method exists & widely used

Tracking efficiency

Tag-and-probe

Existing	Developing
μ	e,π,K,p

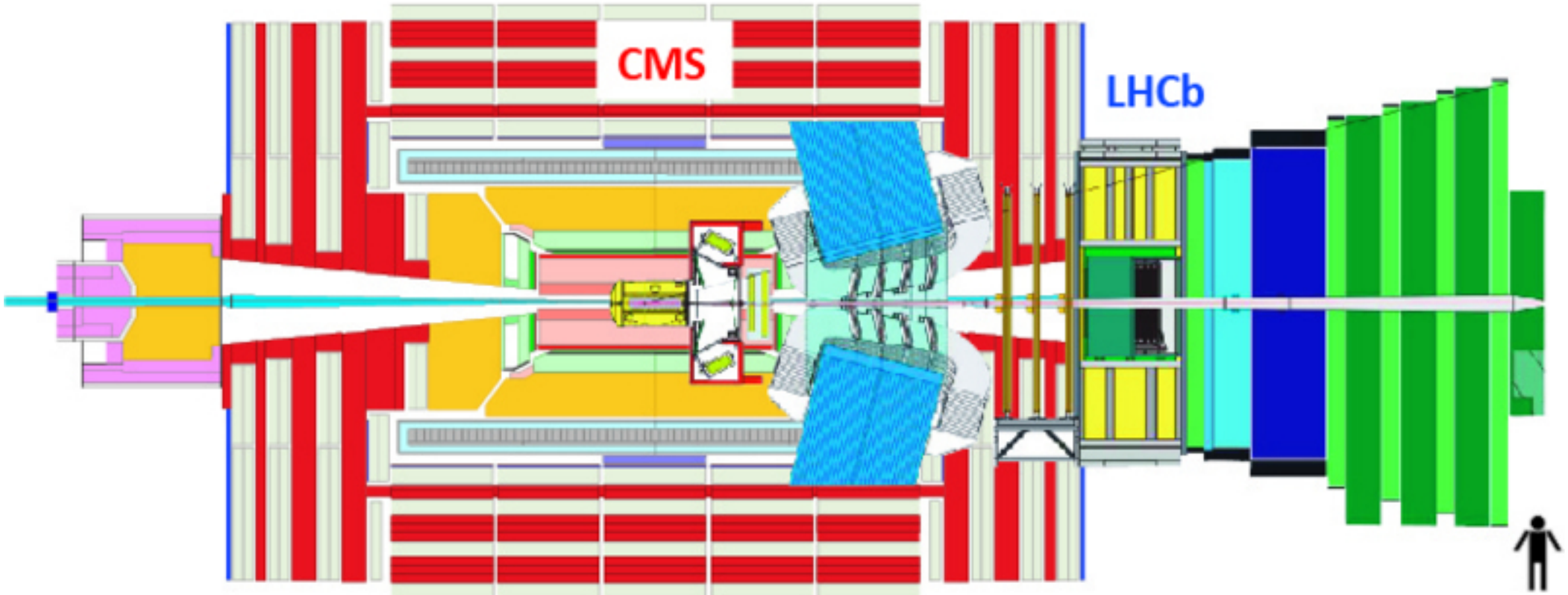
Particle identification

Tag-and-probe

Tag-and-probe calibrations exist for all charged particle species and for π^0/γ, with new sources added over time to improve coverage

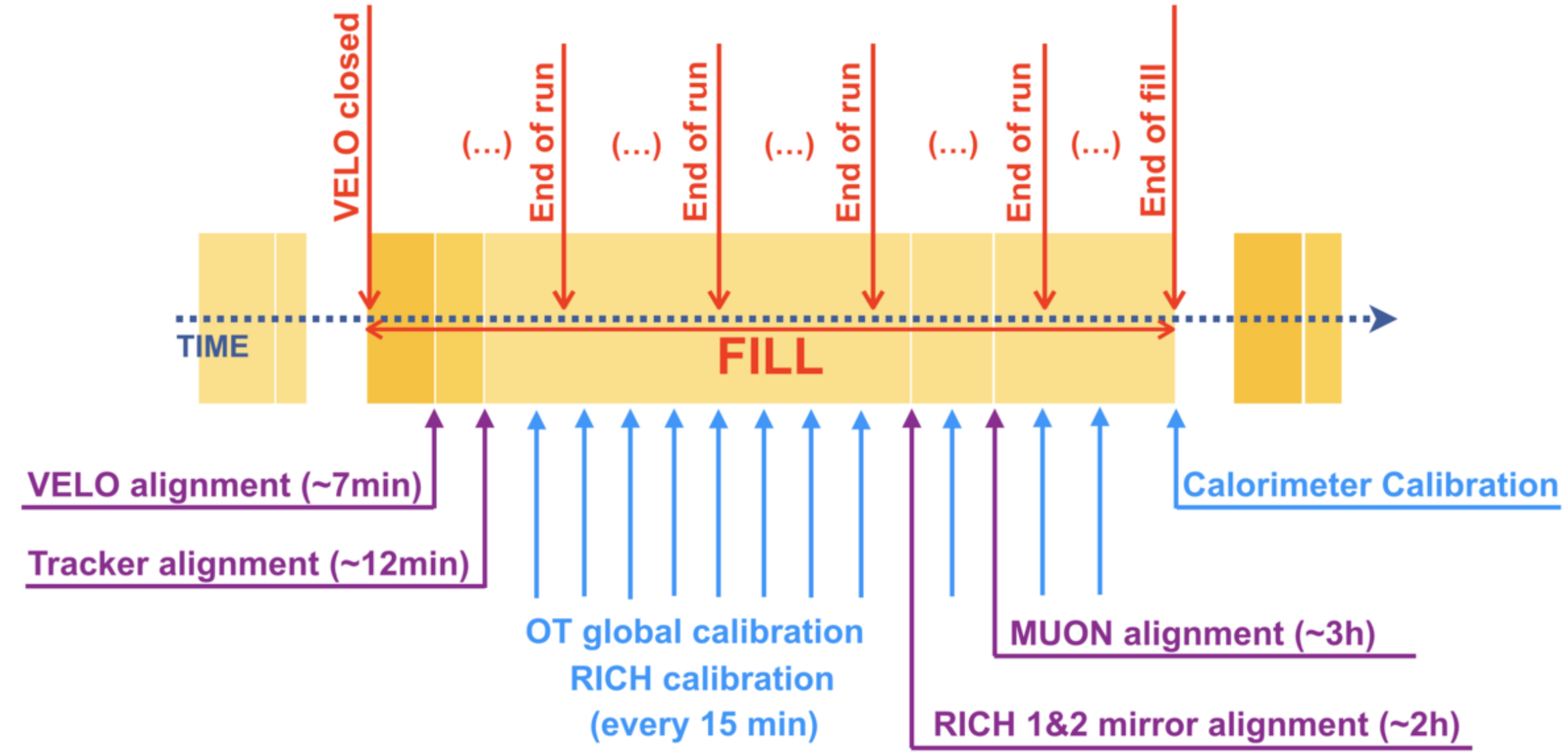
Data driven efficiency calibration key to precision physics

The LHCb detector at the LHC



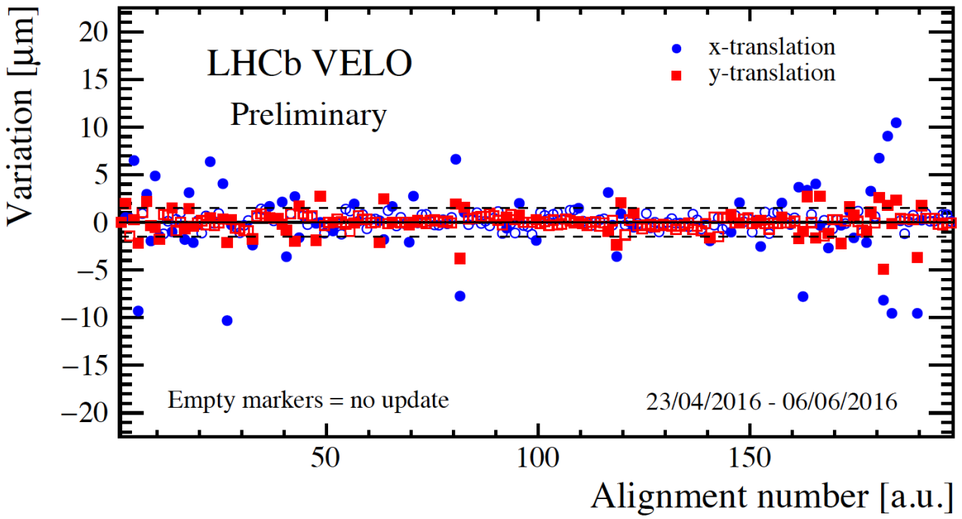
Forward spectrometer optimized for precision physics

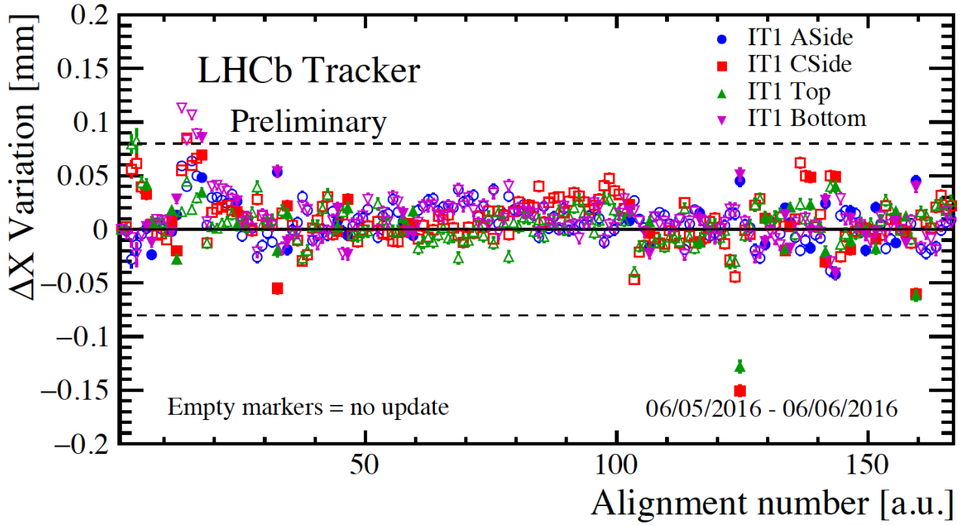
We also need to align and calibrate our detector in real time

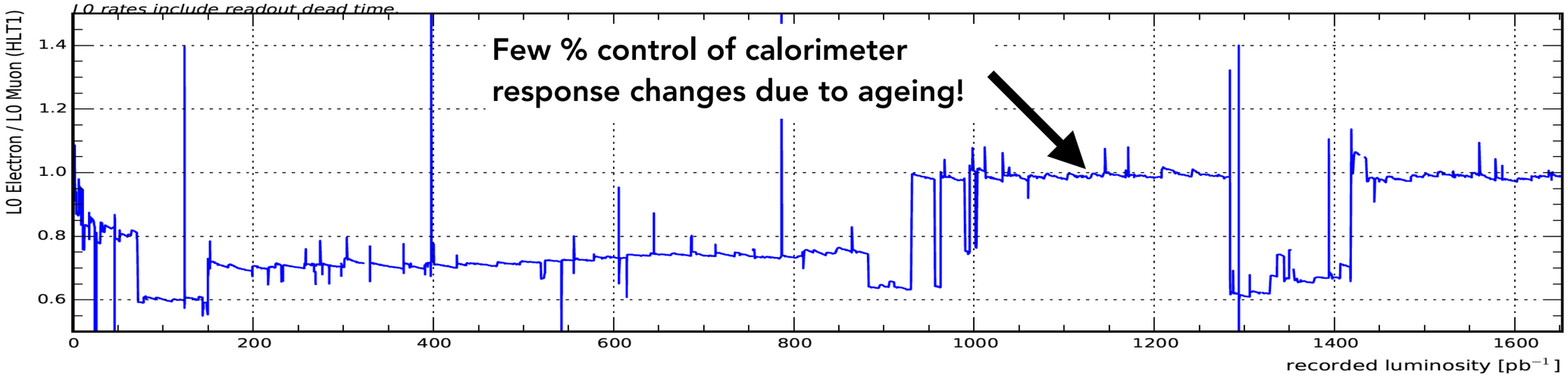


((~7min),(~12min),(~3h),(~2h)) - time needed for both data accumulation and running the task

So we did!



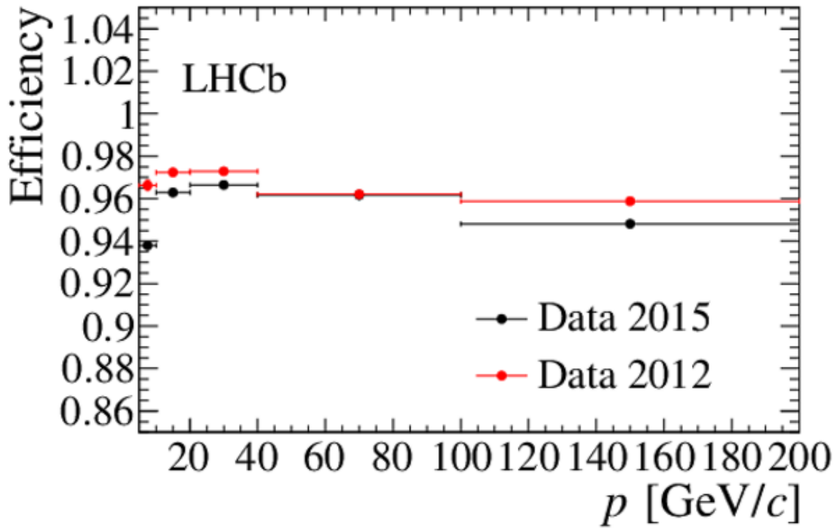


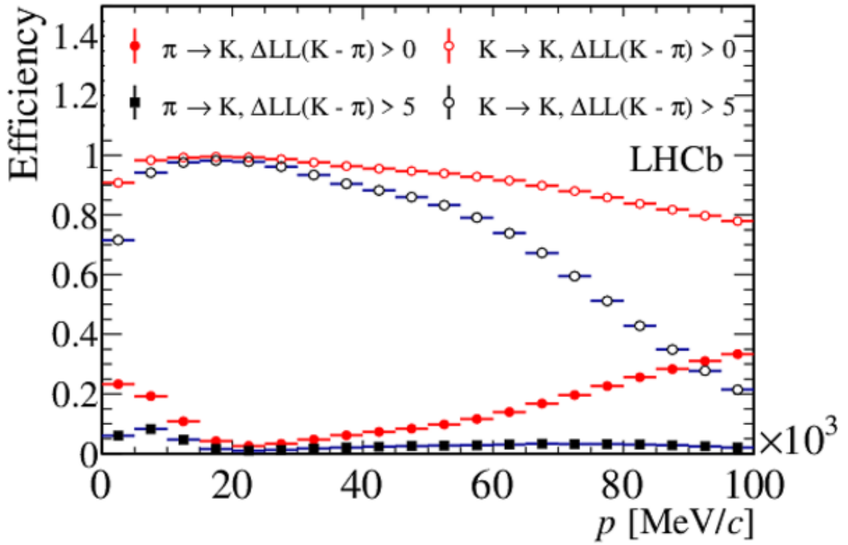


Implemented for the first time in Run 2 with offline like quality from very early in 2015. Not only tracker but also RICH and calorimeter. For me this is the most impressive aspect of LHCb's Run 2 and required a huge team effort across projects and working groups.

We also need to measure our efficiencies in real-time!

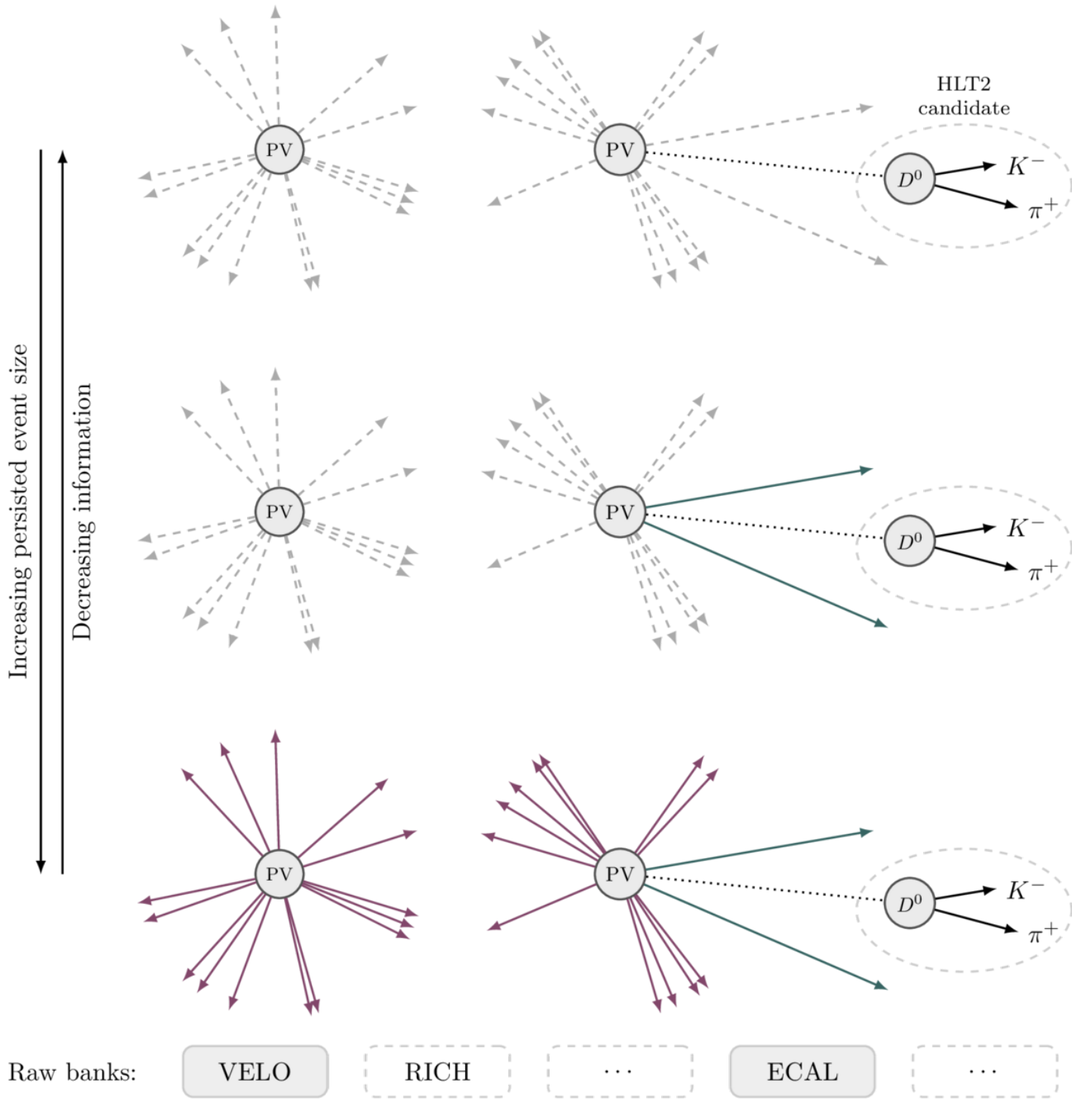
Species	Low momentum	High momentum
$e^{\pm}$	$B^+ \to J/\psi K^+$ with $J/\psi \to e^+e^-$	
$\mu^{\pm}$	$B^+ \to J/\psi K^+$ with $J/\psi \to \mu^+\mu^-$	$J/\psi \to \mu^+\mu^-$
$\pi^{\pm}$	$K^0_S \to \pi^+\pi^-$	$D^{*+} \to D^0\pi^+$ with $D^0 \to K^-\pi^+$
$K^{\pm}$	$D^+_s \to \phi\pi^+$ with $\phi \to K^+K^-$	$D^{*+} \to D^0\pi^+$ with $D^0 \to K^-\pi^+$
p , $\overline{p}$	$\Lambda^0 \to p\pi^-$	$\Lambda^0 \to p\pi^-$; $\Lambda^+_c \to pK^-\pi^+$





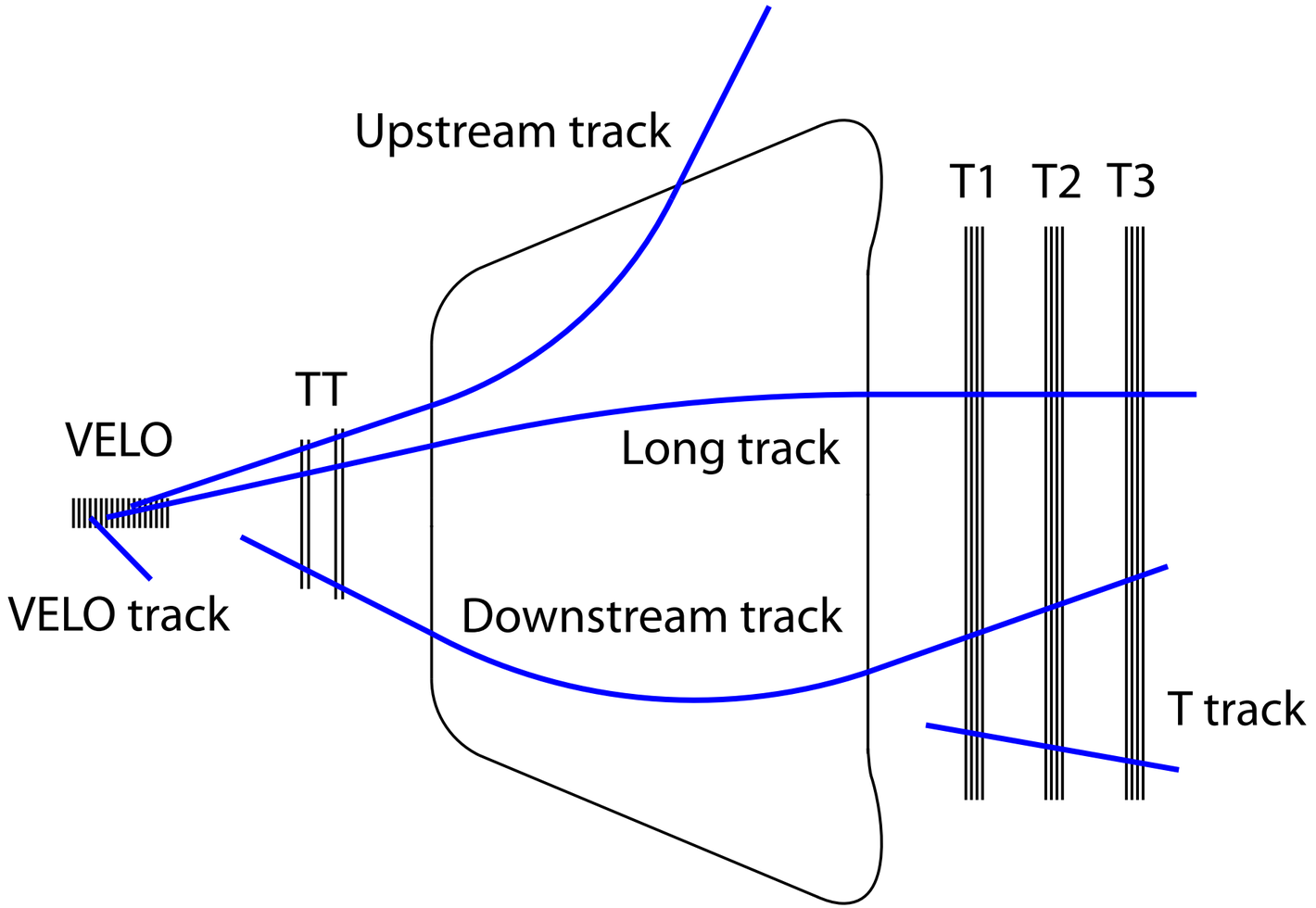
Unlike ATLAS and CMS, LHCb must maintain a data-driven permille level control of its efficiency across the kinematic and geometric acceptance of the detector. Requires collecting an extremely wide range of tag-and-probe samples in real time.

Then select signals and associate them to pp collisions

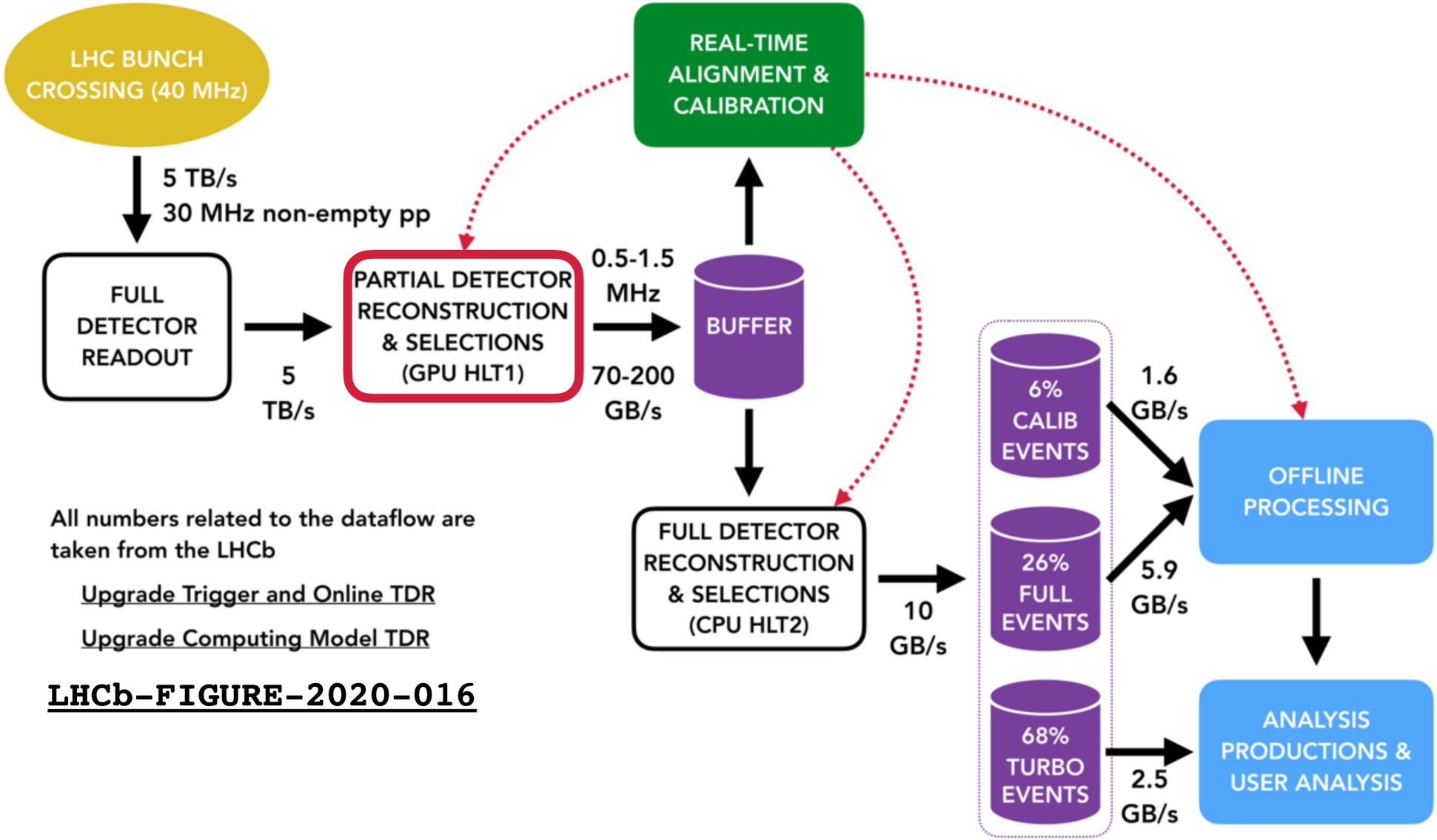


Full flexibility to store "additional" detector information if required by some analyses

Tracks in LHCb



LHCb upgrade dataflow



All numbers related to the dataflow are taken from the LHCb

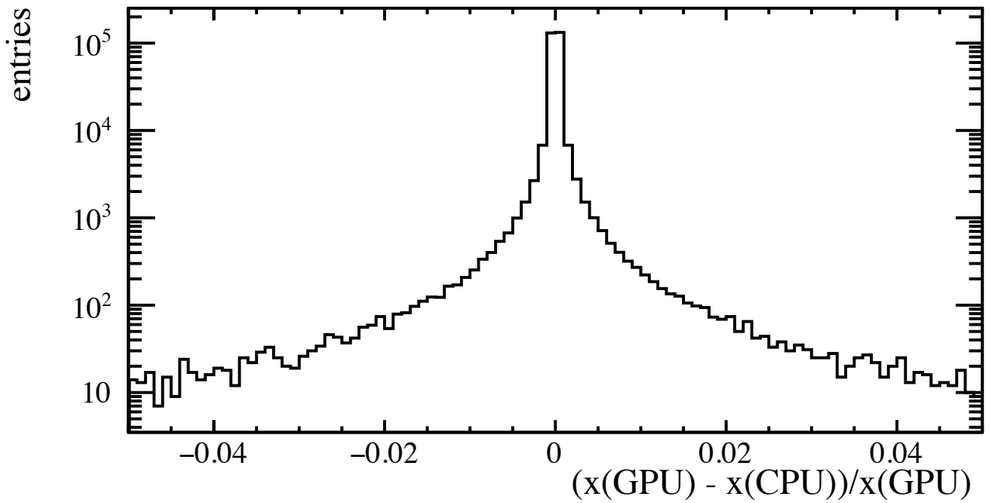
Upgrade Trigger and Online TDR

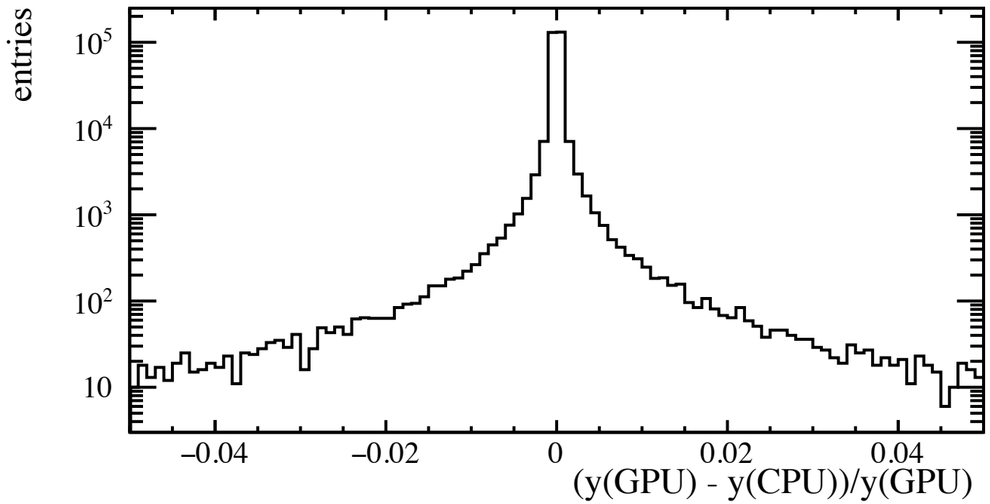
Upgrade Computing Model TDR

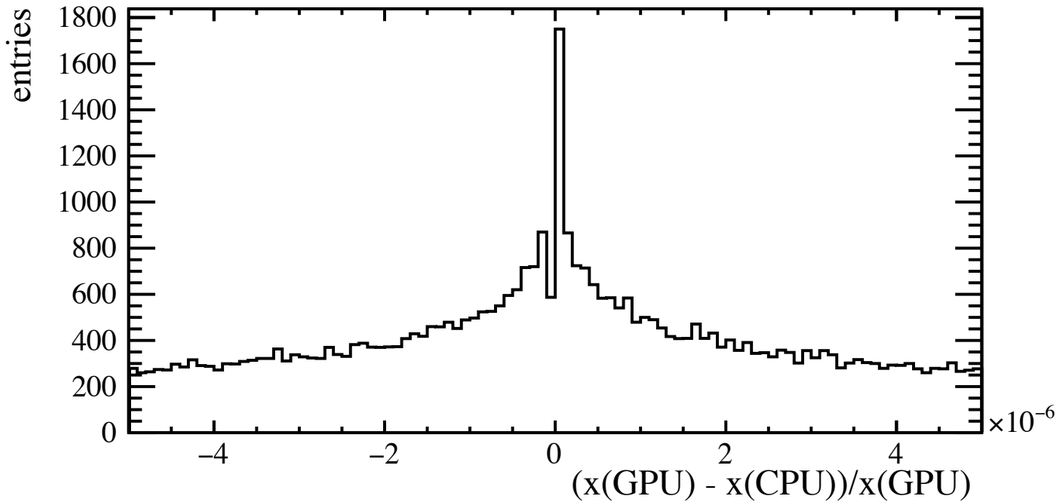
LHCb-FIGURE-2020-016

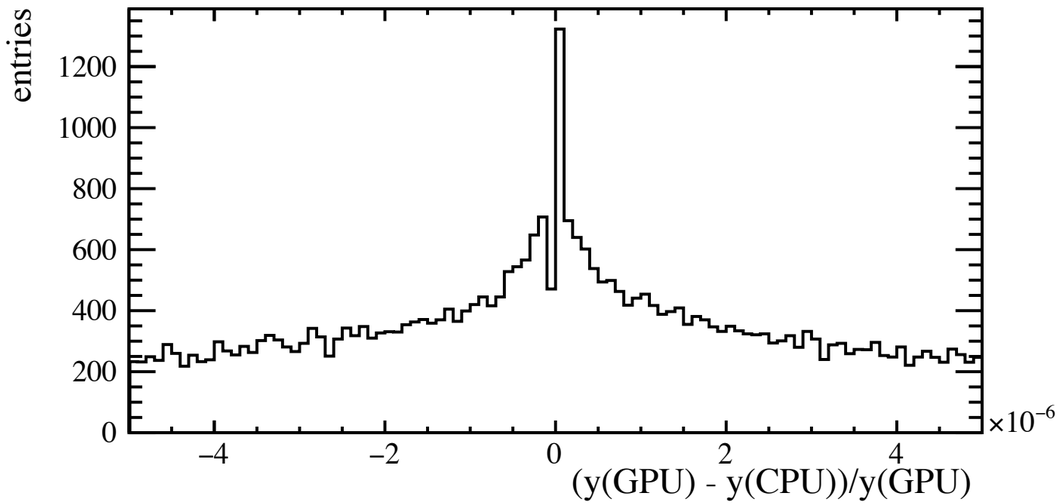
HLT1 is a huge high-throughput challenge — budget of a few M$

Paying attention to 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

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

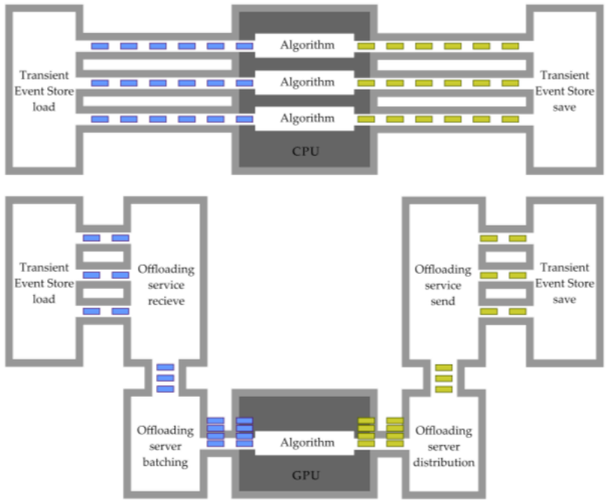
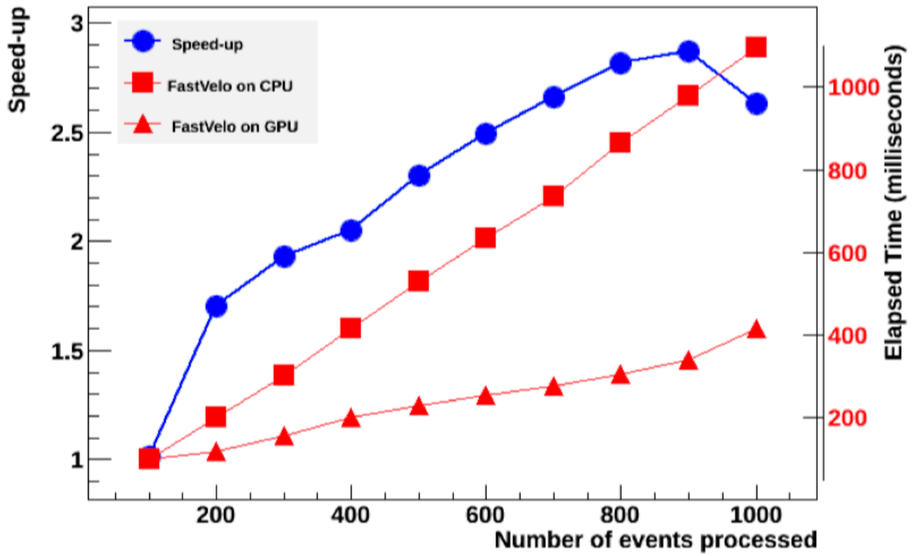


Figure 2 Execution of multiple Gaudi pipelines concurrently using a classical CPU algorithm vs. offloading to a GPU.

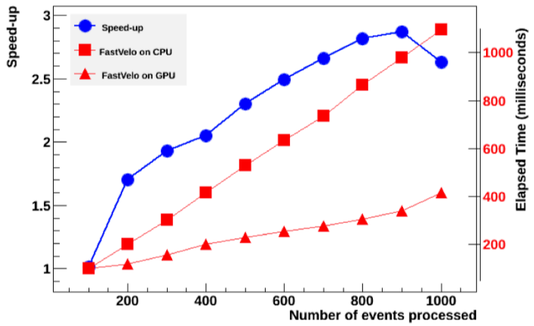


2014 2015 2016 2017 2018 2019 2020

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).

Bibliography: Badalov et al, Badalov et al

By 2017 we had largely concluded this would never work



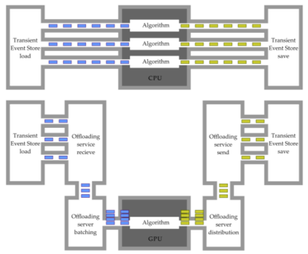


Figure 2 Execution of multiple Gaudi pipelines concurrently using a classical CPU algorithm vs. offloading to a GPU.

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

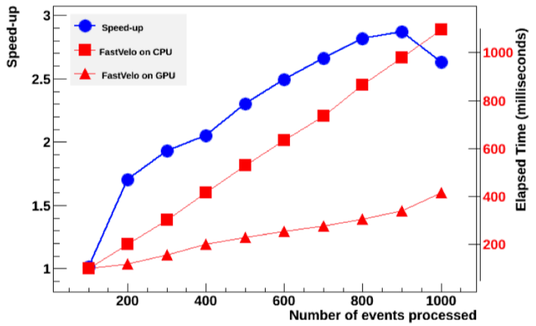
	PrPixel	Track forwarding	
Time per event (ms)	3.6	26.2	batch of 1
		3.5	batch of 40
		2.0	batch of 100
		0.80	batch of 300
Ghost rate	1.7%	0.8%	
Efficiency for long tracks	98.3%	98.0%	
Efficiency for long tracks over 5 GeV	98.8%	98.4%	

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...



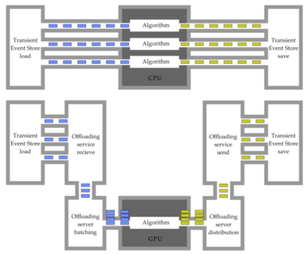
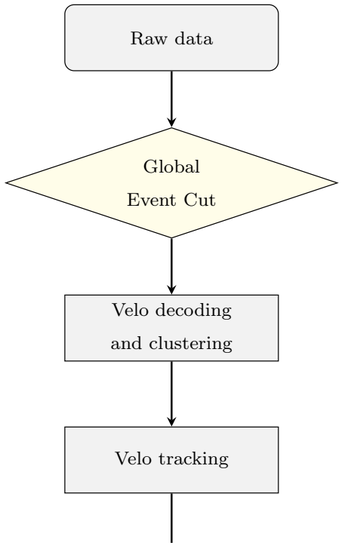


Figure 2 Execution of multiple Gaudi pipelines concurrently using a classical CPU algorithm vs. offloading to a GPU.

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

	PrPixel	Track forwarding	
		batch of 1	26.2
		batch of 40	3.5
		batch of 100	2.0
		batch of 300	0.80
Time per event (ms)	3.6		
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%

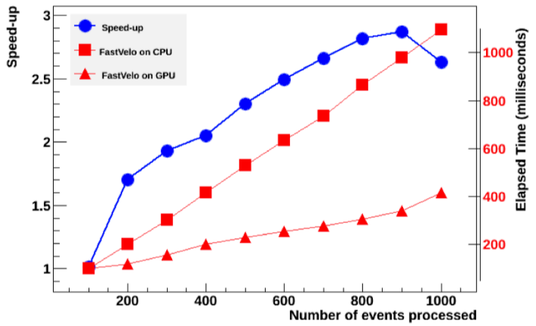


2014 2015 2016 2017 **2018** 2019 2020

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.

Bibliography: Badalov et al, Badalov et al, Badalov et al, Campora et al

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



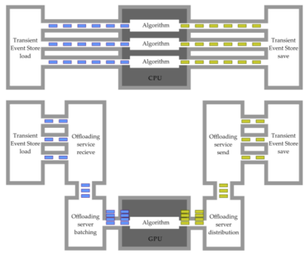
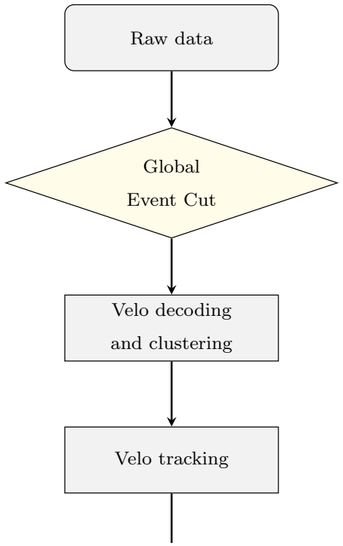
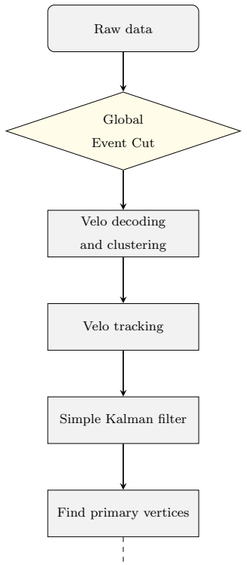


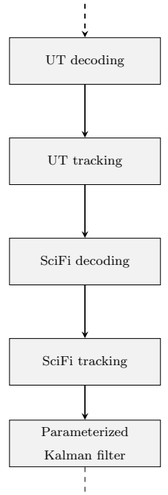
Figure 2 Execution of multiple Gaudi pipelines concurrently using a classical CPU algorithm vs. offloading to a GPU.

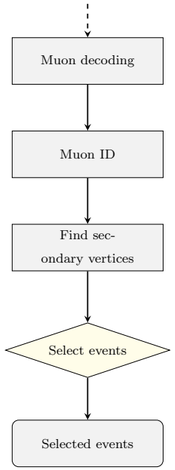
Table 1. Comparison of a CPU and a GPU VELO Pixel tracking algorithm.

	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%	







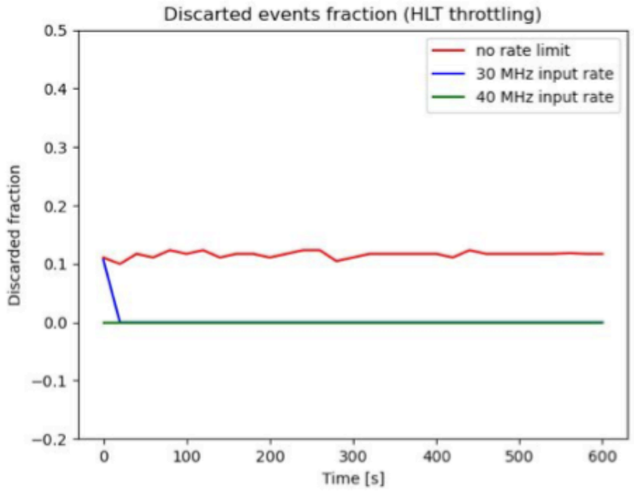


2014 2015 2016 2017 2018 2019 2020

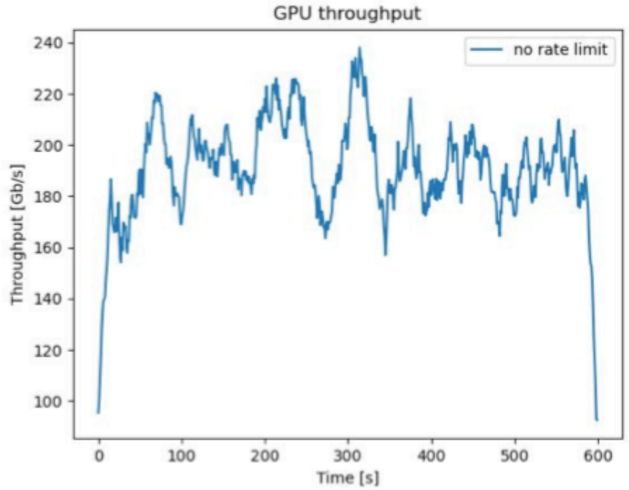
Putting “everything” on the GPU unlocked the power of the architecture and made it cost-effective. Classic accumulation of knowledge on a plateau followed by a phase transition as it came together. Similarly, the vectorization of our CPU reconstruction also came together in parallel to meet the required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

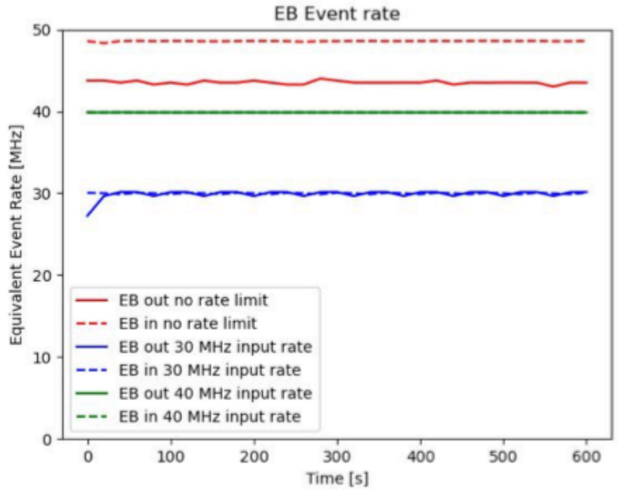
Bibliography: Badalov et al, Badalov et al, Badalov et al, Campora et al, GPU HLT TDR

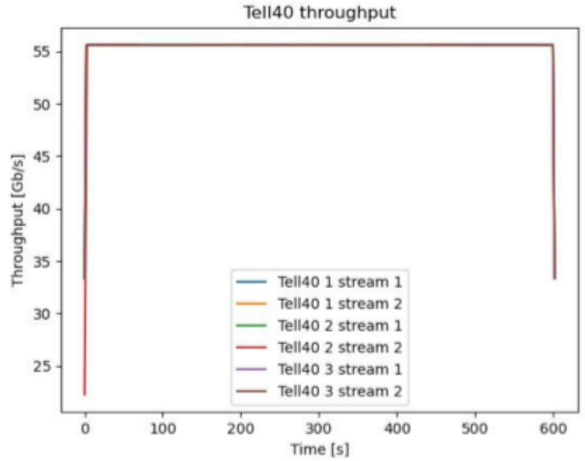
Integration tests

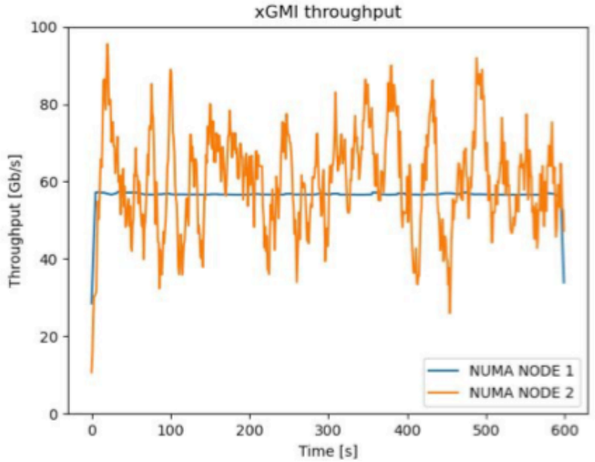


- No discards at 40 MHz
- Bottleneck at 43 MHz: data to GPU
- Test successful





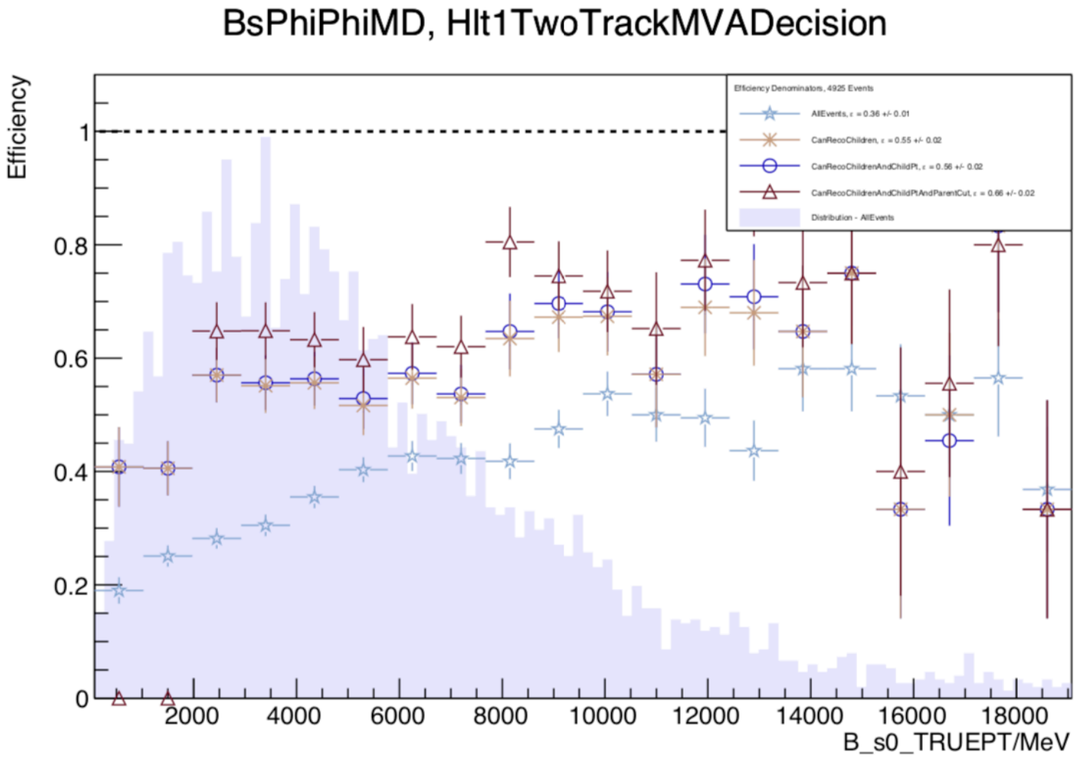


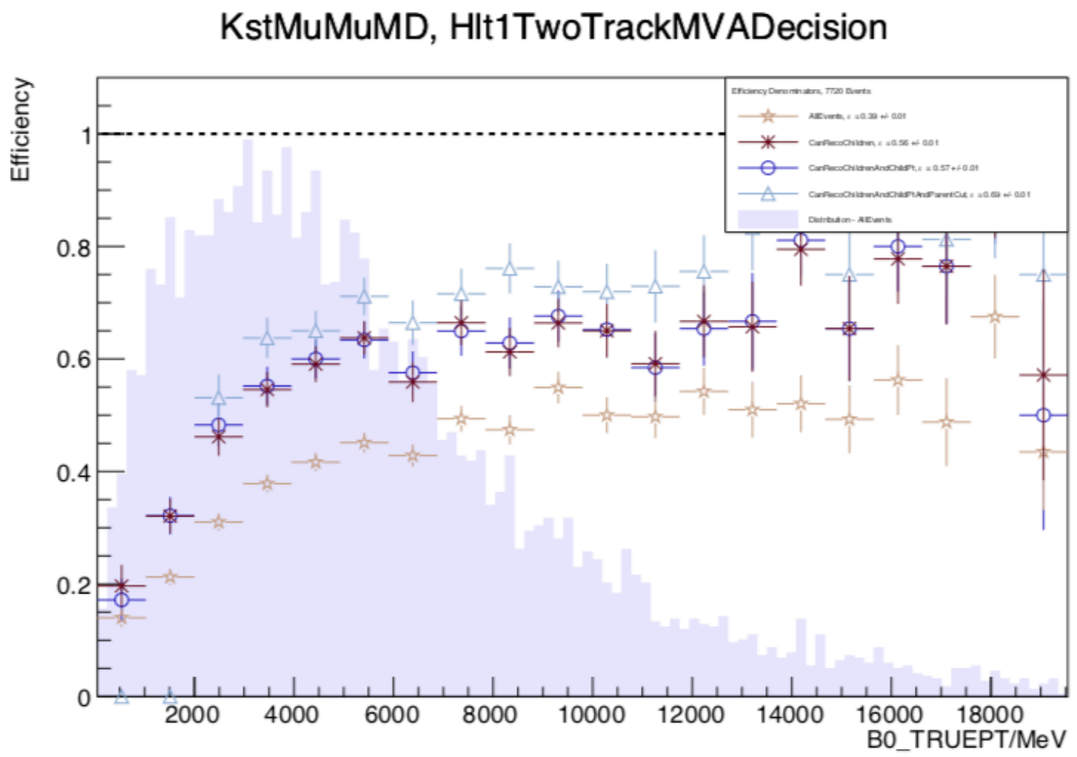


Many tests performed including running GPUs for days without stop for stability and realistic tests of memory pressure and throughput in the servers we will use. Everything looks good!

Further reading: LHCb Upgrade GPU High Level Trigger TDR

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!

References: LHCb Upgrade GPU High Level Trigger TDR