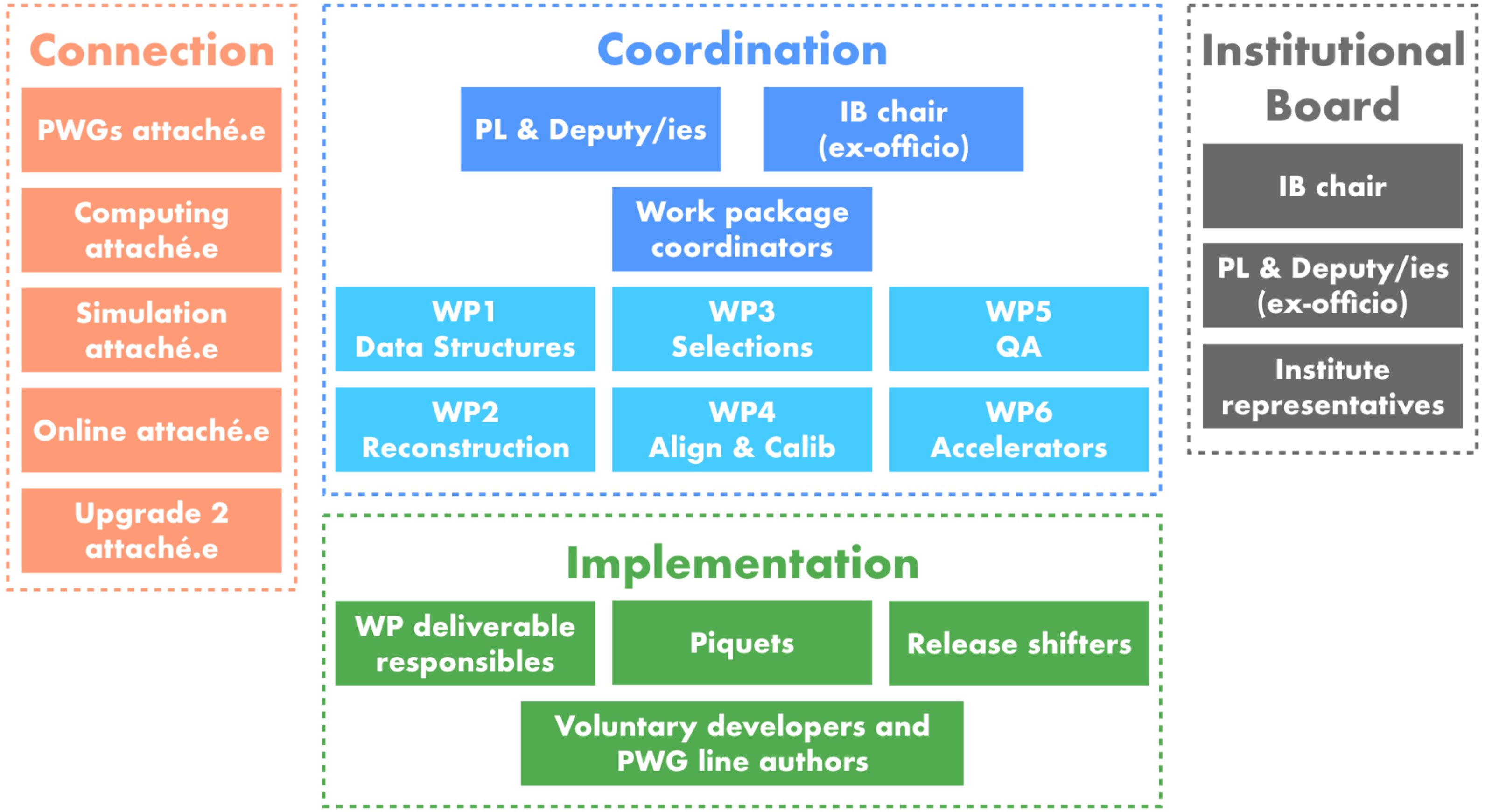
Variable latency tracking @ 30 MHz



European Research Council
Established by the European Commission

V. V. Gligorov, CNRS/LPNHE

Institut Pascal, 16.10.2019

LPNHE PARIS

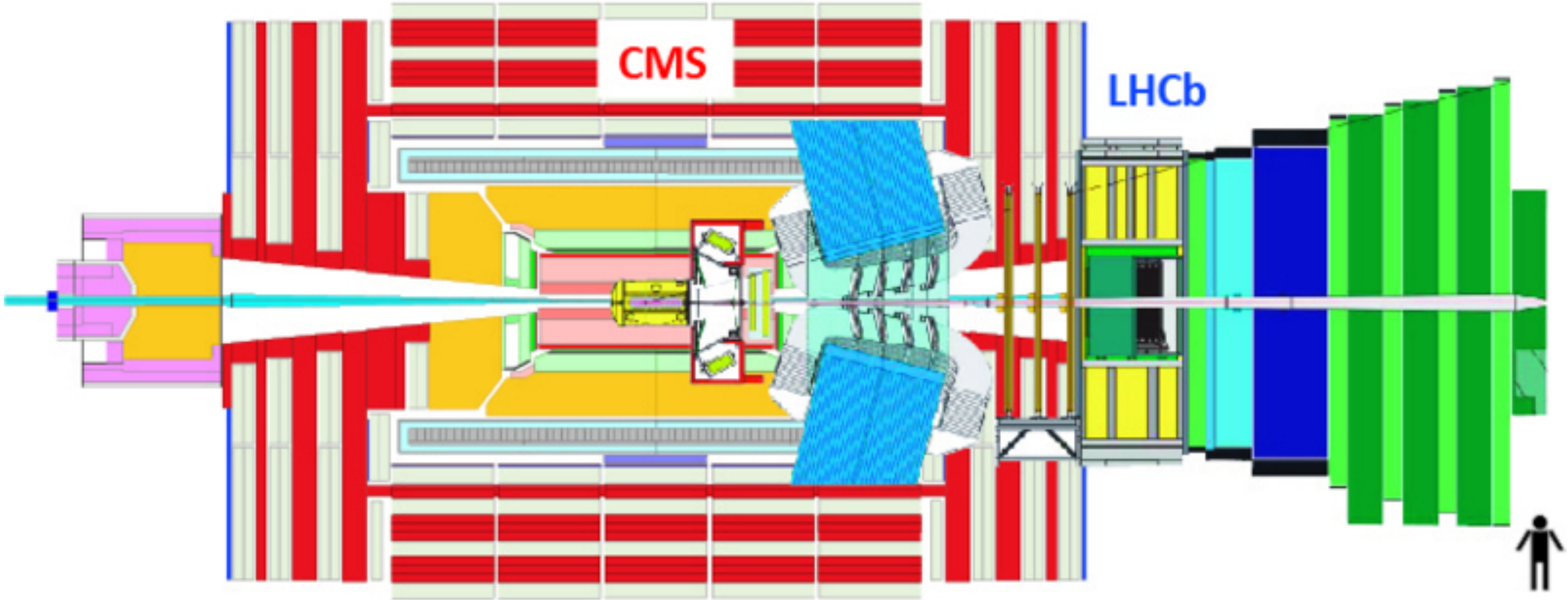
Objectives of this talk

1. Motivate why it is interesting to perform variable latency charged particle reconstruction (tracking) in real-time, which at the LHC means at 30 MHz

2. Describe the challenges involved with delivering such a tracking for the LHCb experiment with reference to two specific architectures: x86 and GPU

3. Give my personal thoughts on what we have learned during this development process in LHCb, and thoughts about where this is going in the future

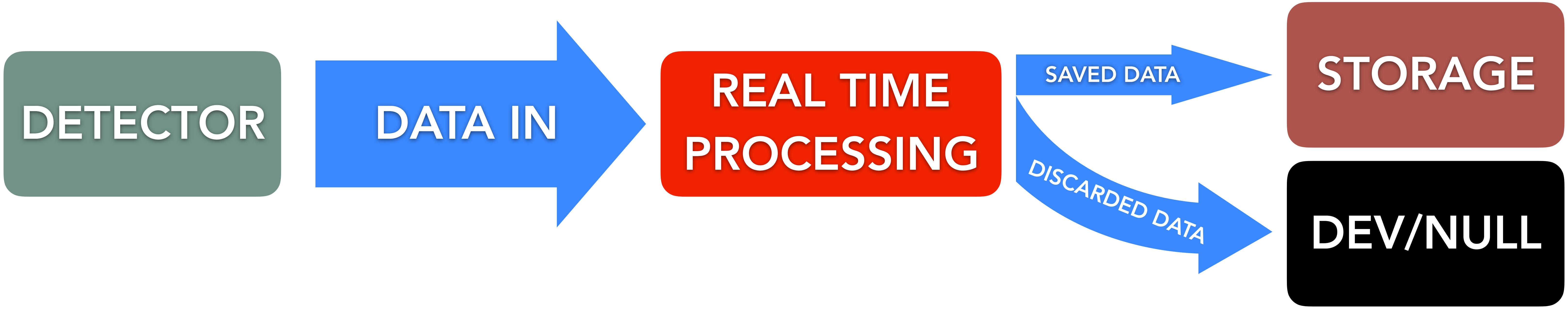
The LHCb detector at the LHC



Forward spectrometer optimized for precision physics

Why tracking @30 MHz?
Why variable latency?

Q : What is real-time?



A : Any processing of data before it is permanently recorded

Why do we need to process data before recording it?

	LHCb	CMS/ATLAS
Data volume at detector in Run 2	~30 Eb/ year	~1 Zb/ year

Why do we need to process data before recording it?

	LHCb		CMS/ATLAS
Data volume at detector in Run 2	~30 Eb/ year		~1 Zb/ year

Global internet dataflow 2015

~640 Eb/ year

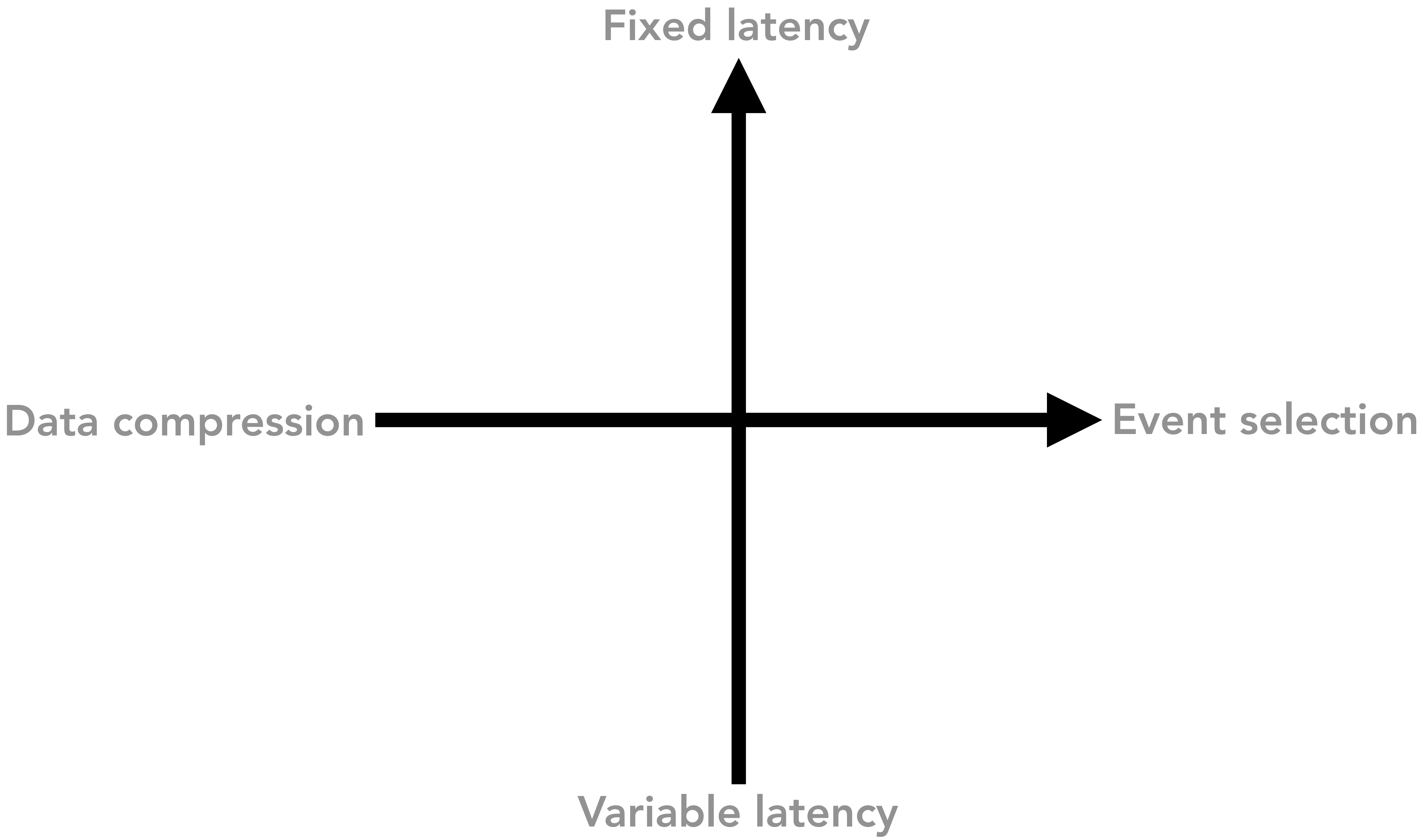
Because HEP detectors produce too much data to store

Data volumes @ LHC after real-time processing

	LHCb		CMS/ATLAS
Data volume at detector in Run 2	~30 Eb/ year	Global internet dataflow 2015: ~640 Eb/ year	~1 Zb/ year
Data volume for analysts	~30 Pb/ year		~40 Pb/ year

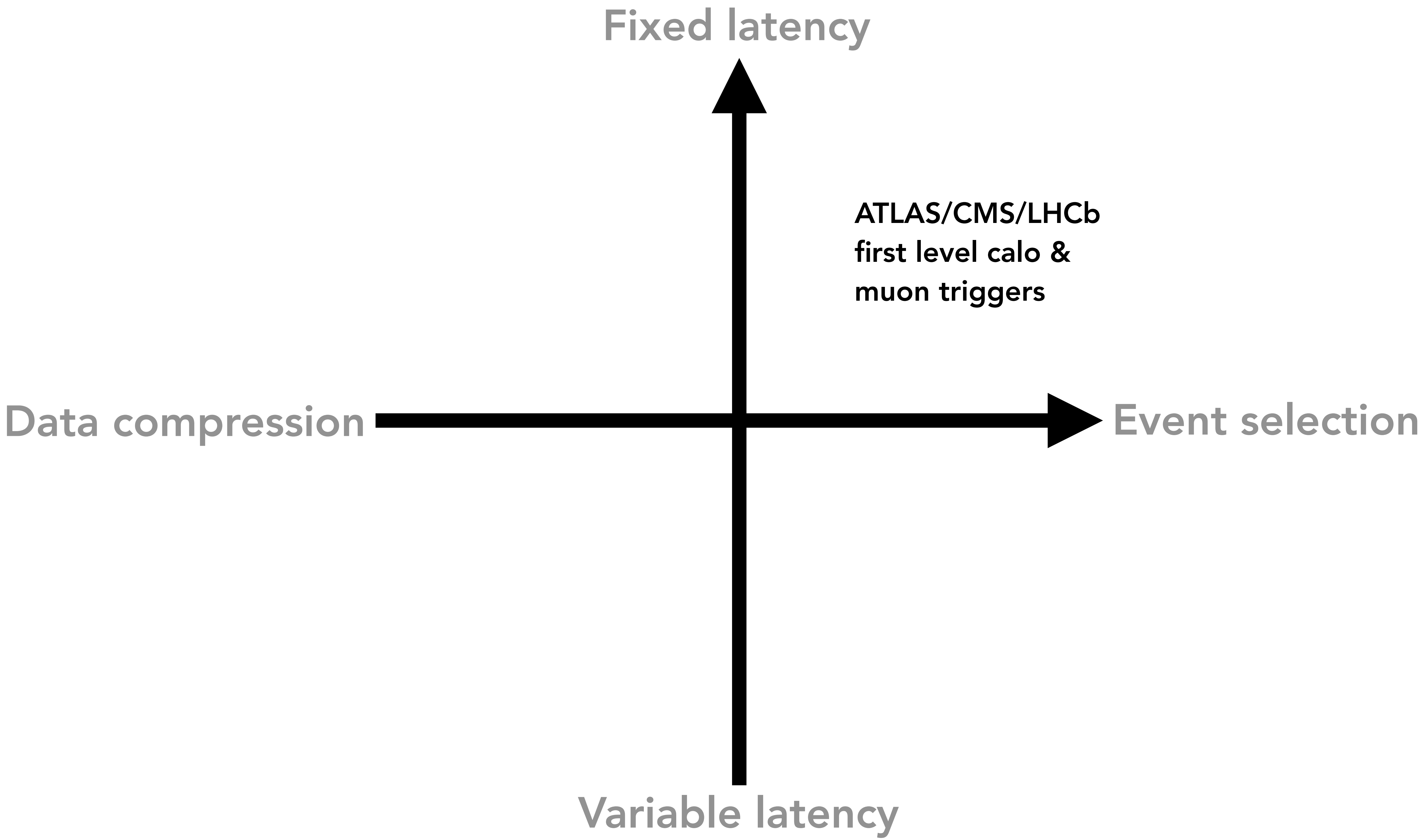
Real-time processing reduces data by 3-5 orders of magnitude

What kinds of real-time data processings exist?



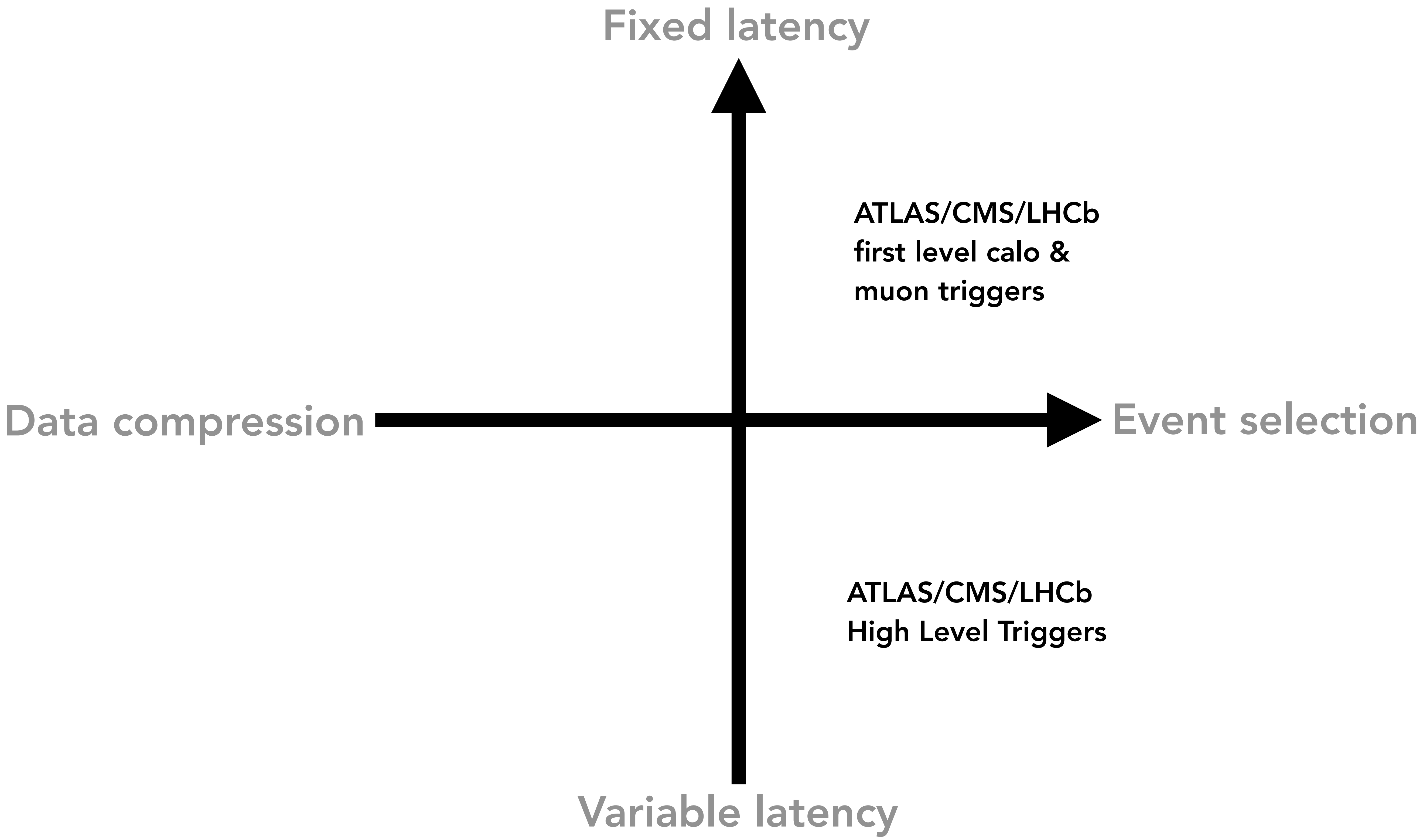
Distinguish fixed & variable latency, selection & compression

What kinds of real-time data processings exist?



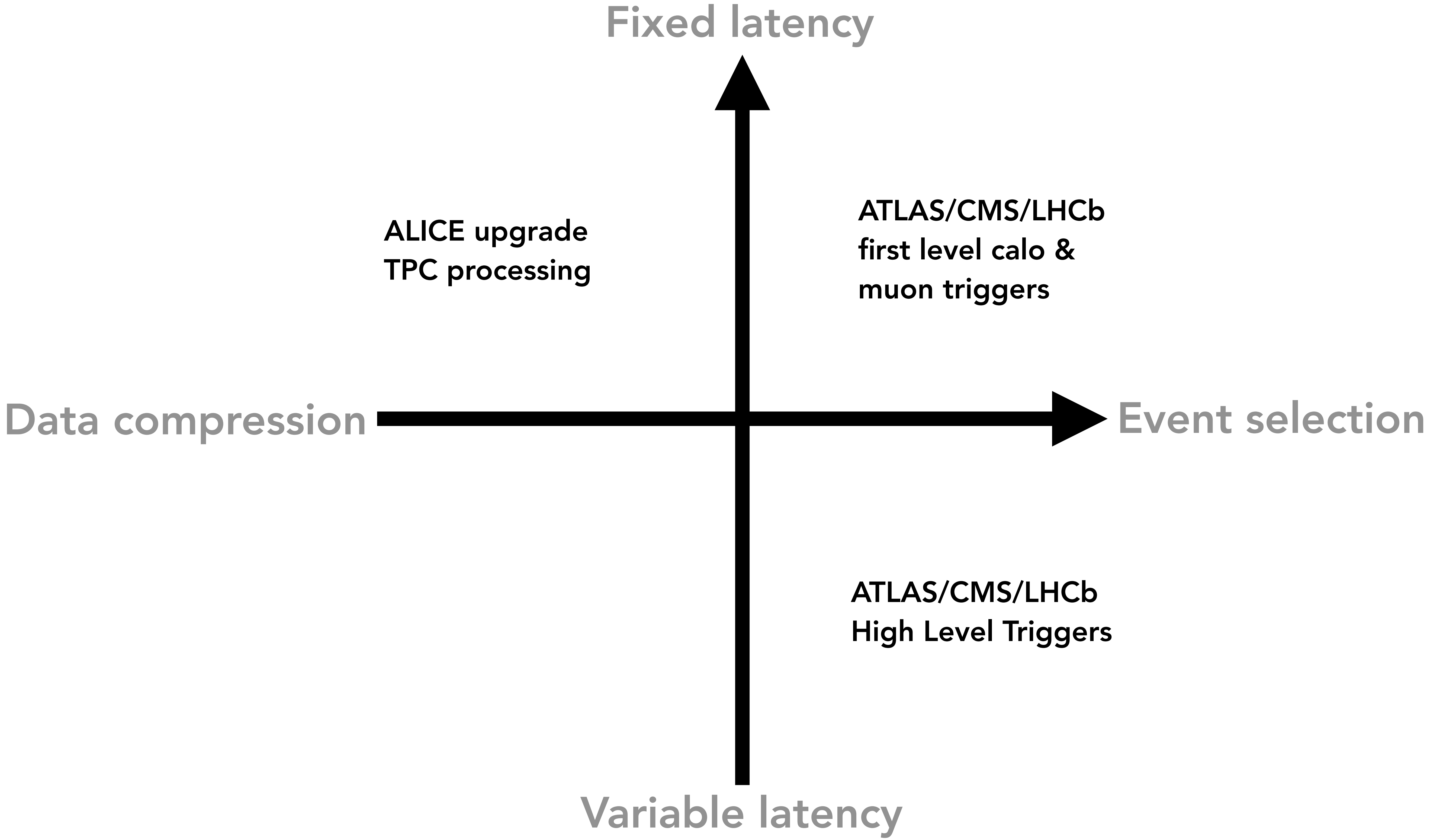
Distinguish fixed & variable latency, selection & compression

What kinds of real-time data processings exist?



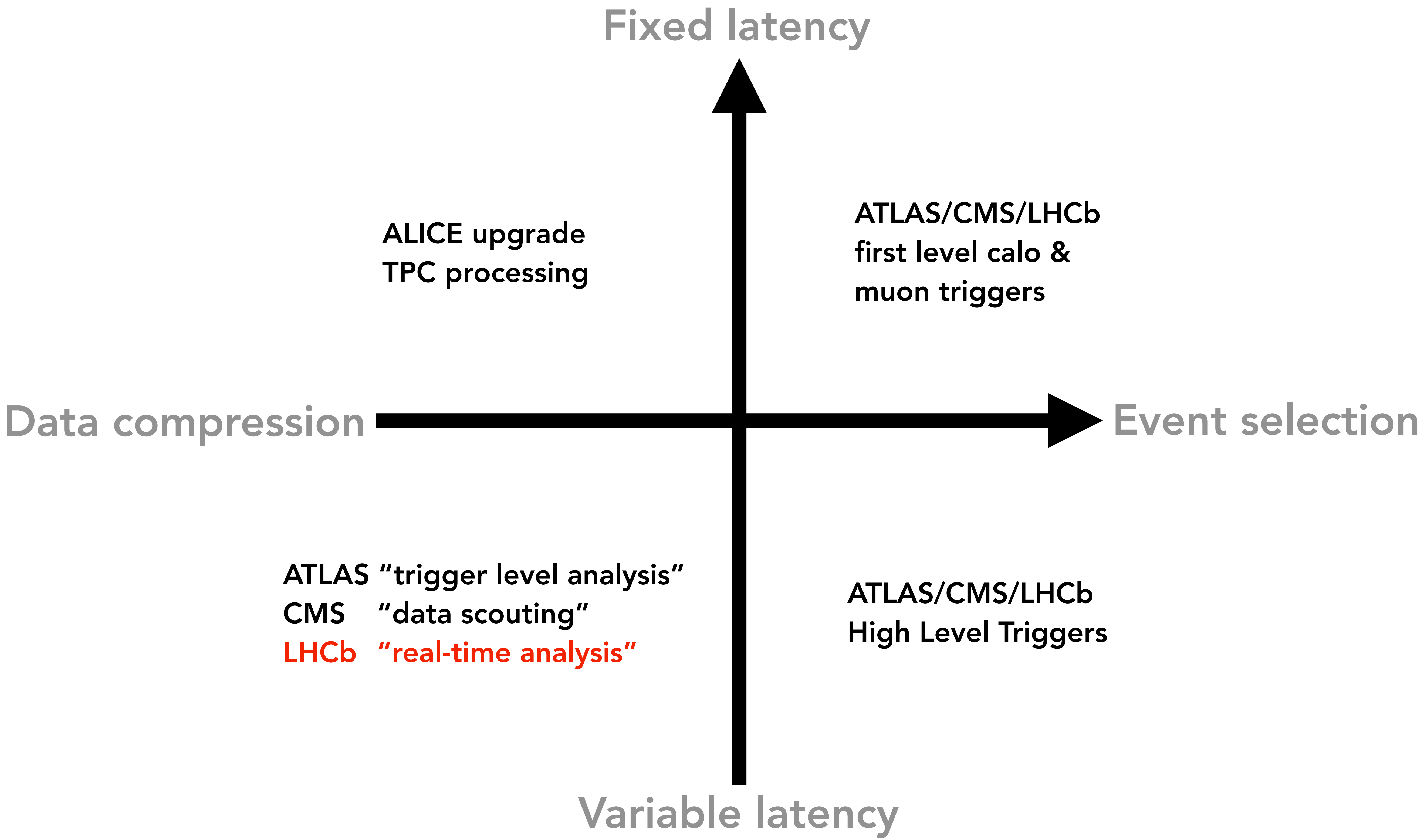
Distinguish fixed & variable latency, selection & compression

What kinds of real-time data processings exist?



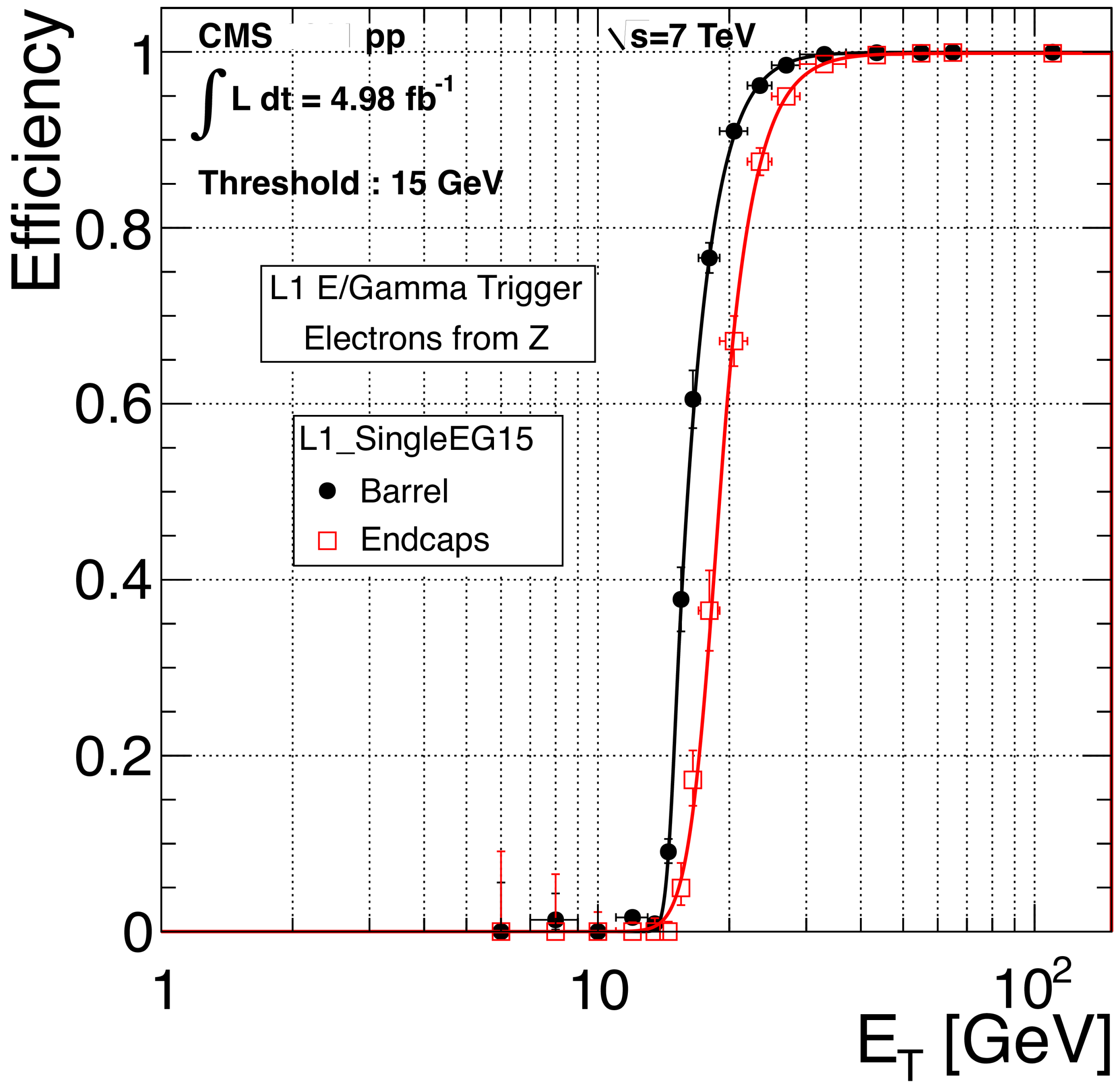
Distinguish fixed & variable latency, selection & compression

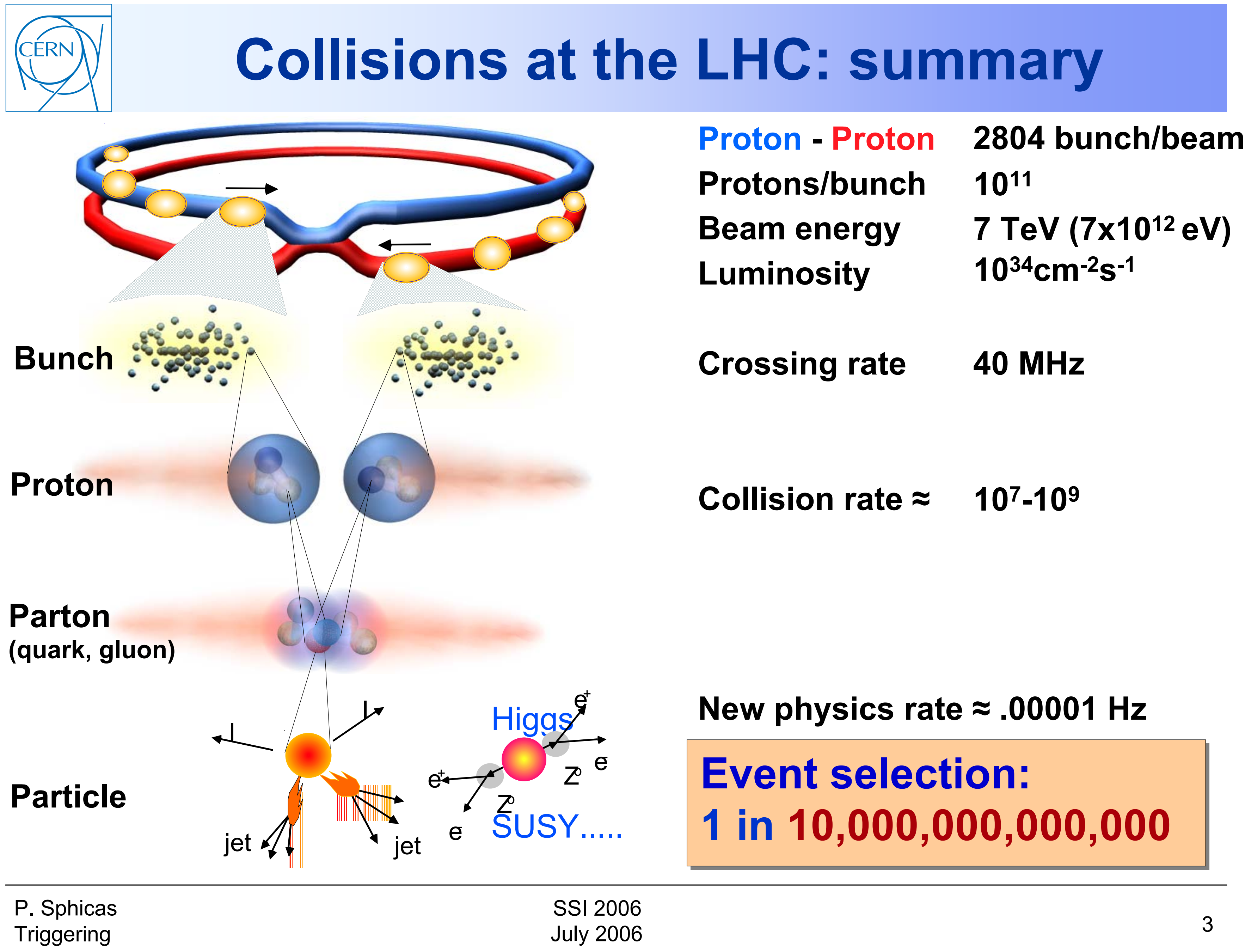
What kinds of real-time data processings exist?



Distinguish fixed & variable latency, selection & compression

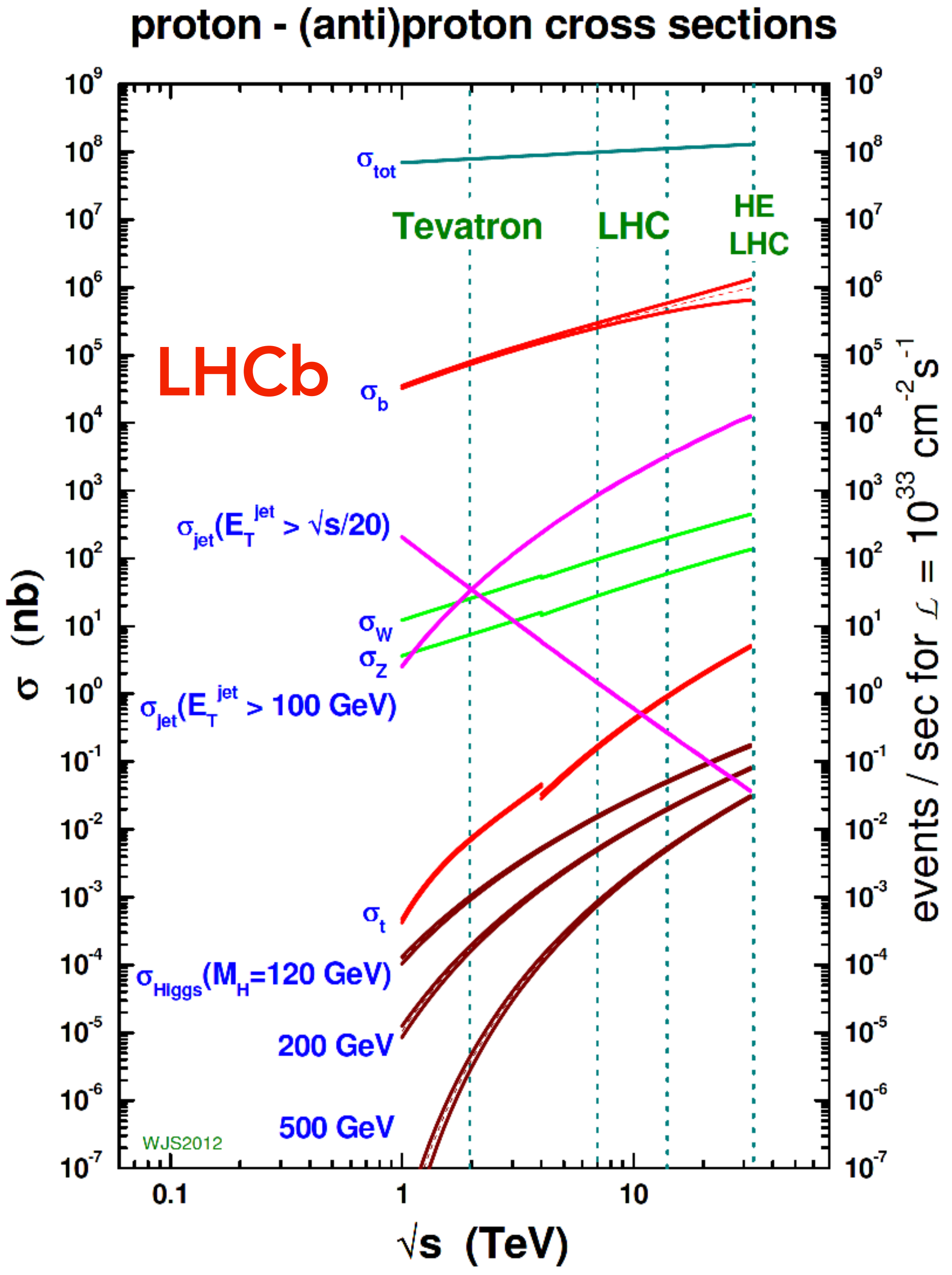
Traditional real-time processing, or "triggering"



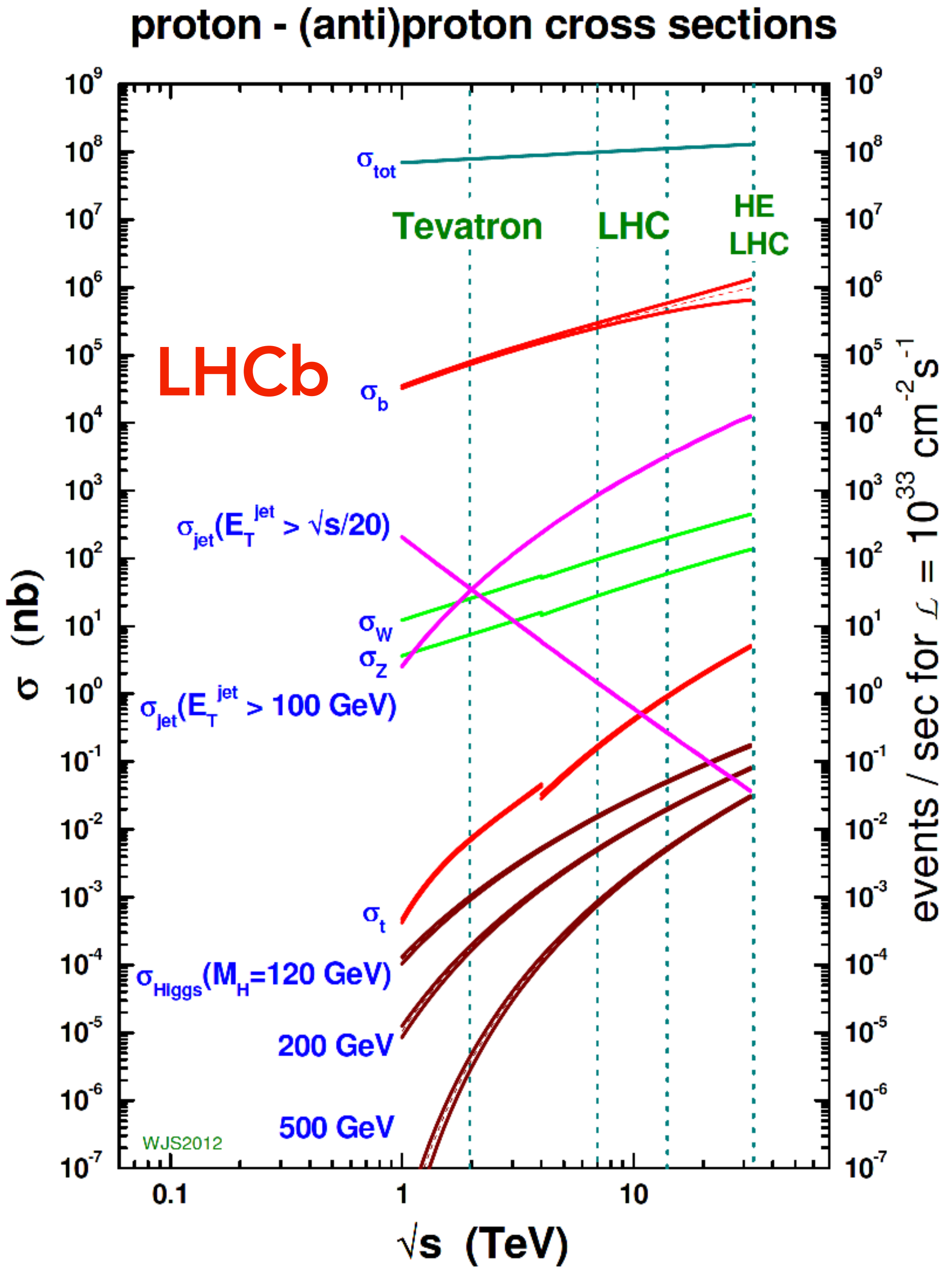


Driven by fixed-latency selection, analysis on efficiency plateau

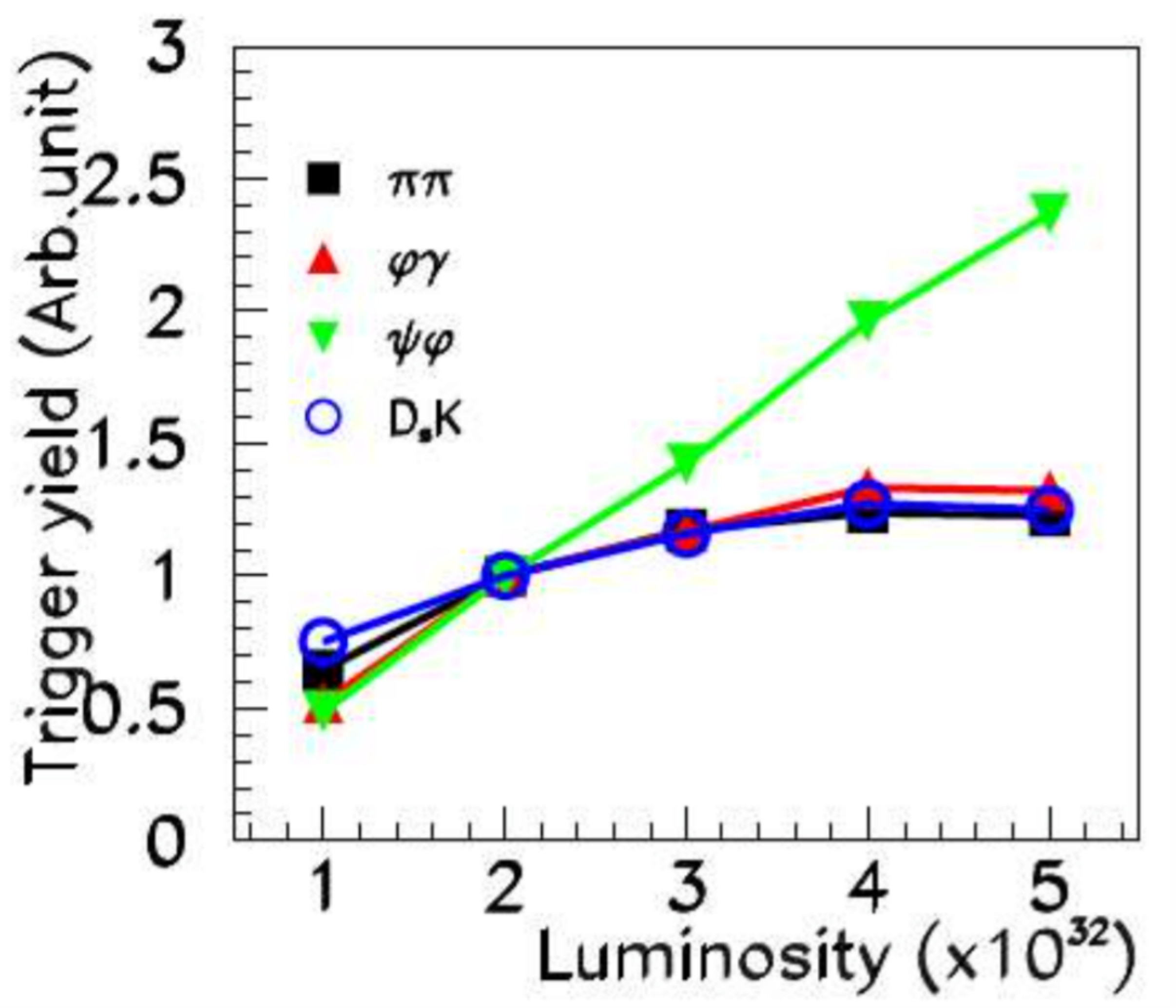
Why does LHCb not run at ATLAS/CMS luminosities today?



Why does LHCb not run at ATLAS/CMS luminosities today?

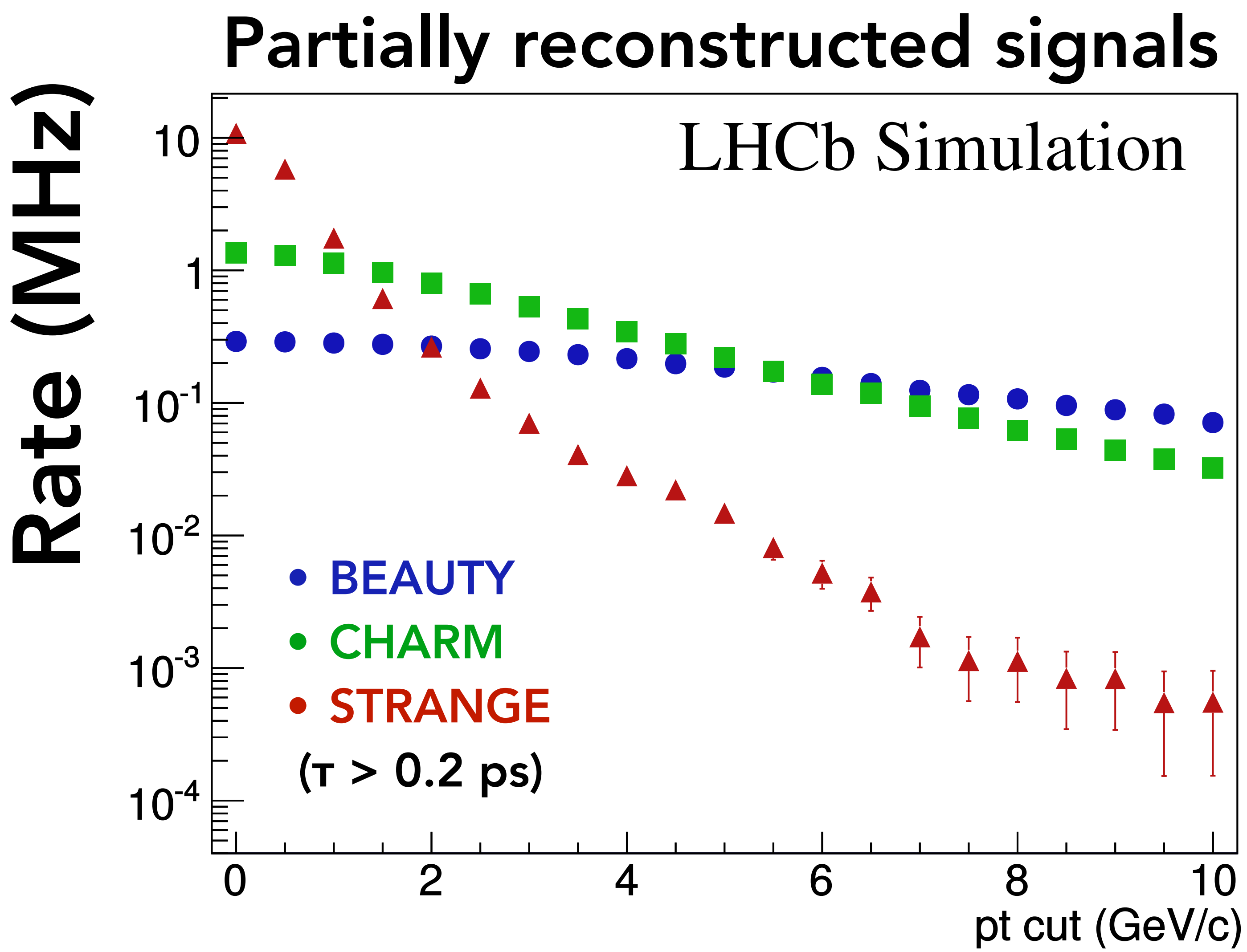


The plot which basically motivated the LHCb upgrade



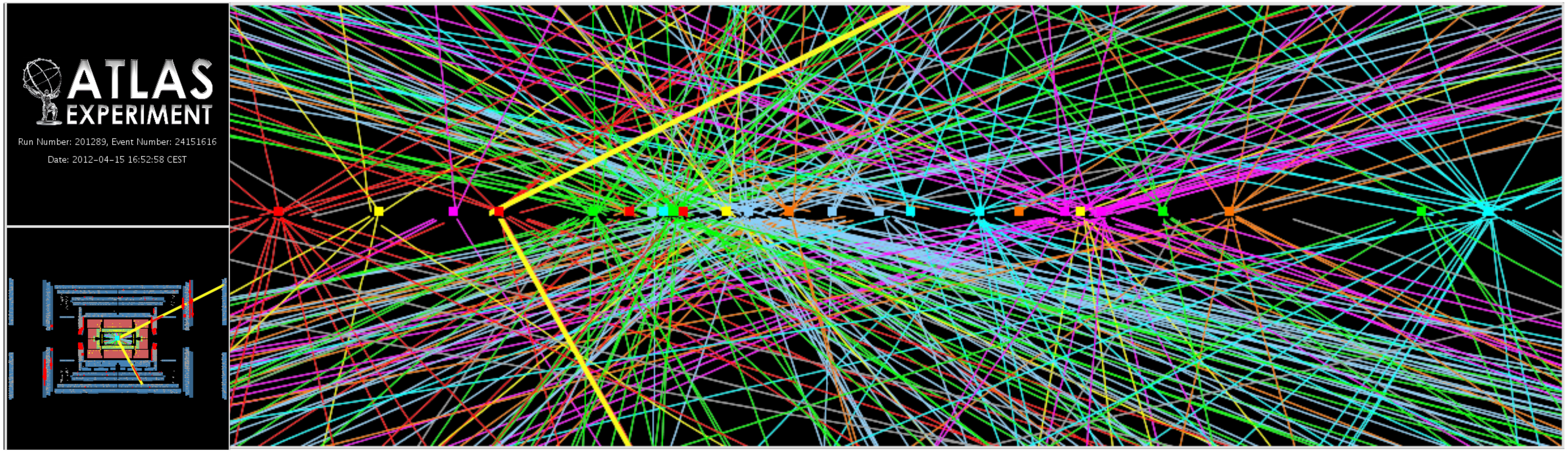
Fixed-latency CALO trigger only effective up to $4 \cdot 10^{32}$

Signal rates @ $2 \cdot 10^{33}$ in the LHCb upgrade



We will have MHz of signals in our acceptance! Can only reject up to 1/60 efficiently with inclusive selections. Require real-time analysis beyond this.

From selection to compression : real-time analysis



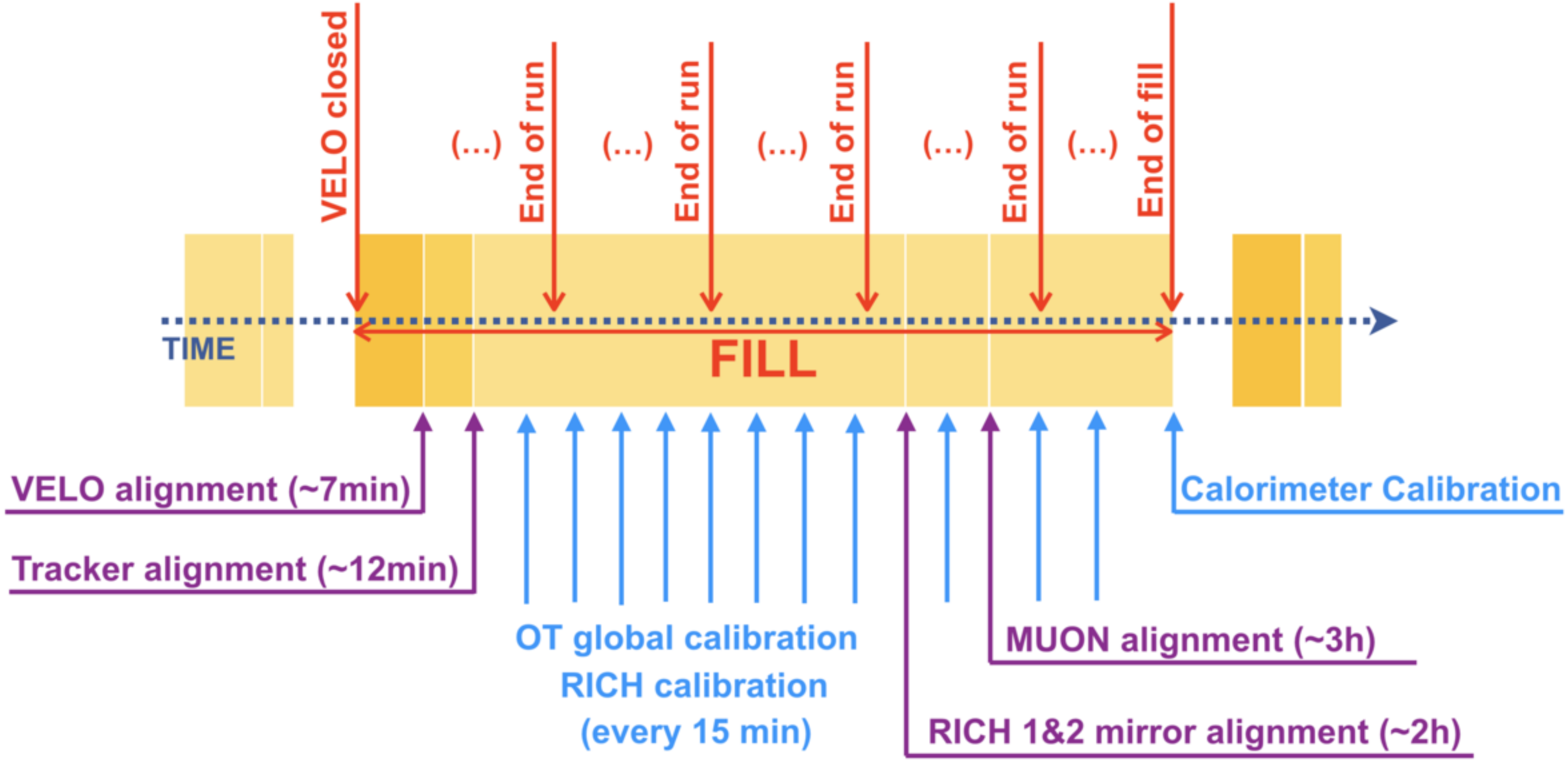
Most physics measurements require only a signal candidate and information about the specific pp collision which produced it → the rest is pileup

The higher the luminosity, the larger the fraction of event data caused by pileup

Hence create more room for signal by compressing & removing pileup in real-time!

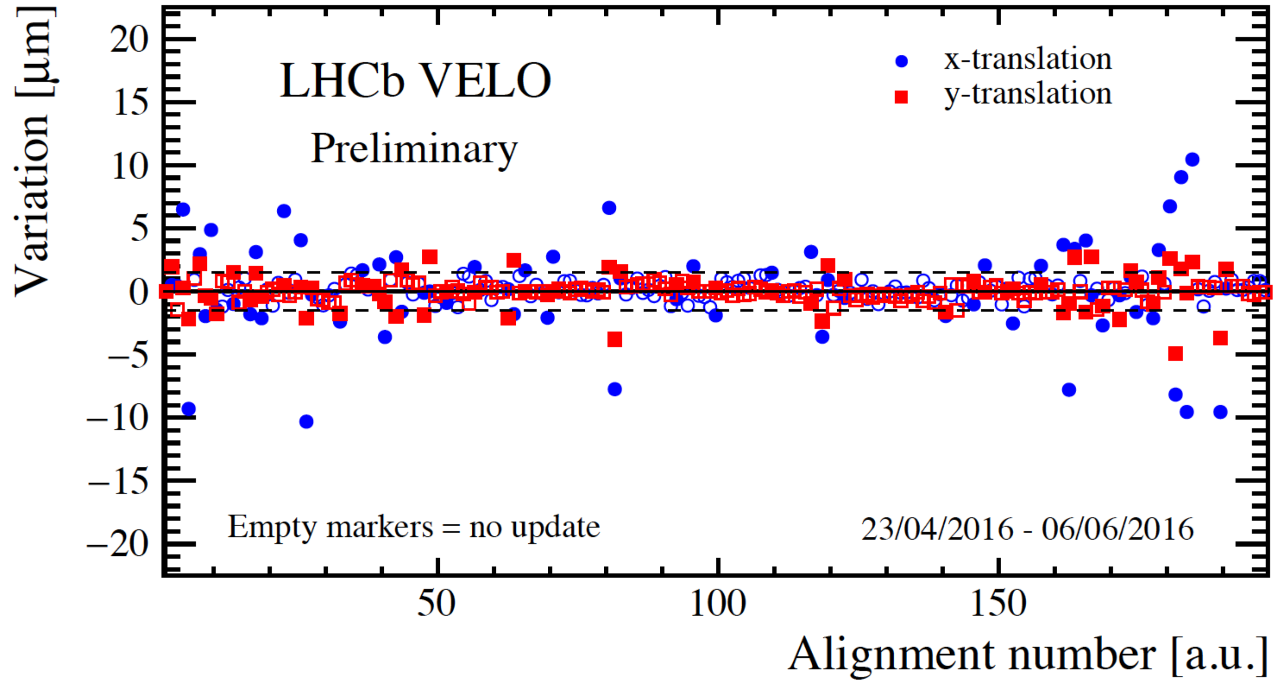
So how do we carry out precise pileup suppression?

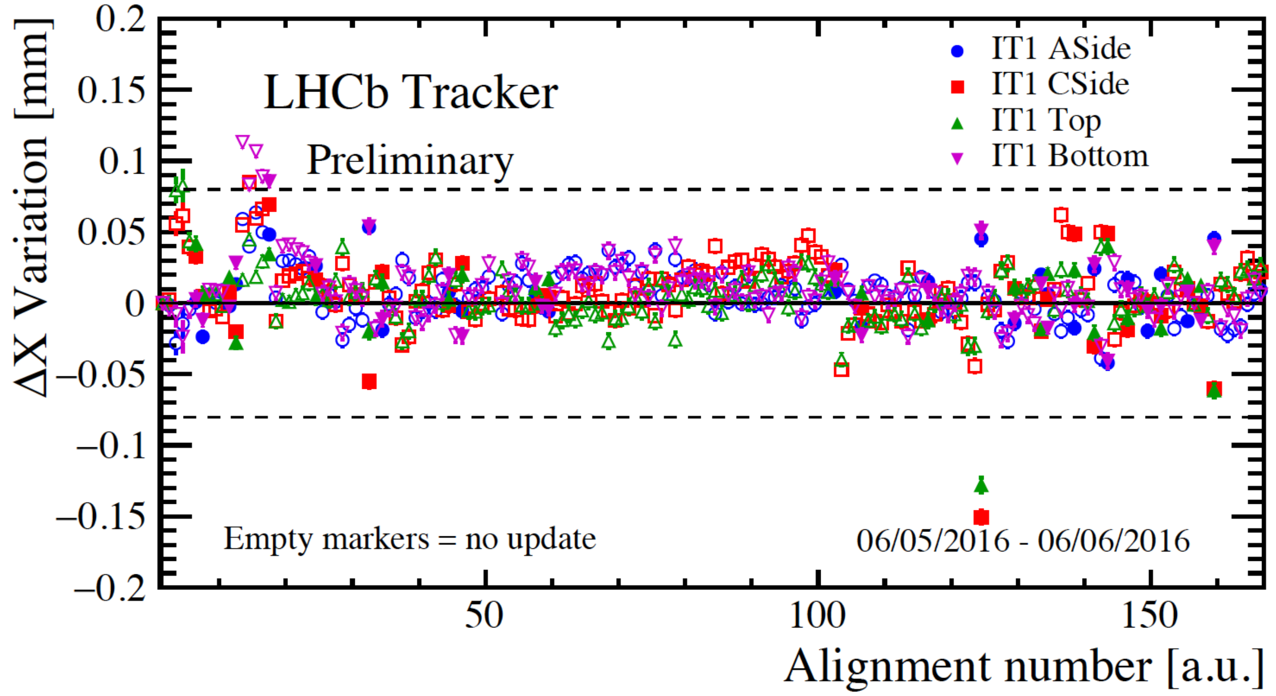
We also need to align and calibrate our detector in real time

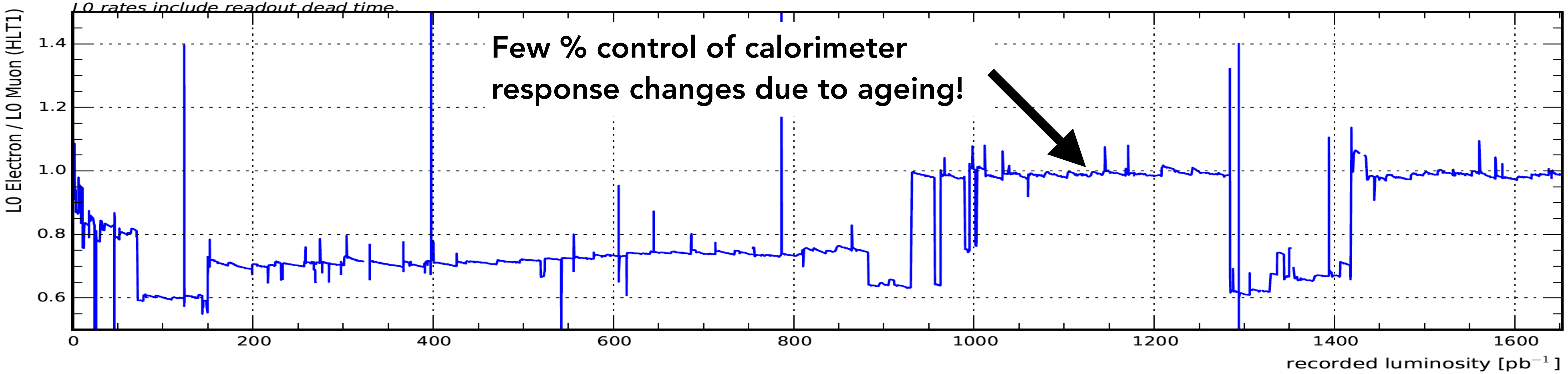


((~7min),(~12min),(~3h),(~2h)) - time needed for both data accumulation and running the task

So we did!



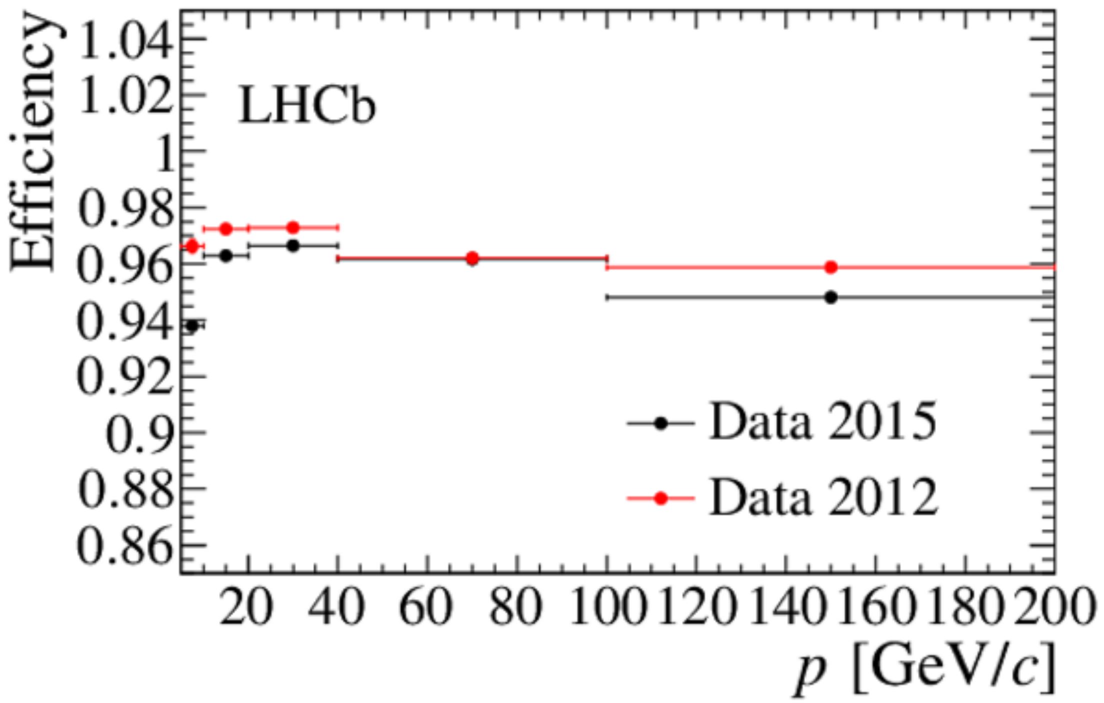


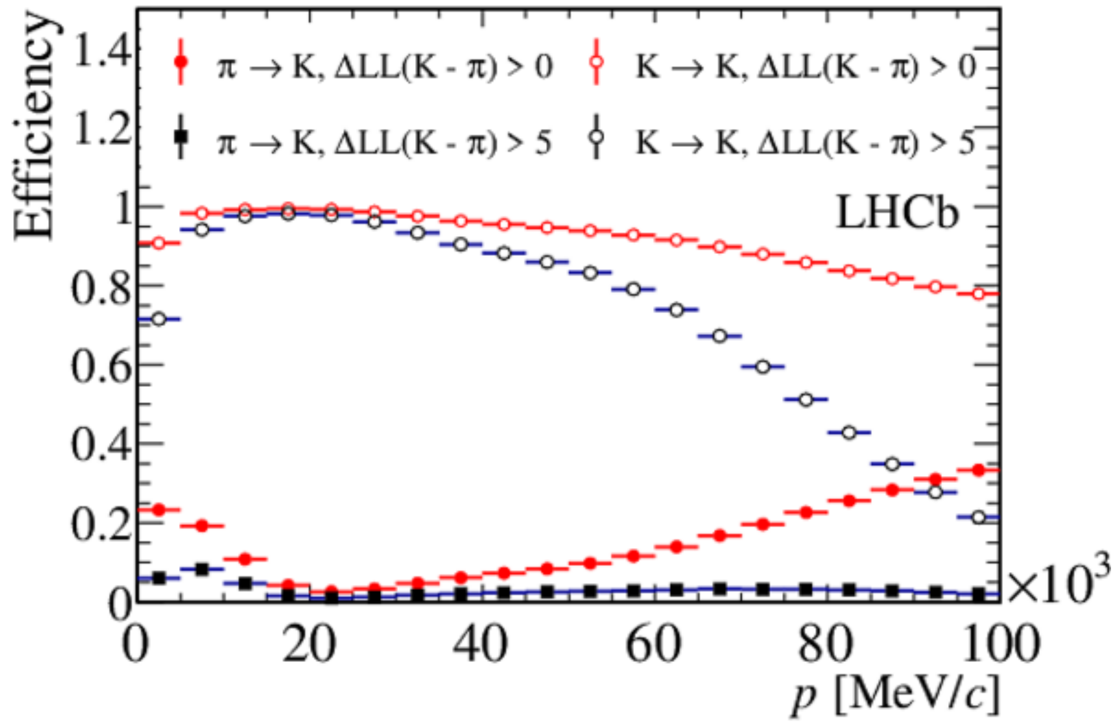


Implemented for the first time in Run 2 with offline like quality from very early in 2015. Not only tracker but also RICH and calorimeter. For me this is the most impressive aspect of LHCb's Run 2 and required a huge team effort across projects and working groups.

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

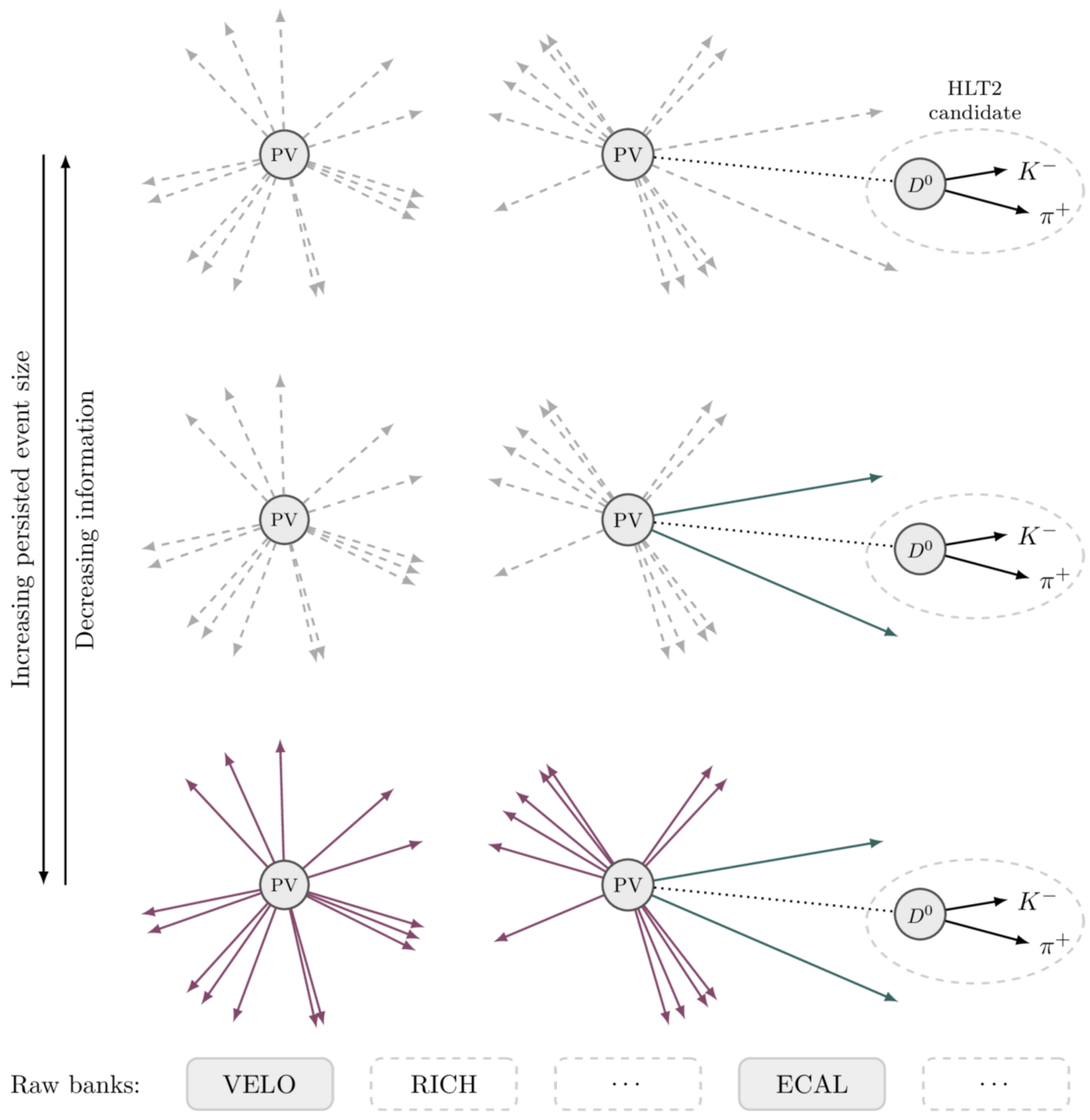
Species	Low momentum	High momentum
$e^{\pm}$	$B^+ \to J/\psi K^+$ with $J/\psi \to e^+e^-$	
$\mu^{\pm}$	$B^+ \to J/\psi K^+$ with $J/\psi \to \mu^+\mu^-$	$J/\psi \to \mu^+\mu^-$
$\pi^{\pm}$	$K^0_S \to \pi^+\pi^-$	$D^{*+} \to D^0\pi^+$ with $D^0 \to K^-\pi^+$
$K^{\pm}$	$D^+_s \to \phi\pi^+$ with $\phi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





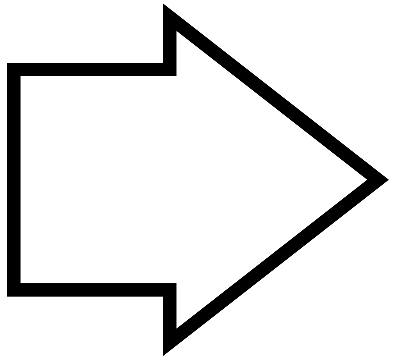
Unlike ATLAS and CMS, LHCb must maintain a data-driven permille level control of its efficiency across the kinematic and geometric acceptance of the detector. Requires collecting an extremely wide range of tag-and-probe samples in real time.

Then select signals and associate them to pp collisions

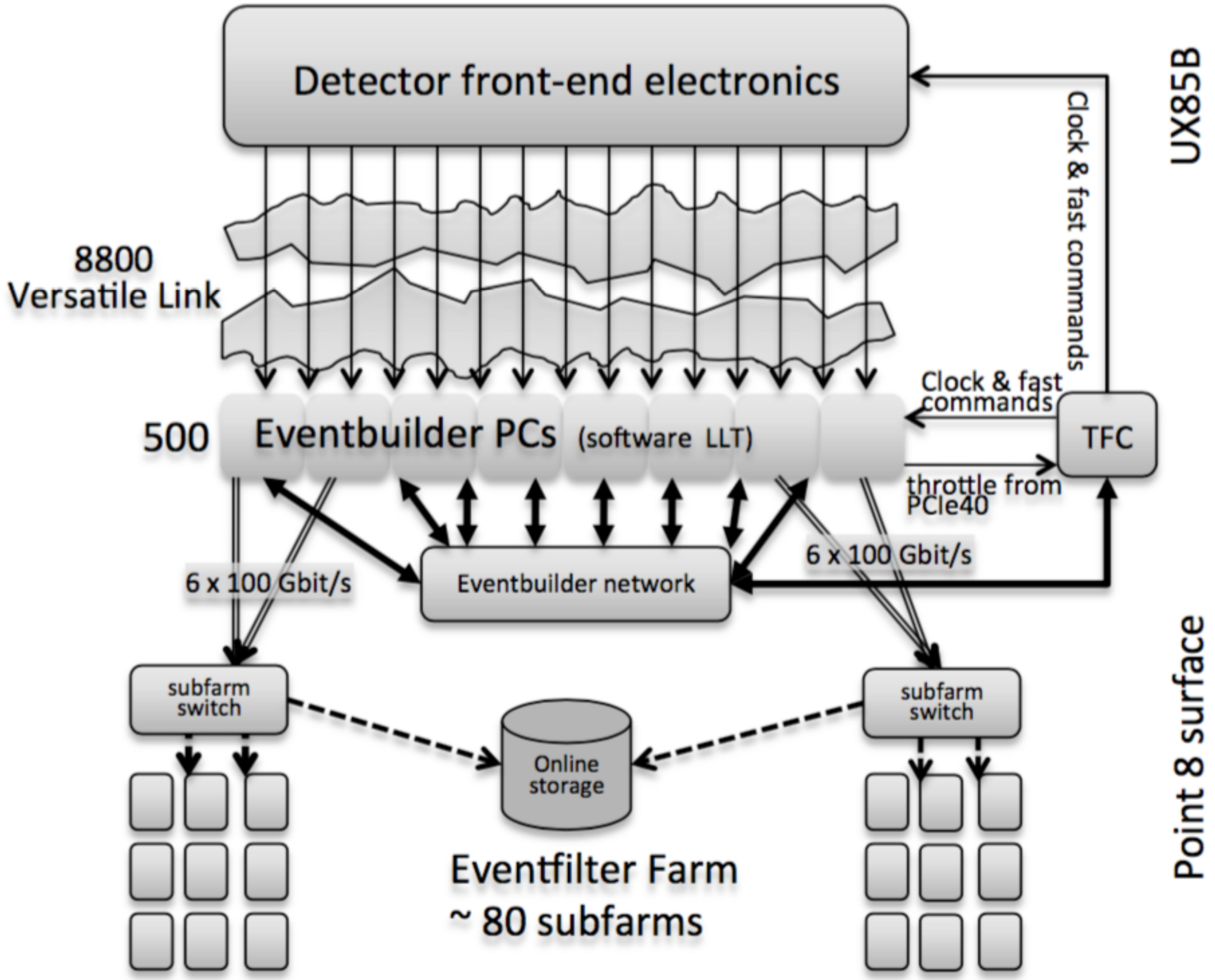


Full flexibility to store "additional" detector information if required by some analyses

Or in a picture…



From this follows the LHCb DAQ design for the upgrade



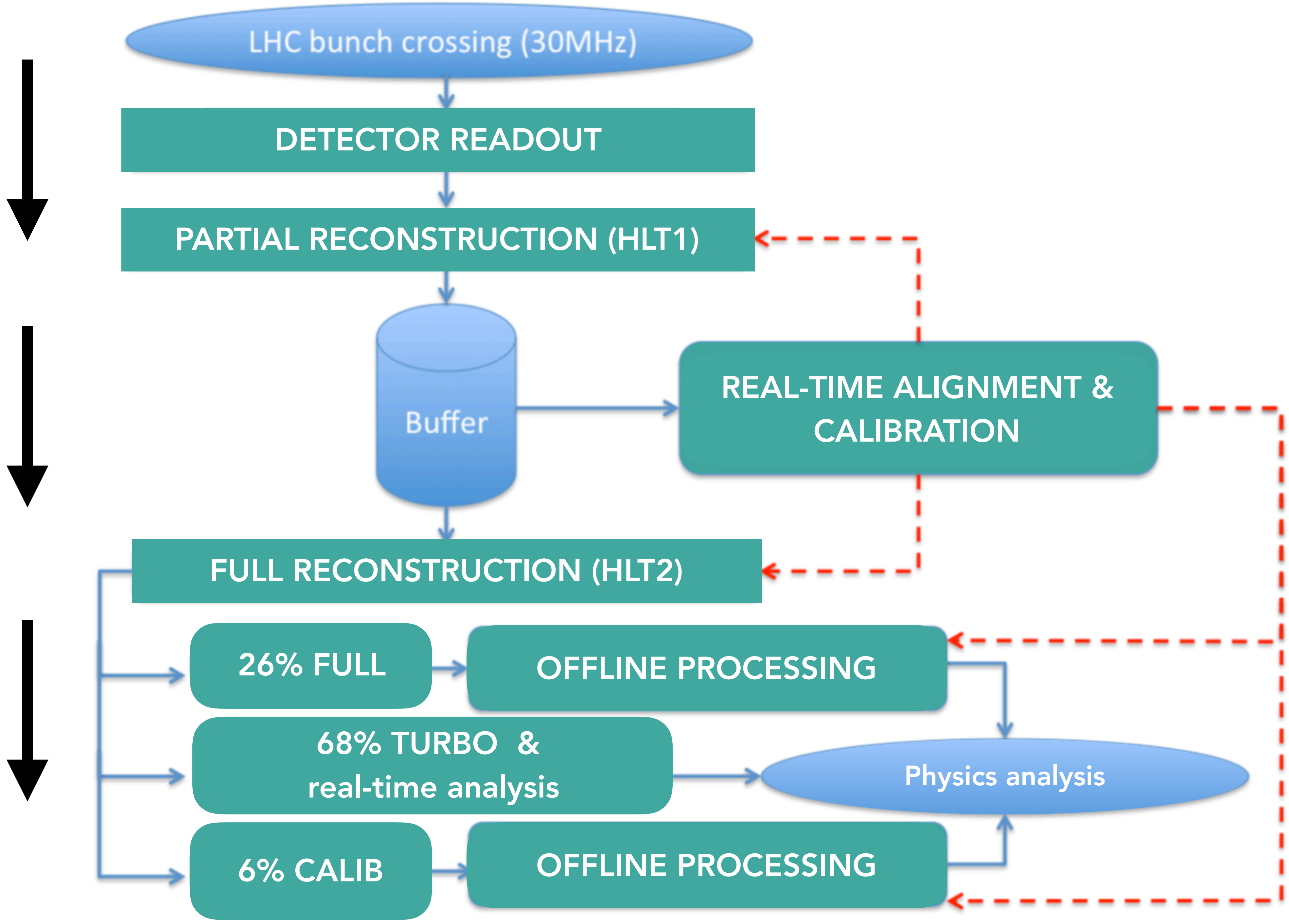
40 Tbit/s full event building & processing in a data centre

LHCb upgrade dataflow

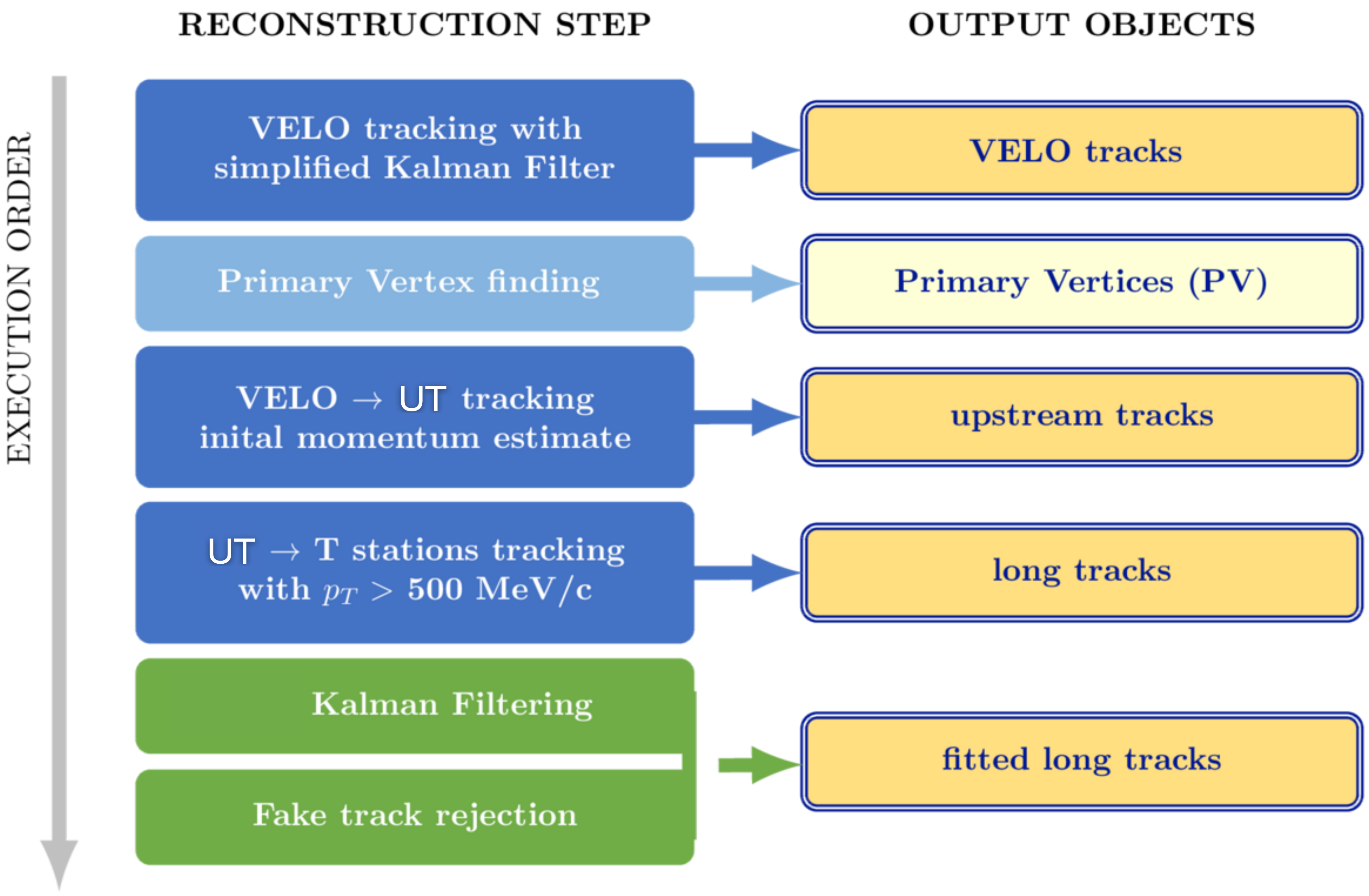
40 Tb/s
30 MHz non-empty pp

1-2 Tb/s
0.5 - 1.5 MHz

80 Gb/s

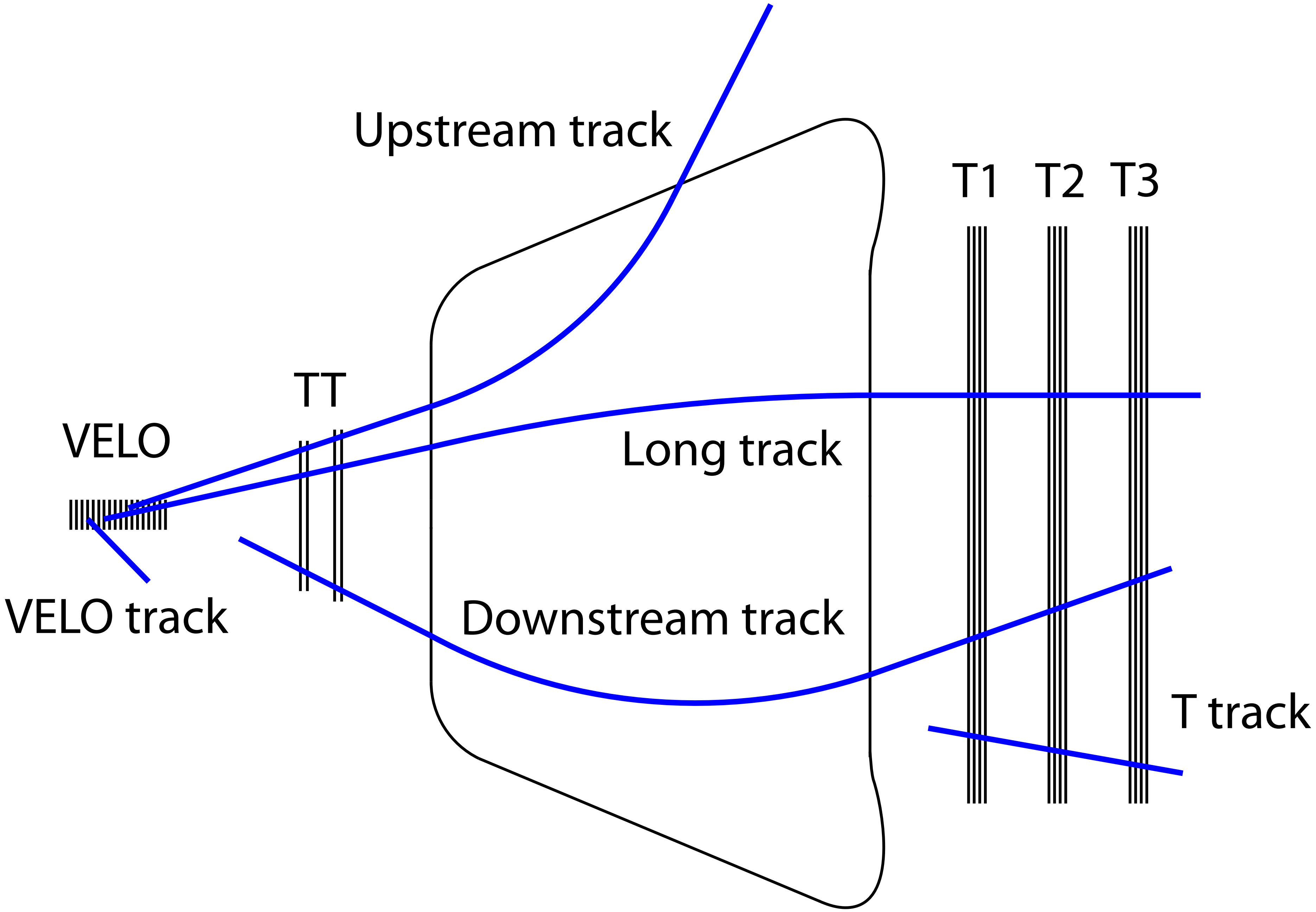


What is the physics content of HLT1 which runs @30 MHz?



"Traditional" inclusive selections selecting bunch crossings.
Must be based on tracks, so require 30 MHz tracking at $2 \cdot 10^{33}$!

But what does that have to do with latency?



Because LHCb is a dipole spectrometer, tracking inherently requires non-local data from multiple subdetectors to be brought together.

You can build a fixed-latency track trigger but you will have to build the biggest part of the detector readout for it anyway — might as well just read everything out upfront and work in variable latency.

This is *not* an argument about e.g. not using FPGAs, just you first build events, then process them in whatever way is most cost-effective.

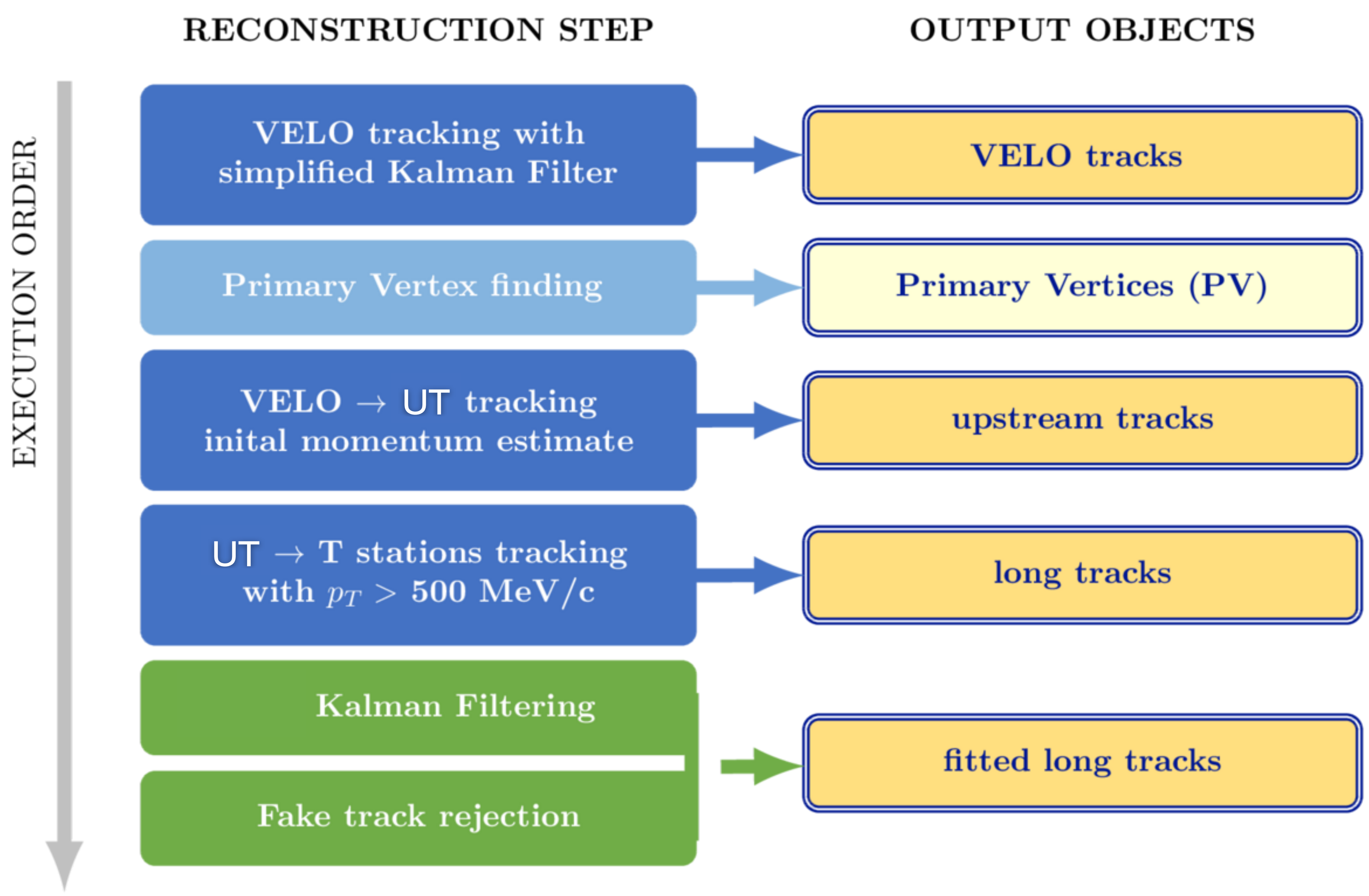
Pause and compare this to ATLAS/CMS HL-LHC processing

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

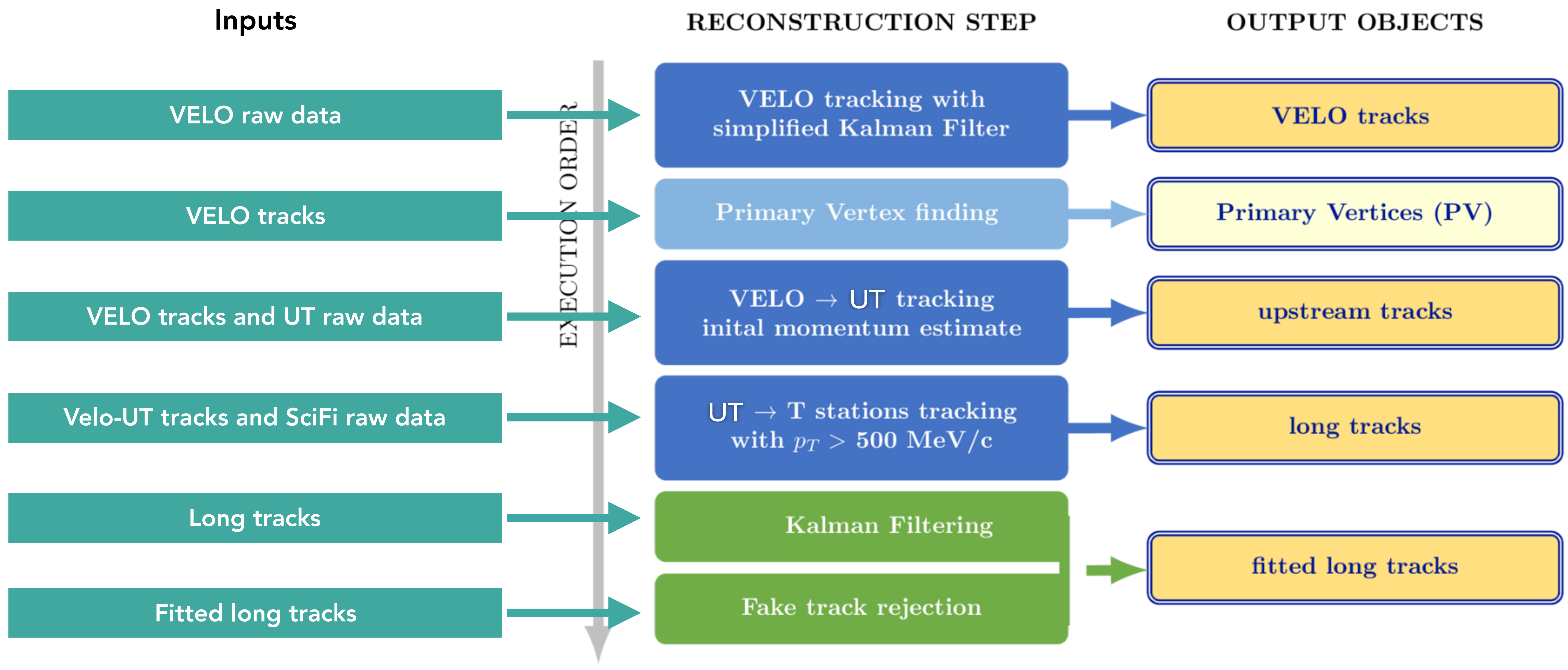
LHCb 2021 real-time tracking has to handle the same data volume as ATLAS/CMS HL-LHC upgrades! But earlier and for less money...

Challenges and solutions

Let's look at this sequence in more detail

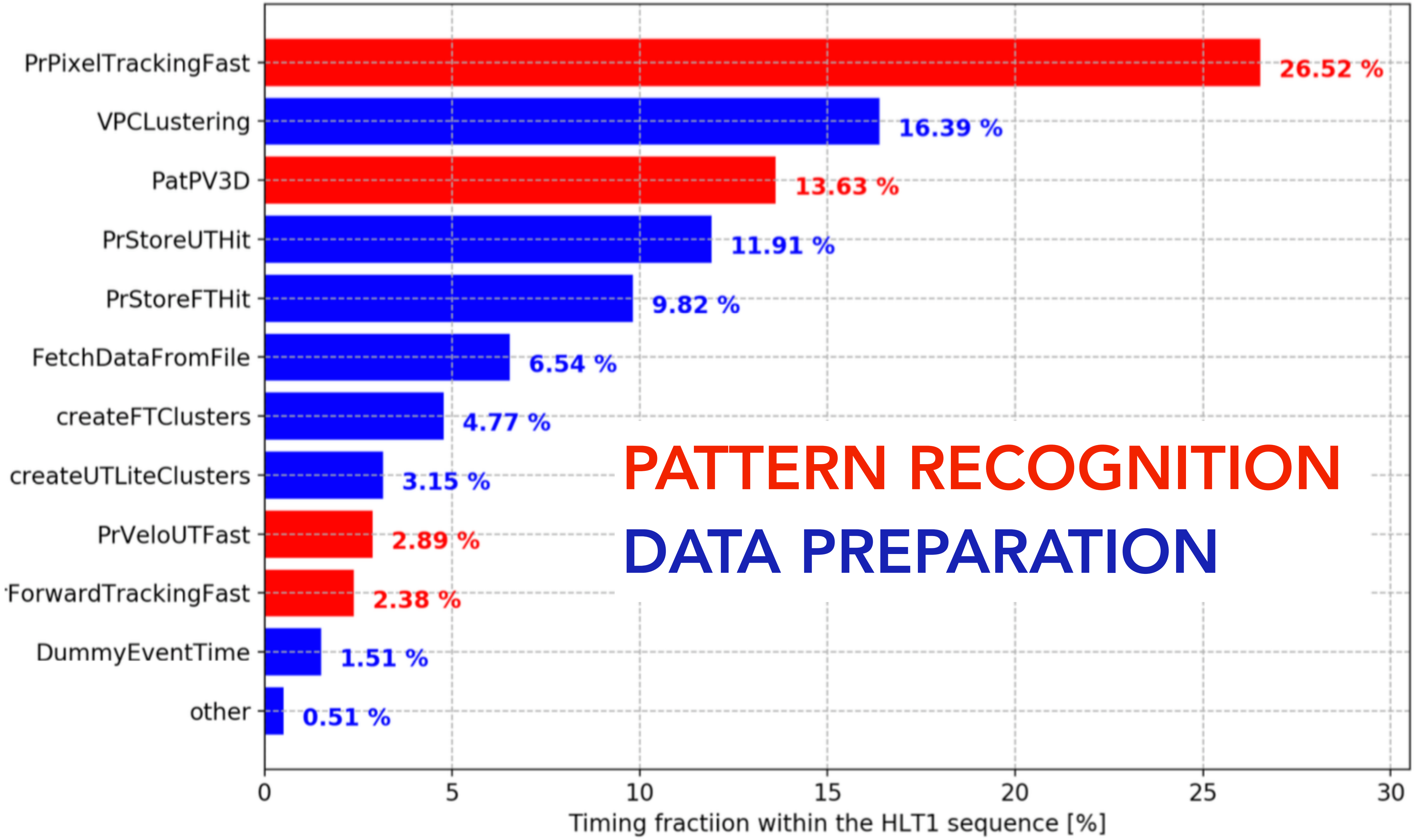


Let's look at this sequence in more detail



To run at 30 MHz we need to get data off the detector, transform it to the global coordinate system, and do pattern recognition

Where did we start from? Roughly 3 MHz...

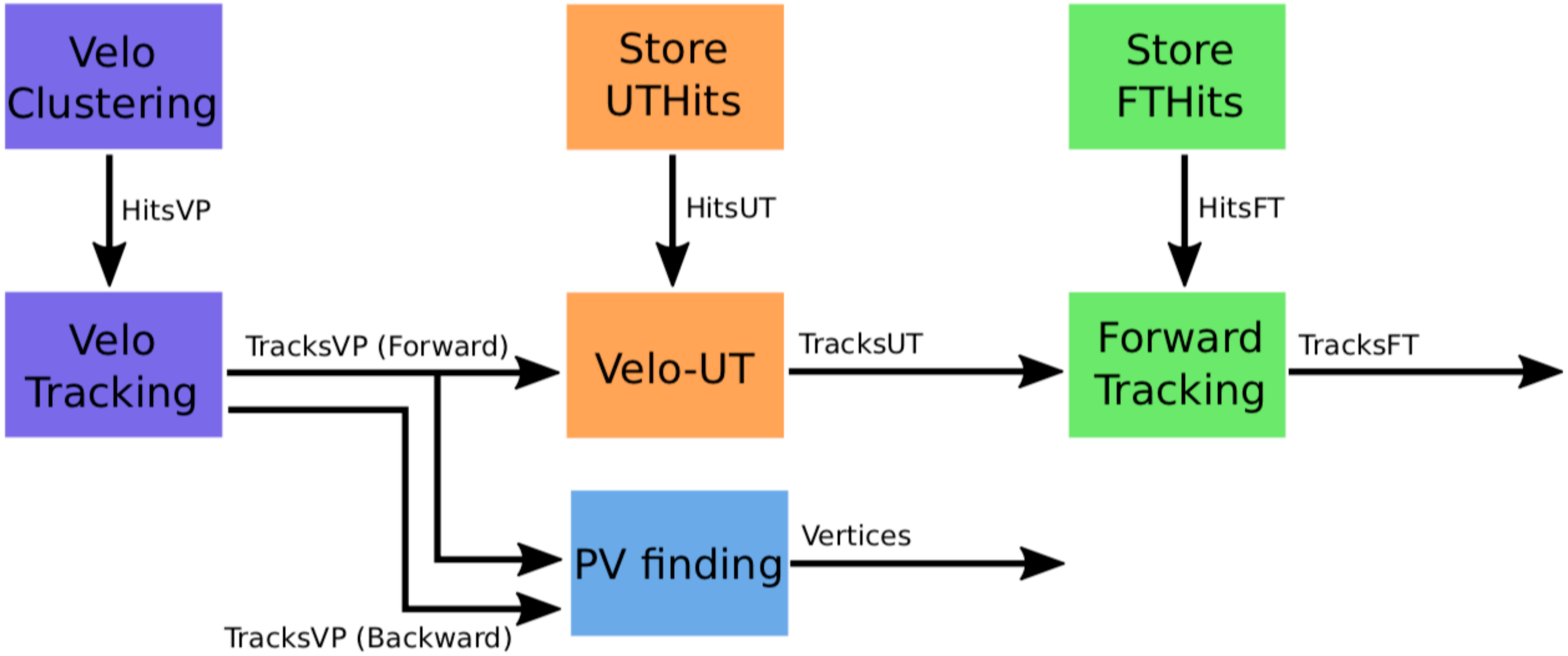


Early 2018 after about 3 years of work to make the framework thread-safe. Data preparation as important as pattern reco!

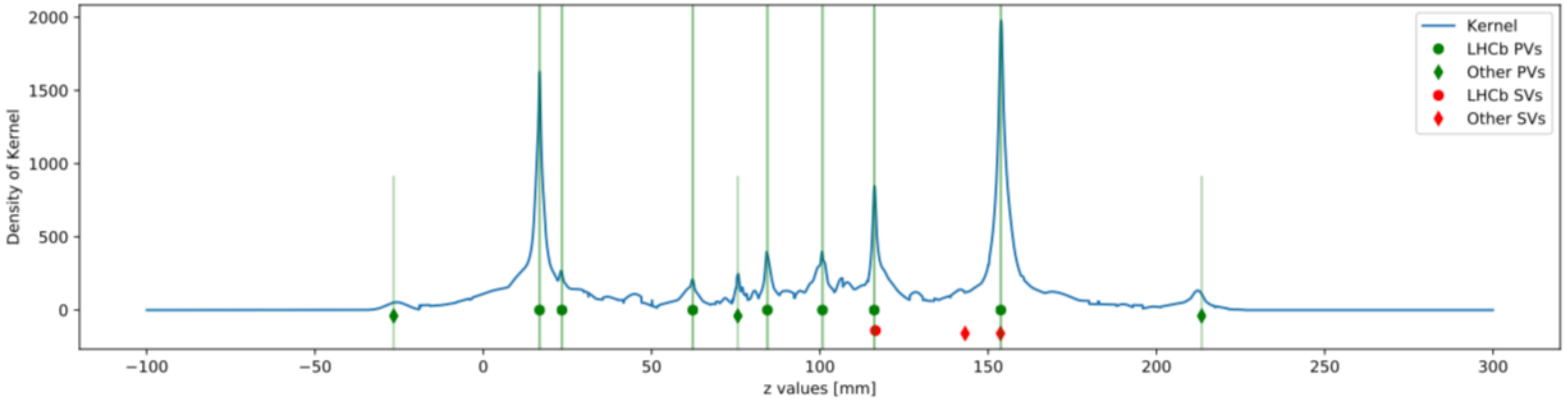
How to improve it?

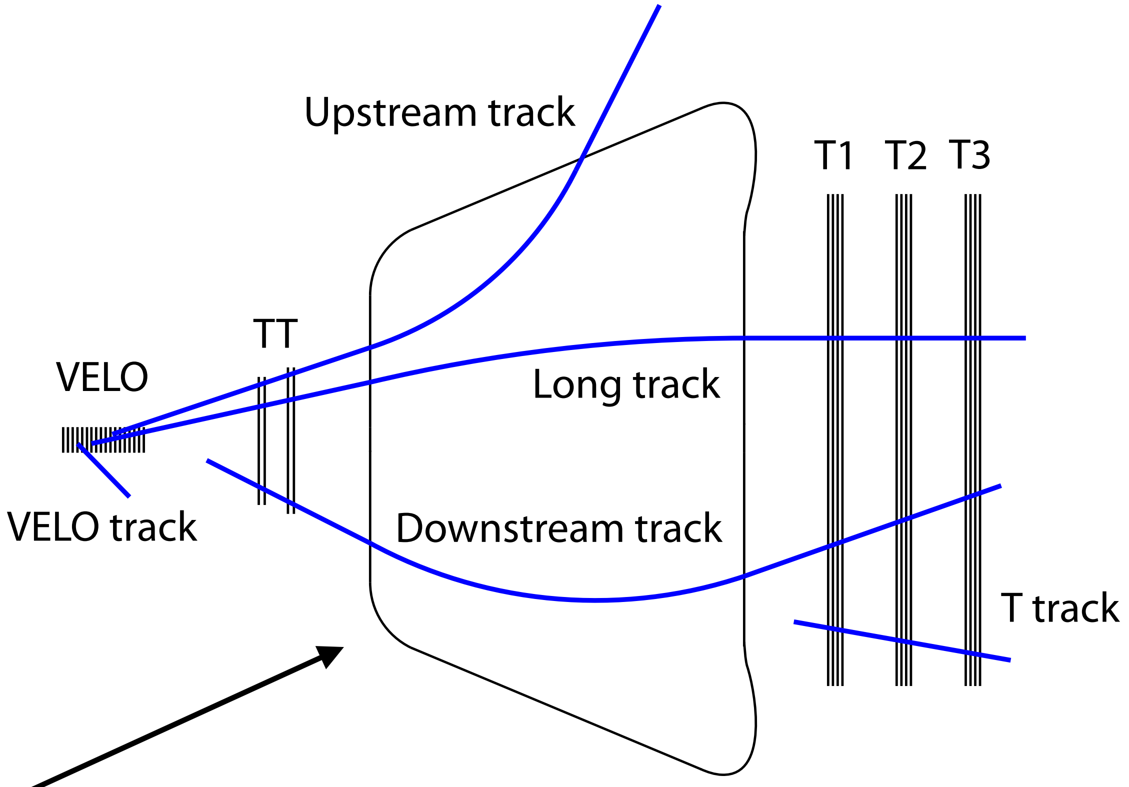
1. Do what you can on the readout boards! Output the data in the most useful format possible, perform clustering in the readout if you can.
2. Write custom throughput oriented data structures which only contain the absolute minimum needed by pattern recognition. “Plain old data”.
3. Work with SOA structures wherever possible to enable vectorization.
4. Minimize copying of information by breaking up large structures, for example tracks, into smaller pieces — for example track parameters and indices pointing to the track hits in one place, tracks states in another, fit results in a third. Prefer to join these later when needed.

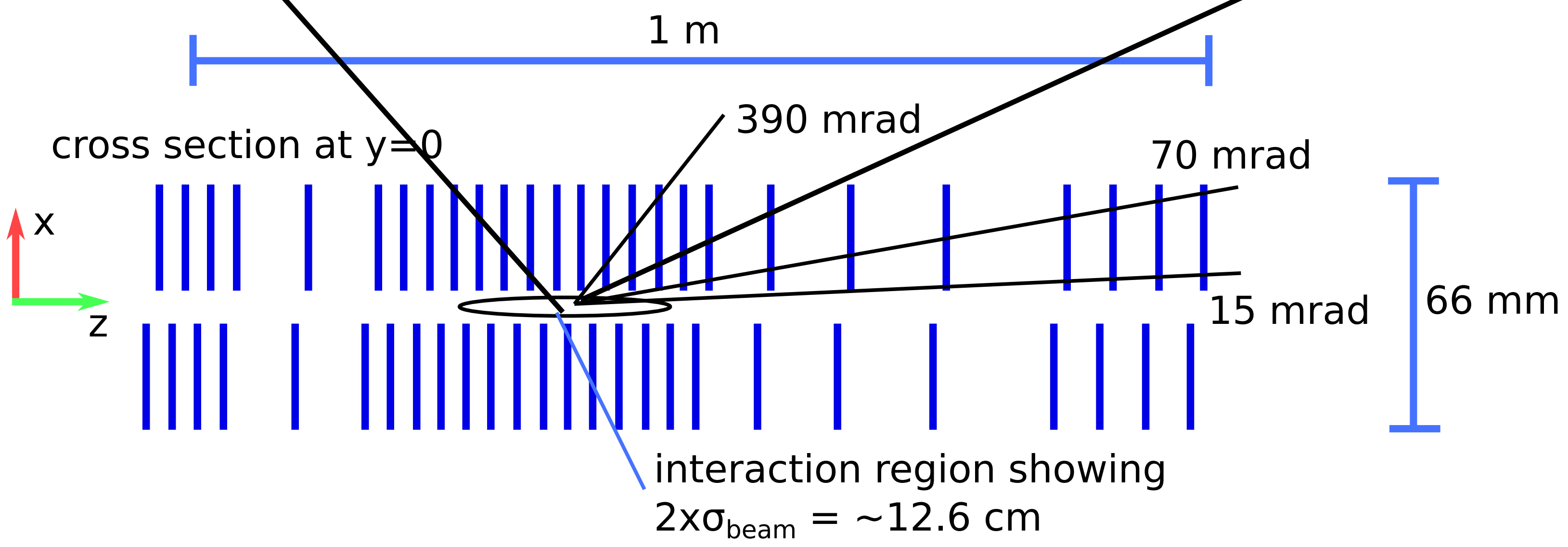
So what does the new sequence then concretely look like?



Small illustrative example – why split the VELO tracks?

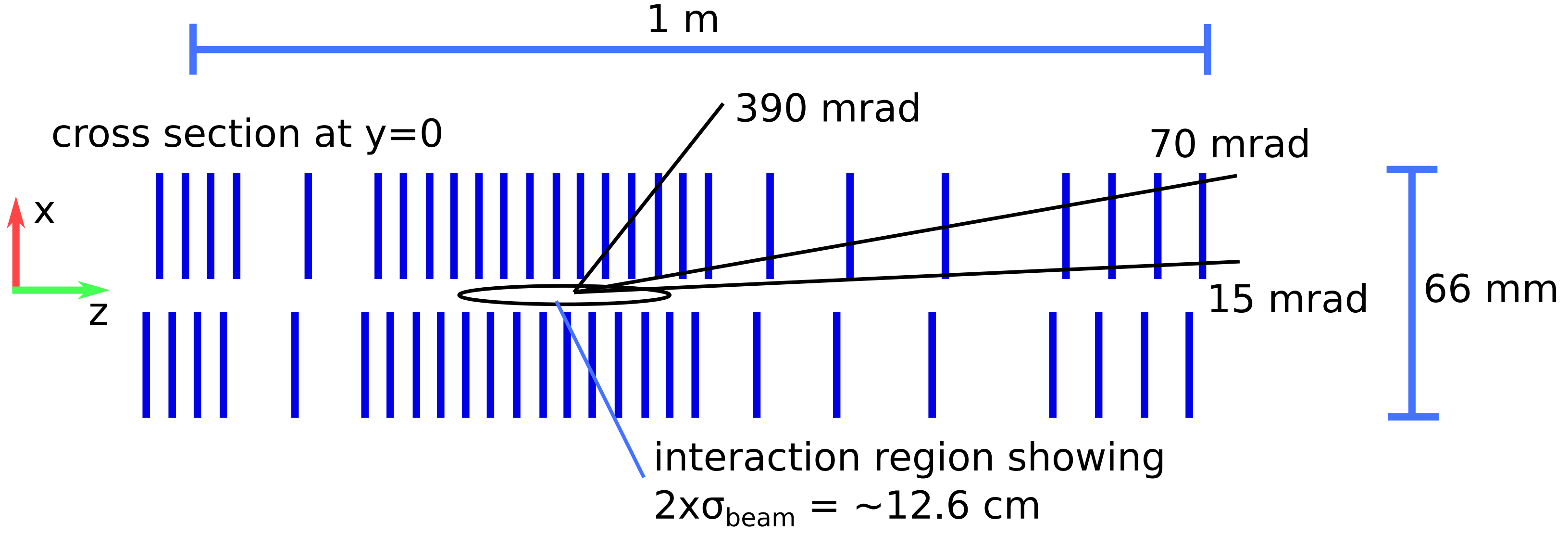


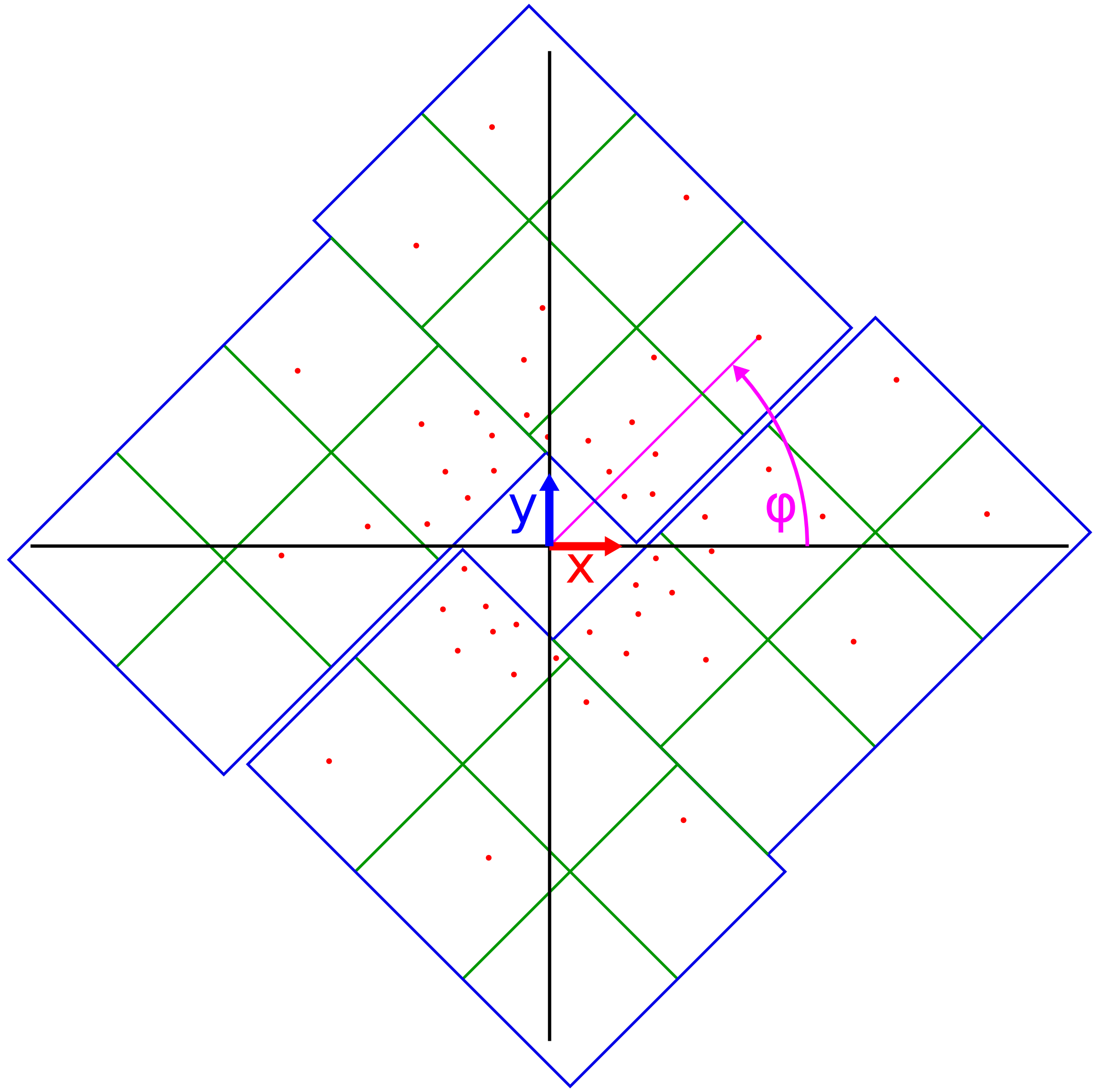




Emphasize memory locality of objects used for similar tasks

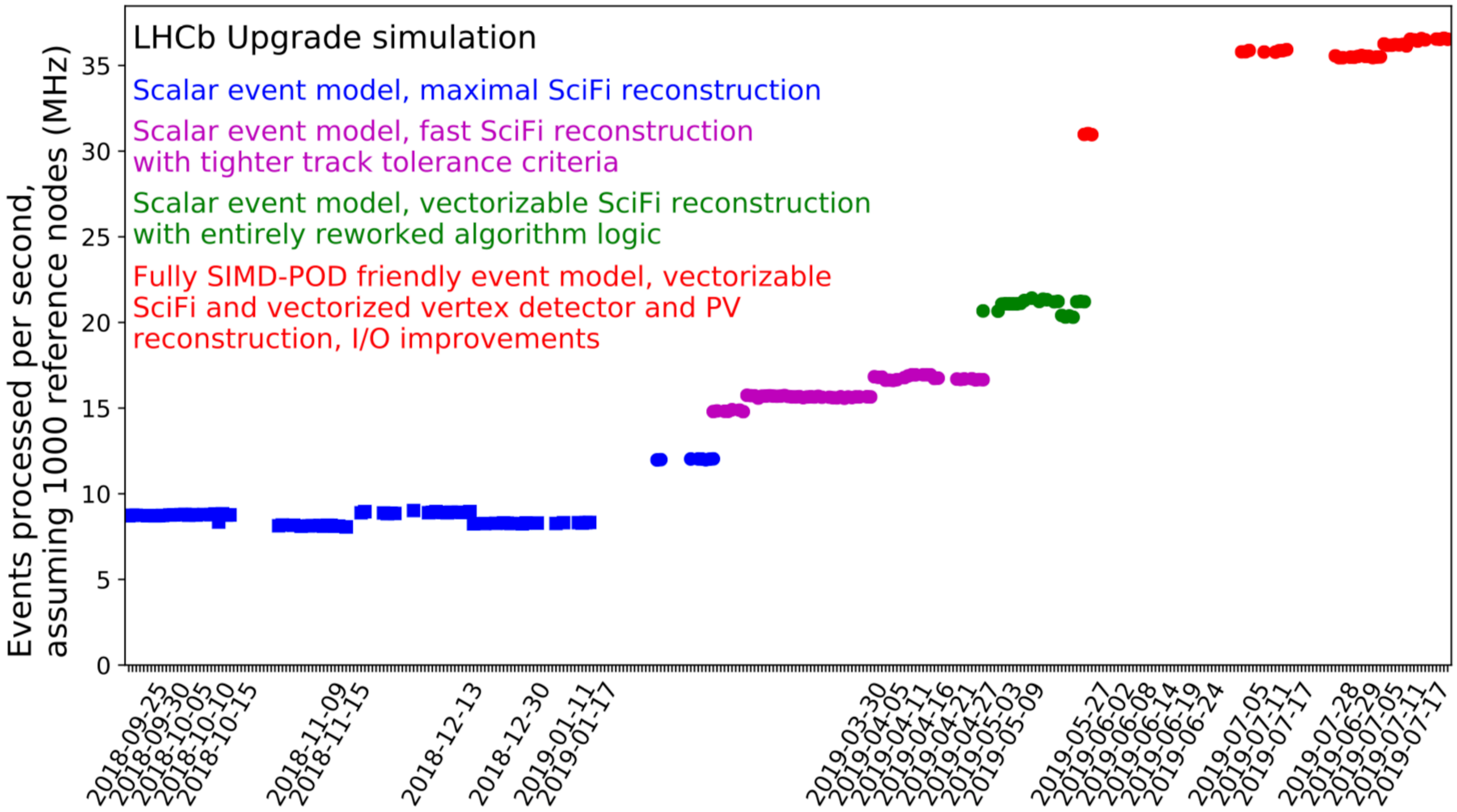
Another illustrative example – use the detector geometry





VELO tracks from the beamline traverse lines of constant φ
When extrapolating, looking for N nearest neighbours in φ is more effective than searching for hits in a search window in φ

And we are there!



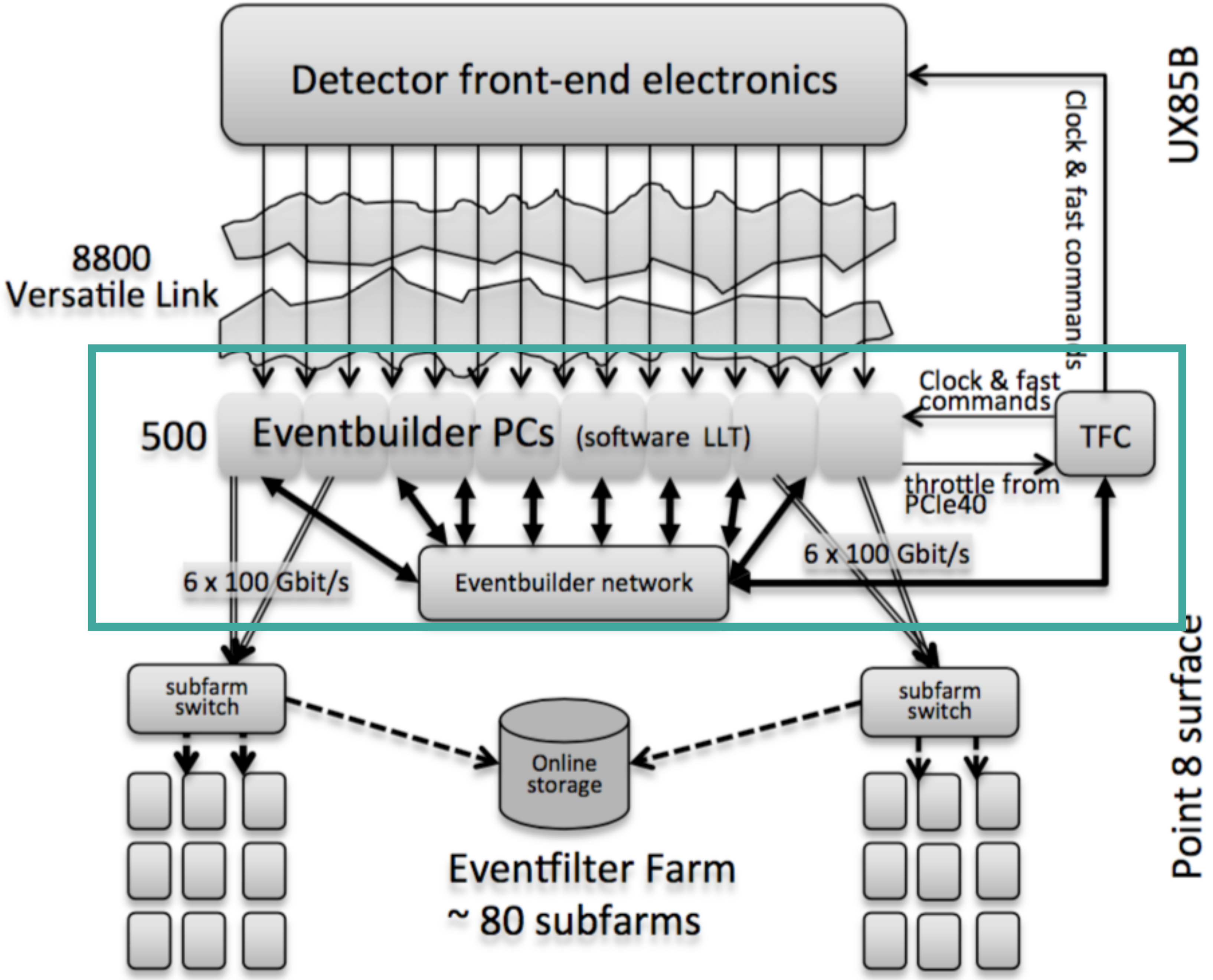
Required HLT1 throughput achieved in 2019! Gains from plain and local data and vectorization add non-linearly.

And we also developed a GPU HLT1!

LHCb-ANA-20XX-YYY
May 31, 2019

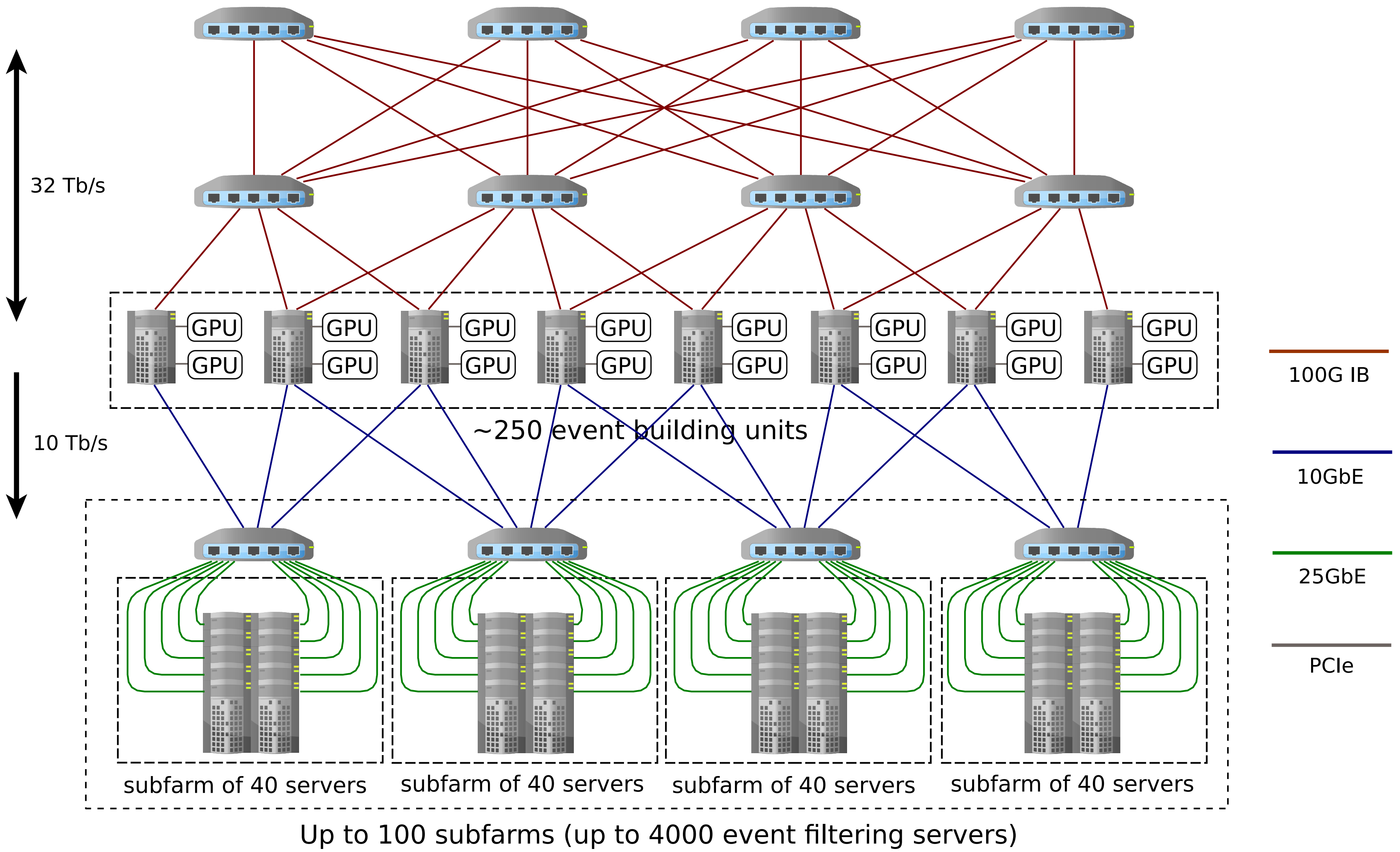
Proposal for an HLT1 implementation on GPUs for the LHCb experiment

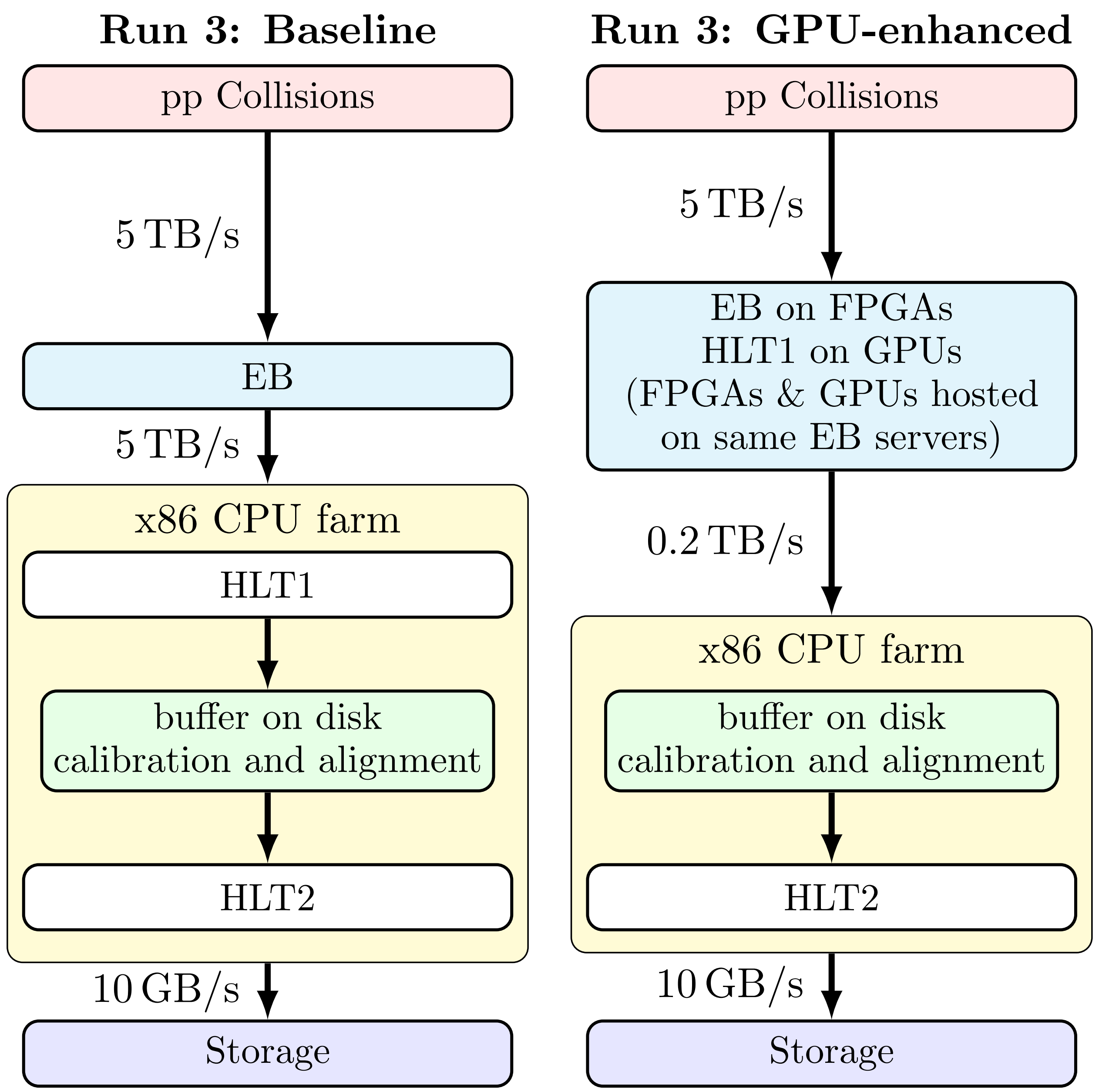
R. Aaij[1], J. Albrecht[2], M. Belous[a,3], T. Boettcher[4], A. Brea Rodríguez[5], D. vom Bruch[6], D. H. Cámpora Pérez[b,7], A. Casais Vidal[5], P. Fernandez Declara[c,7], L. Funke[2], V. V. Gligorov[6], B. Jashal[9], N. Kazeev[a,3], D. Martínez Santos[5], F. Pisani[d,e,7], D. Pliushchenko[f,3], S. Popov[a,3], M. Rangel[10], F. Reiss[6], C. Sánchez Mayordomo[9], R. Schwemmer[7], M. Sokoloff[11], A. Ustyuzhanin[a,3], X. Vilasís-Cardona[8], M. Williams[4]



Exploits flexibility of our Run 3 DAQ by implementing HLT1 directly in the servers receiving the data from the detector. Judged viable by external review, full cost-benefit analysis ongoing to decide if we will use this in Run 3.

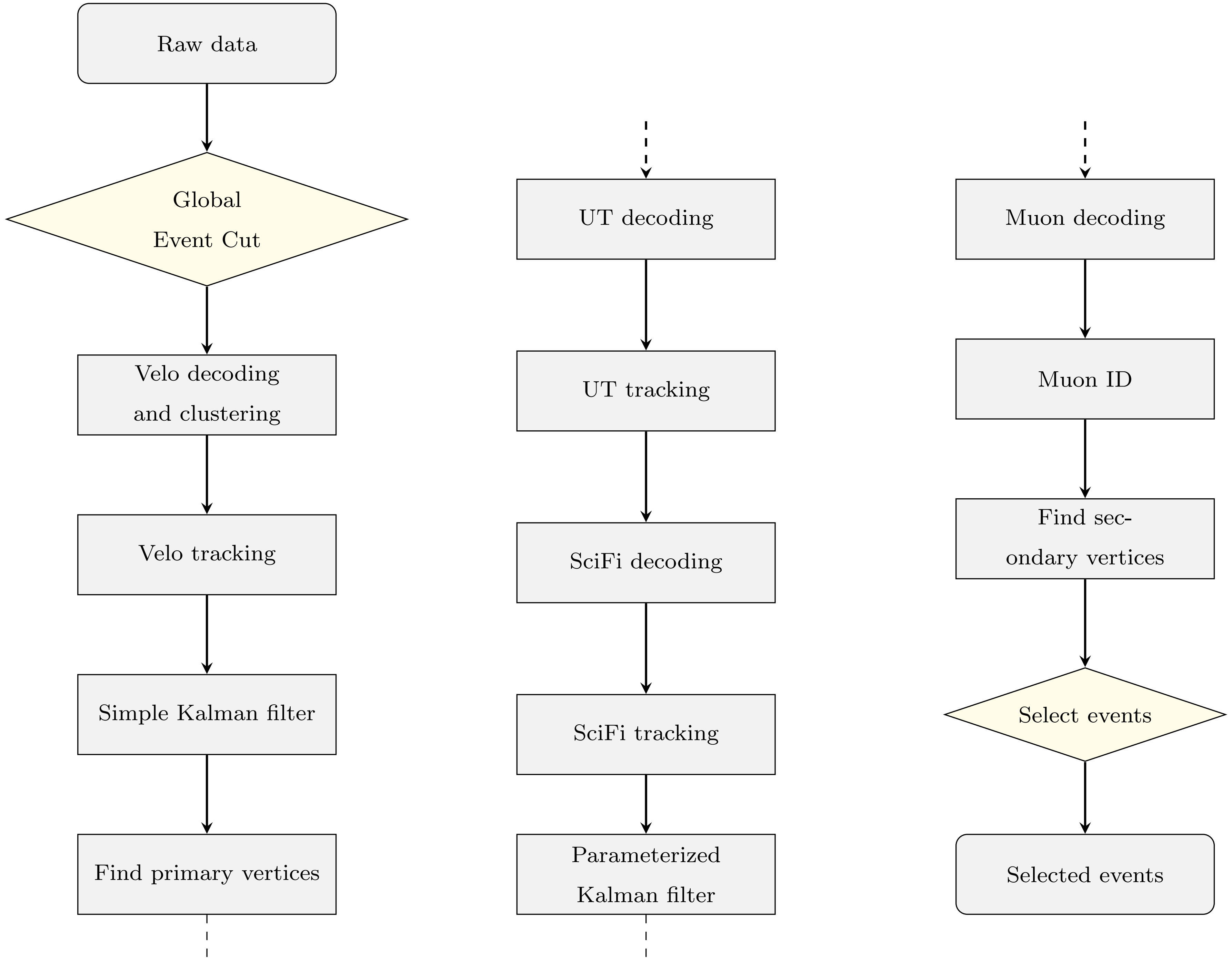
Architecture of a GPU based trigger @ 30 MHz





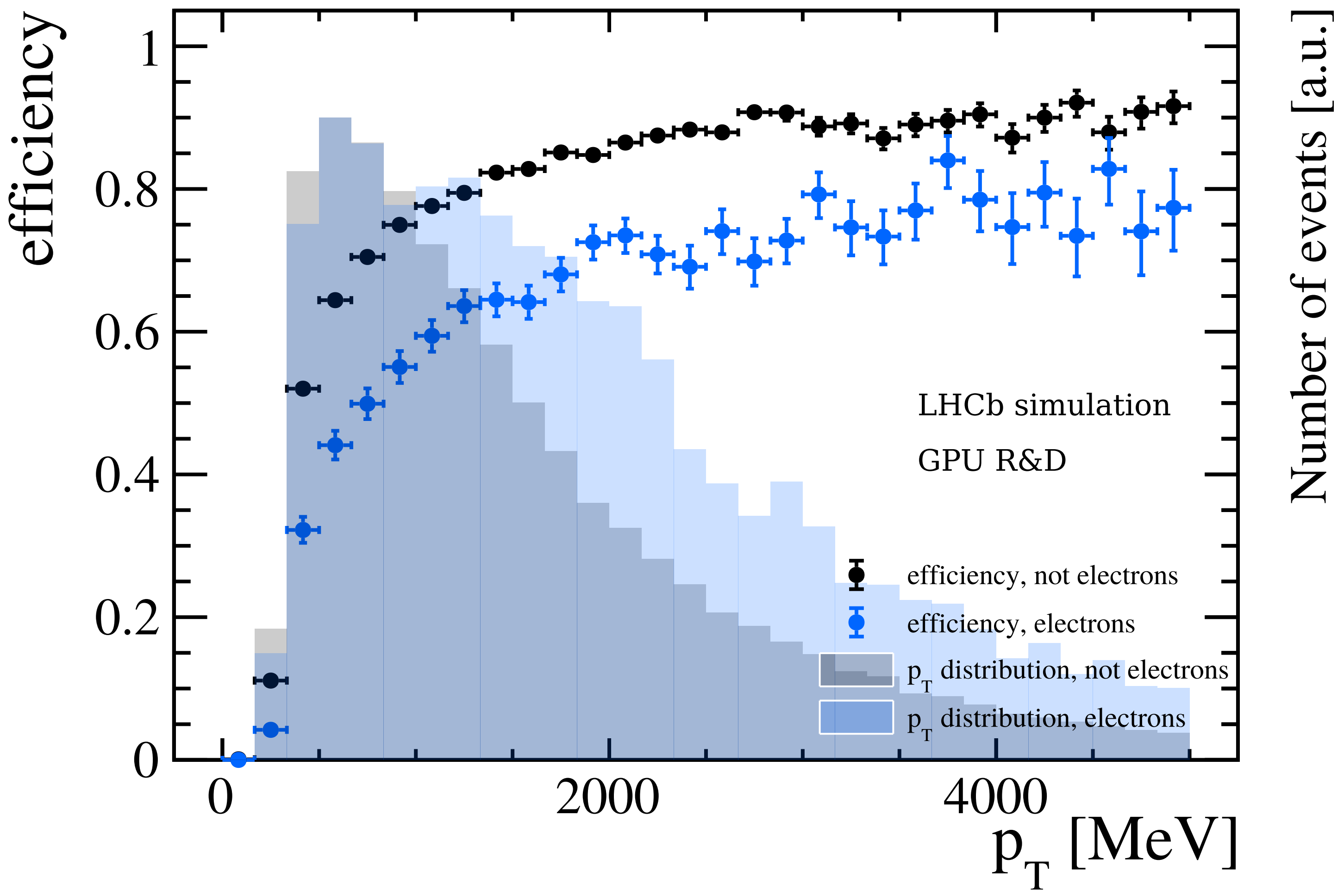
Exploit empty slots on the event building servers — opportunistic but efficient
Each GPU eats 6 GB/s — first integration tests look fine for I/O, final ongoing

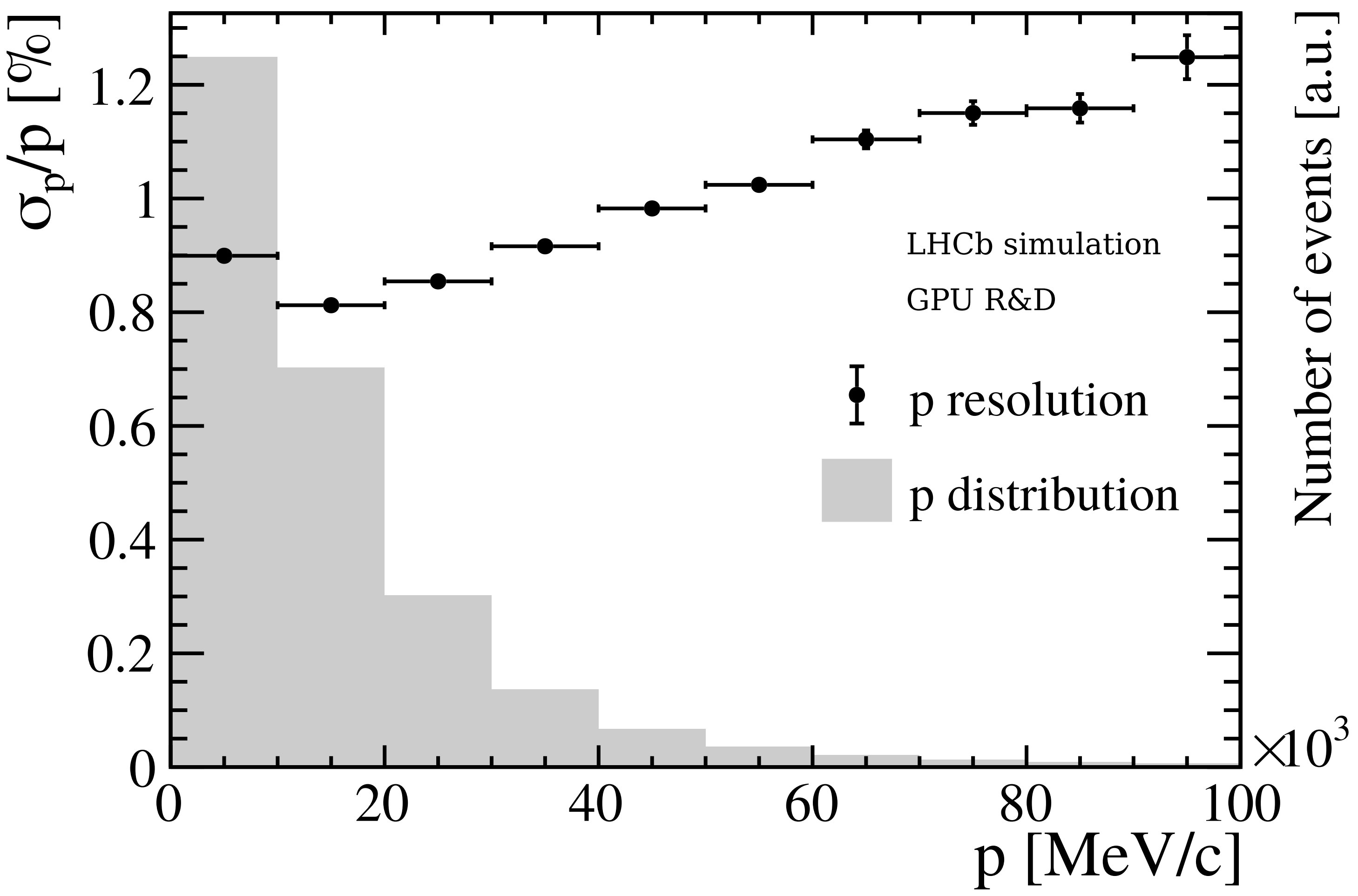
Basic principles of the GPU reconstruction...



Are really the same as multithreaded x86. Optimal degree of paralellism/branching is different, but plain local data is key!

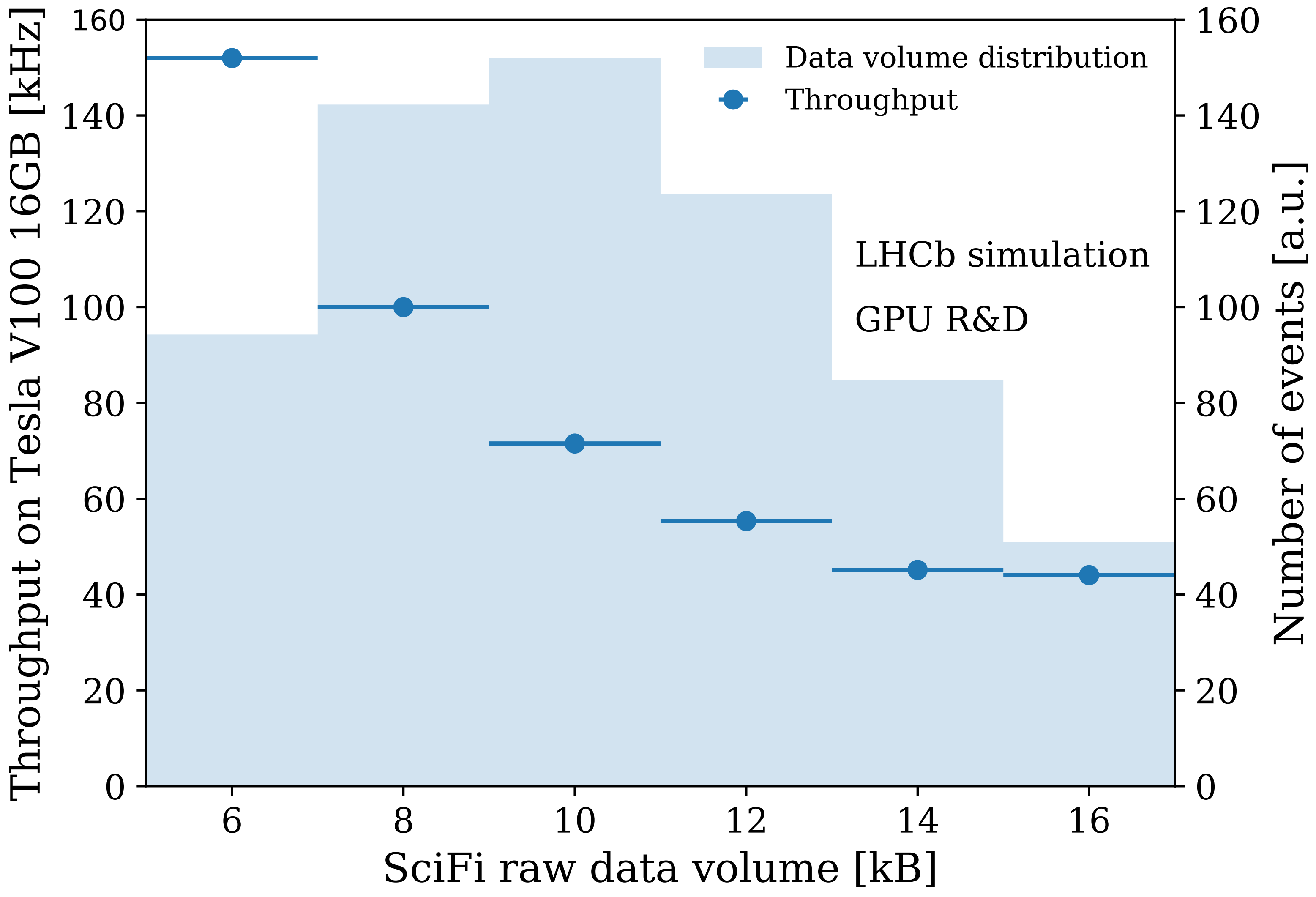
Physics performance

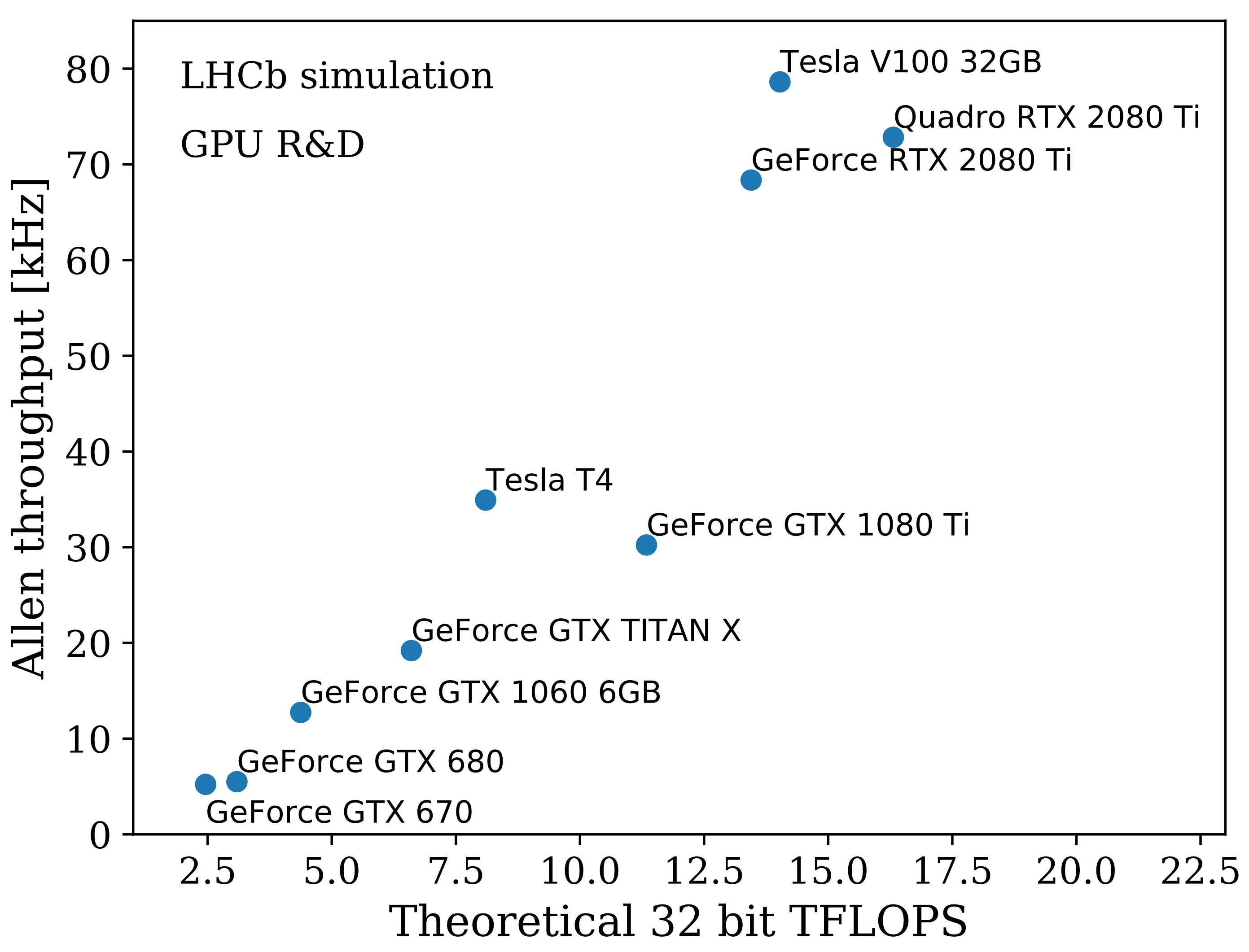




As good as x86 baseline (no approved plots for the baseline...)

GPU throughput scaling

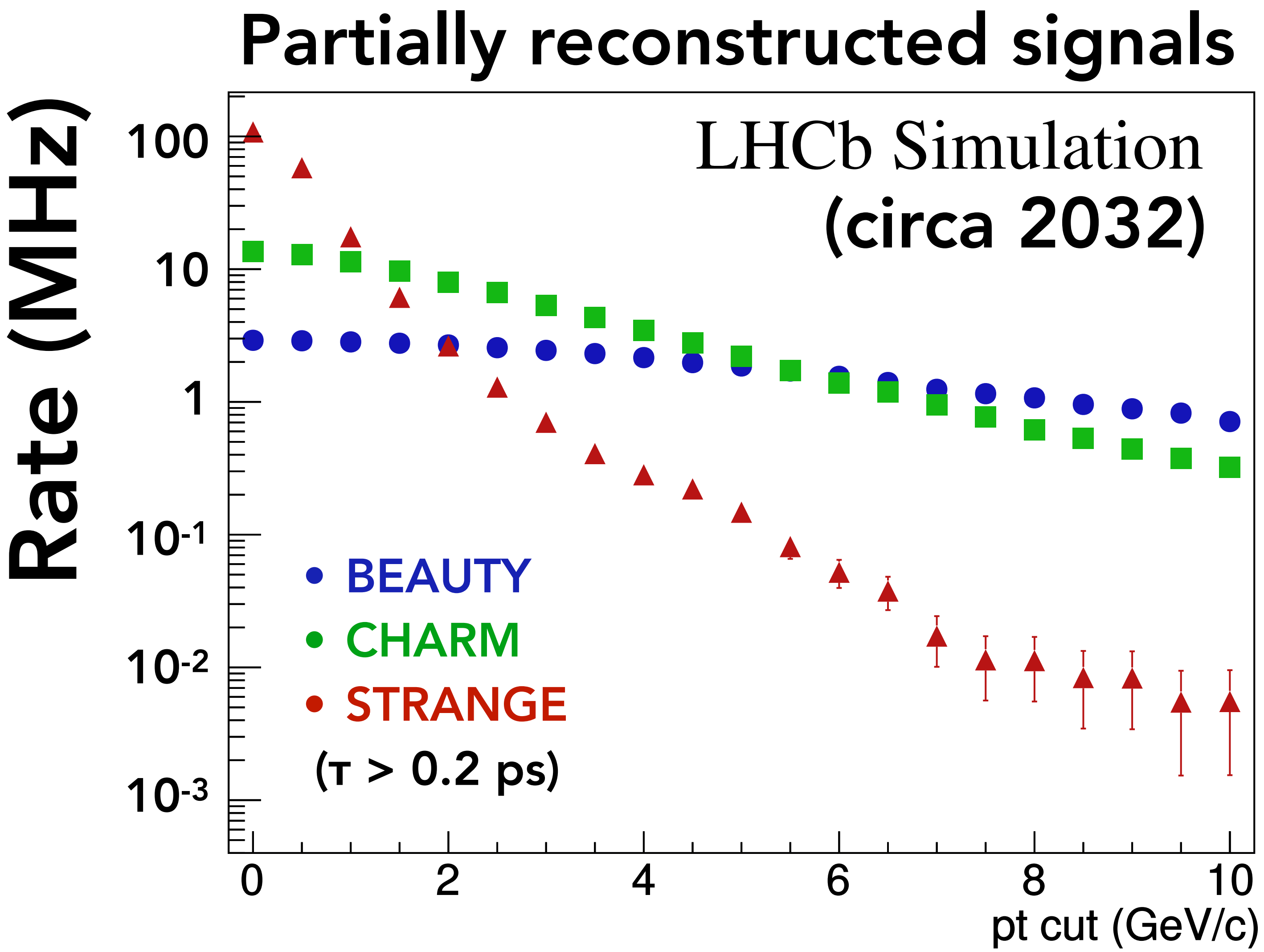




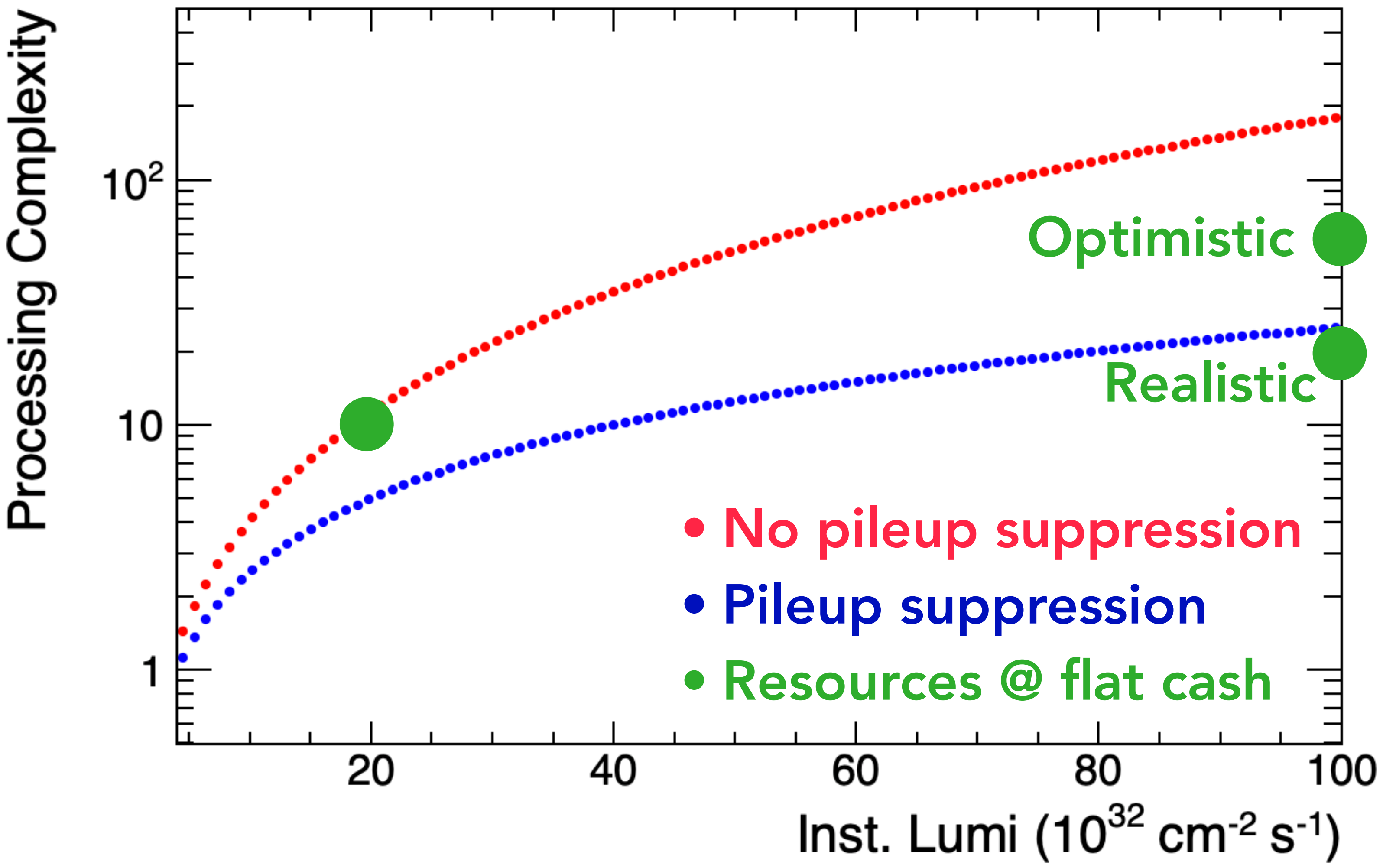
Linear scaling of throughput vs. occupancy, and throughput vs. the theoretical TFLOPS of each card. Optimal use of hardware!

Looking towards the future

Looking beyond to a potential second LHCb upgrade



Fine print: this plot assumes that processing complexity goes linearly with detector occupancy, which is in itself an optimistic assumption before we even get to the pileup suppression part!



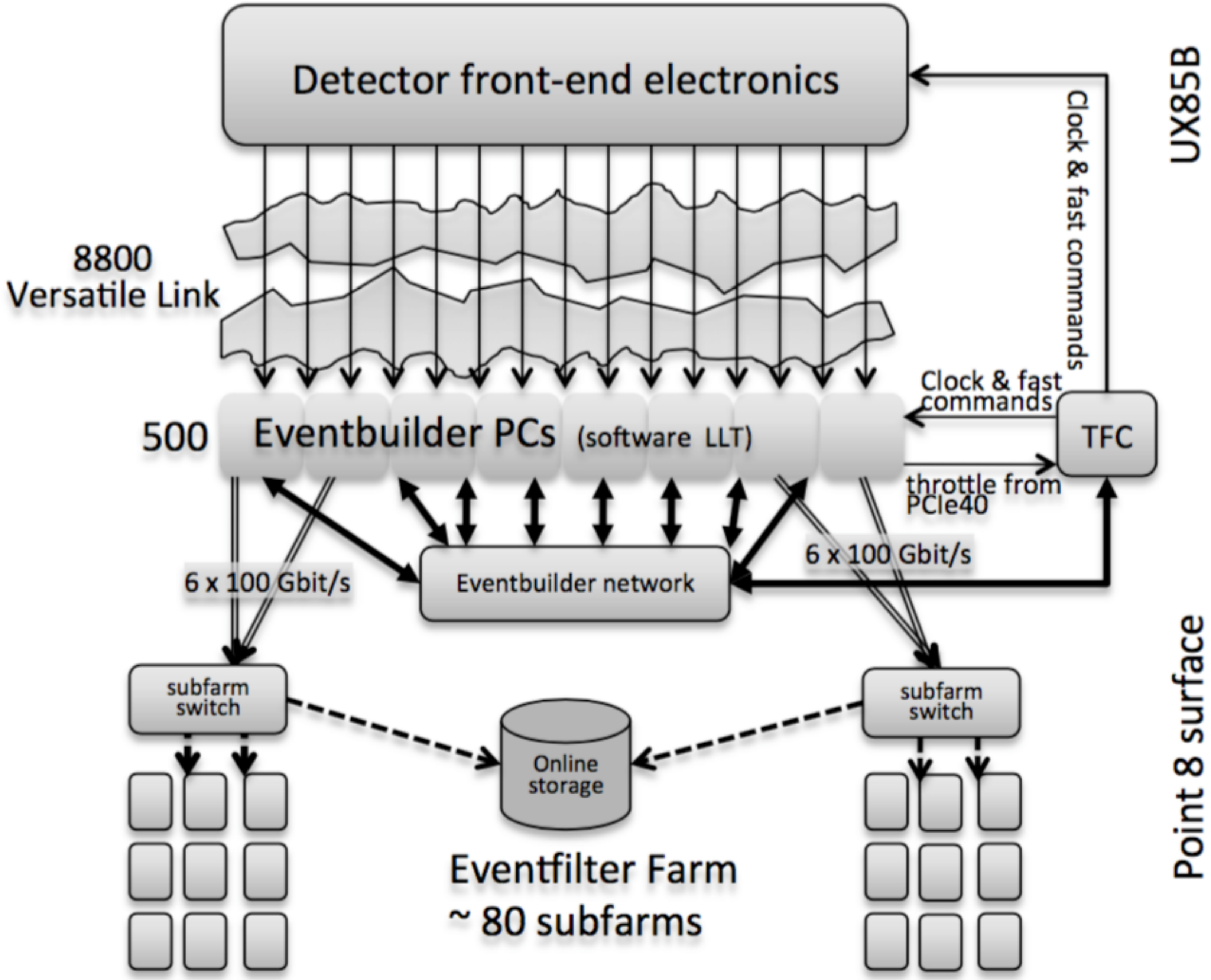
How to suppress pileup with O(60) pp collisions per bunch crossing?

Maintaining the flexibility of our processing will be crucial

GBT link : 4.8 Gb/s Upgrade I
Assume evolution to 10 Gb/s for HL-LHC using aggressive error handling : missing factor 5 compared to data rate growth.

Event-building : current network is 500 servers with 100 Gb/s links. 200 Gb/s readily available, keep an eye on price/performance scaling beyond this?

Farm : carry out R&D in next years on optimal use of hybrid architectures (GPU/CPU/FPGA), remain flexible

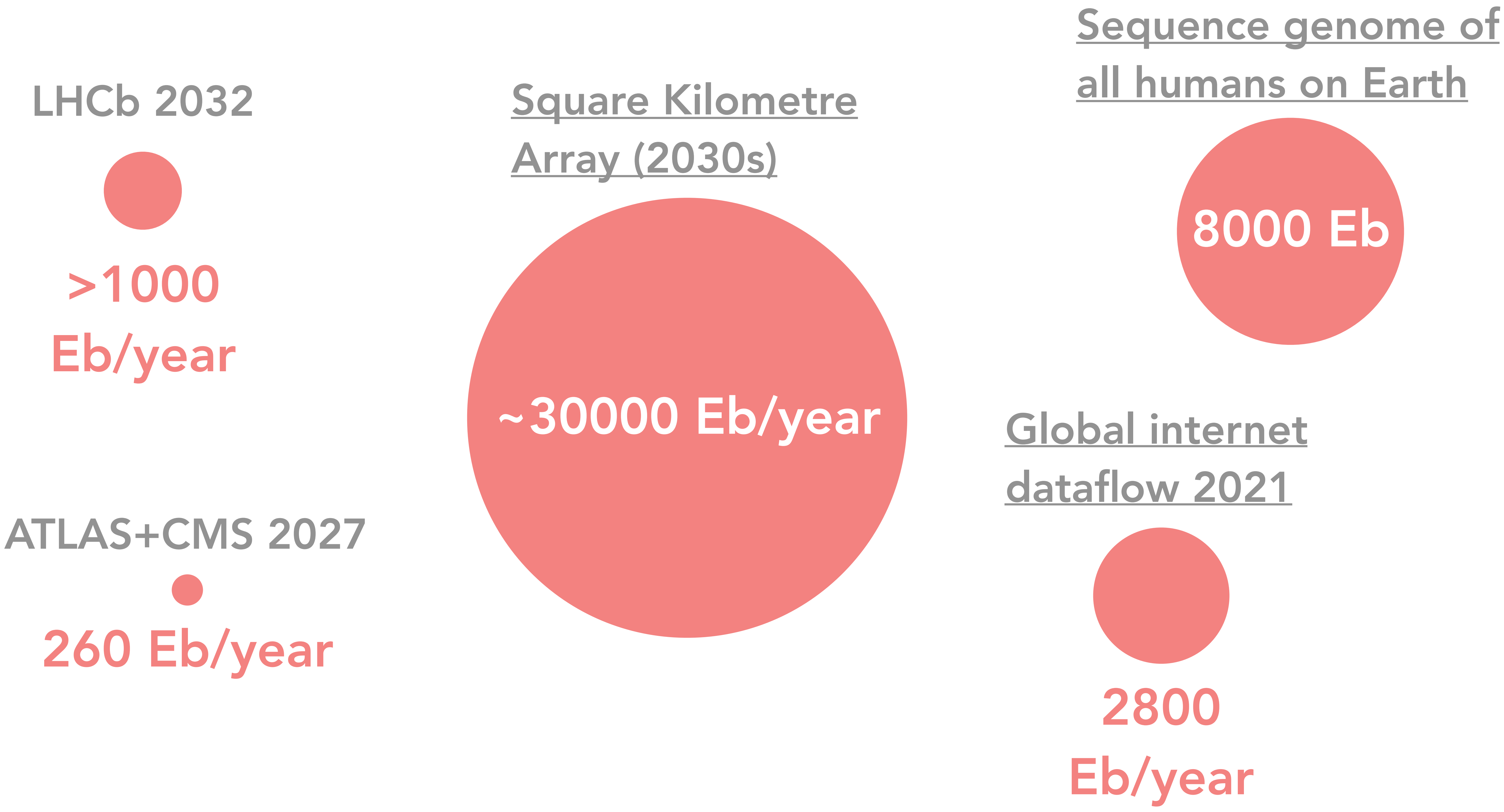


We now have two viable HLT1 models, on x86 and on GPU, already for Run 3! Ability to exploit hybrid architectures crucial to maximize physics/Euro in the long term.

Personal observations on working in a hybrid world

1. The computing landscape is moving towards hybrid architectures. We are developing the skills to move with it!
2. If the basic principles of high throughput software 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. A variable latency trigger is a home for API designers, physicists and selection authors, throughput experts, algorithm designers… it's a very diverse community and personal architecture preferences are real. It is more work to keep a diverse community coherent, but it's worth it.

Conclusions and final thoughts



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

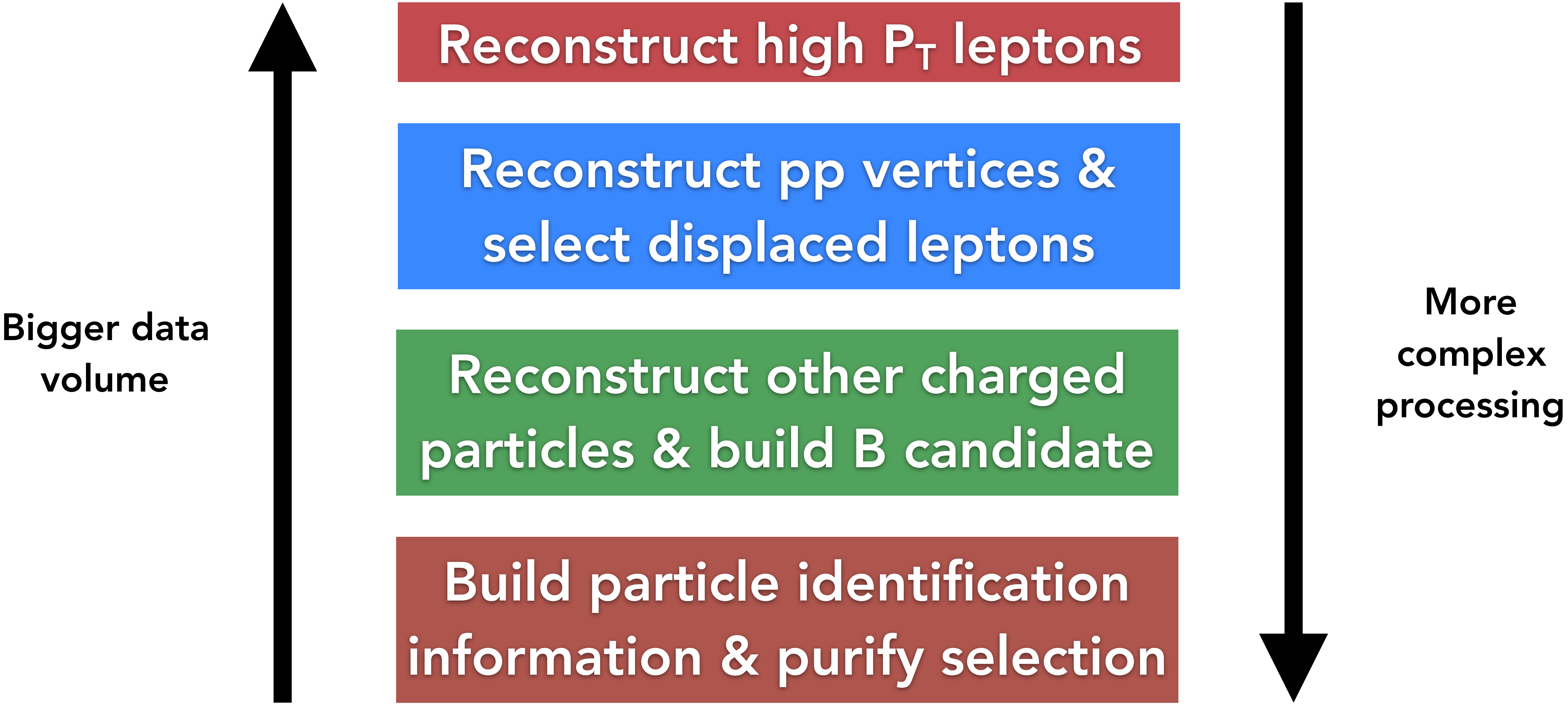
Particle identification

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

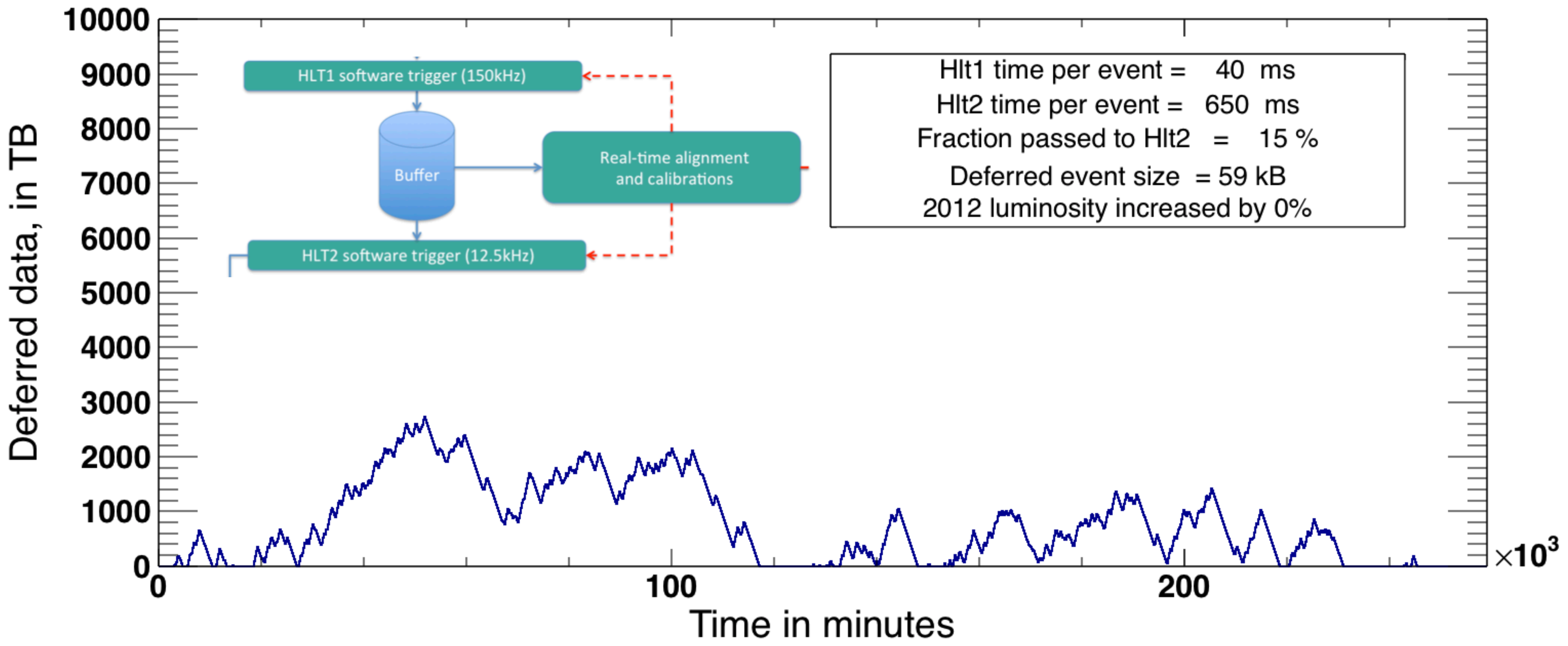
Data driven efficiency calibration key to precision physics

What is a cascade buffer?



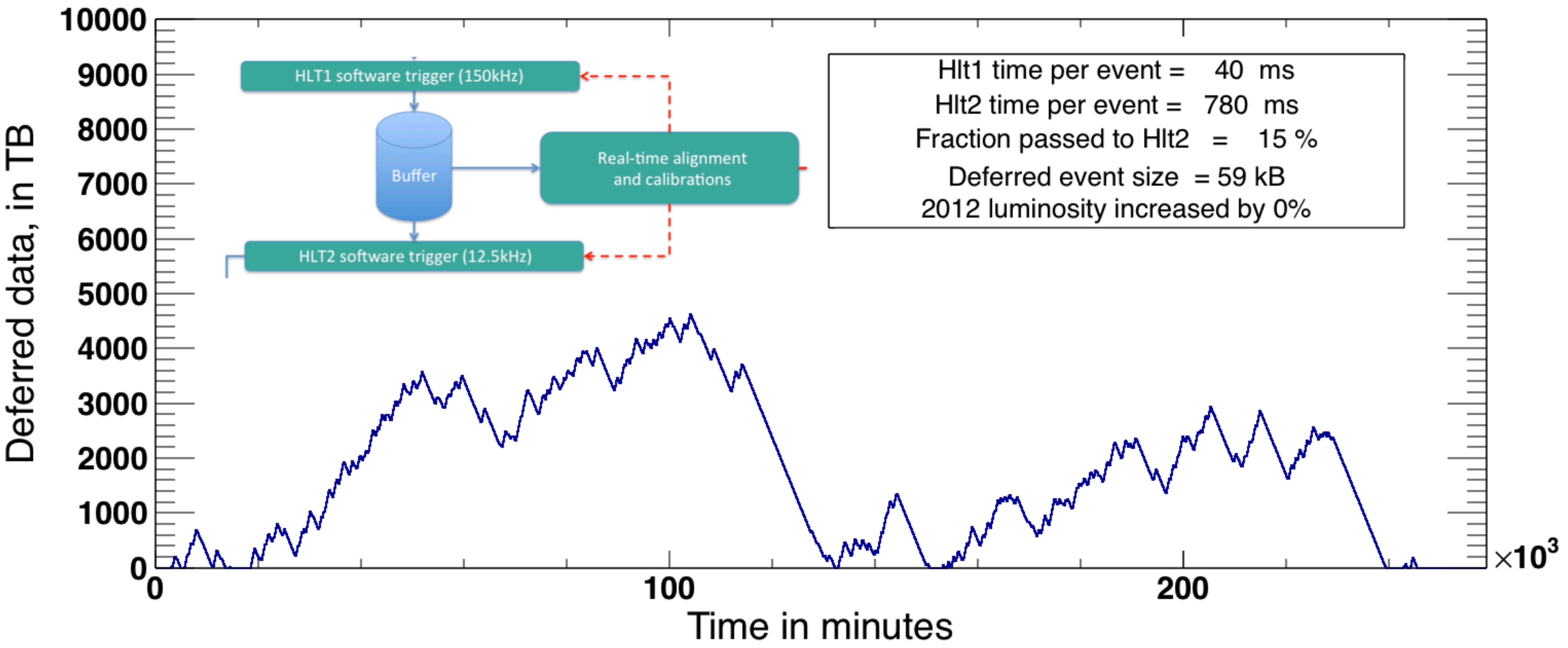
A staged data reduction using increasingly complex algorithms

Optimization of the Run 2 LHCb cascade buffer



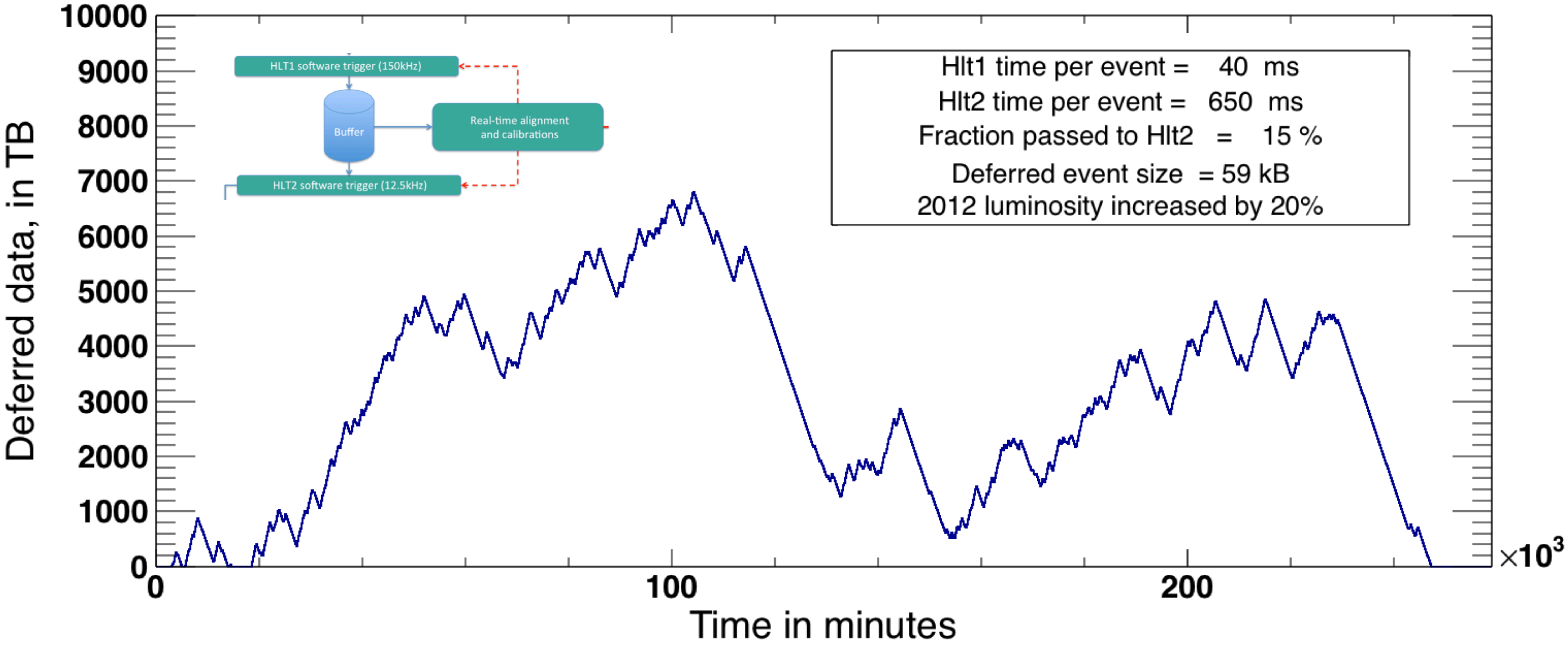
Use Run I LHC fill structure to simulate disk buffer usage

Optimization of the Run 2 LHCb cascade buffer



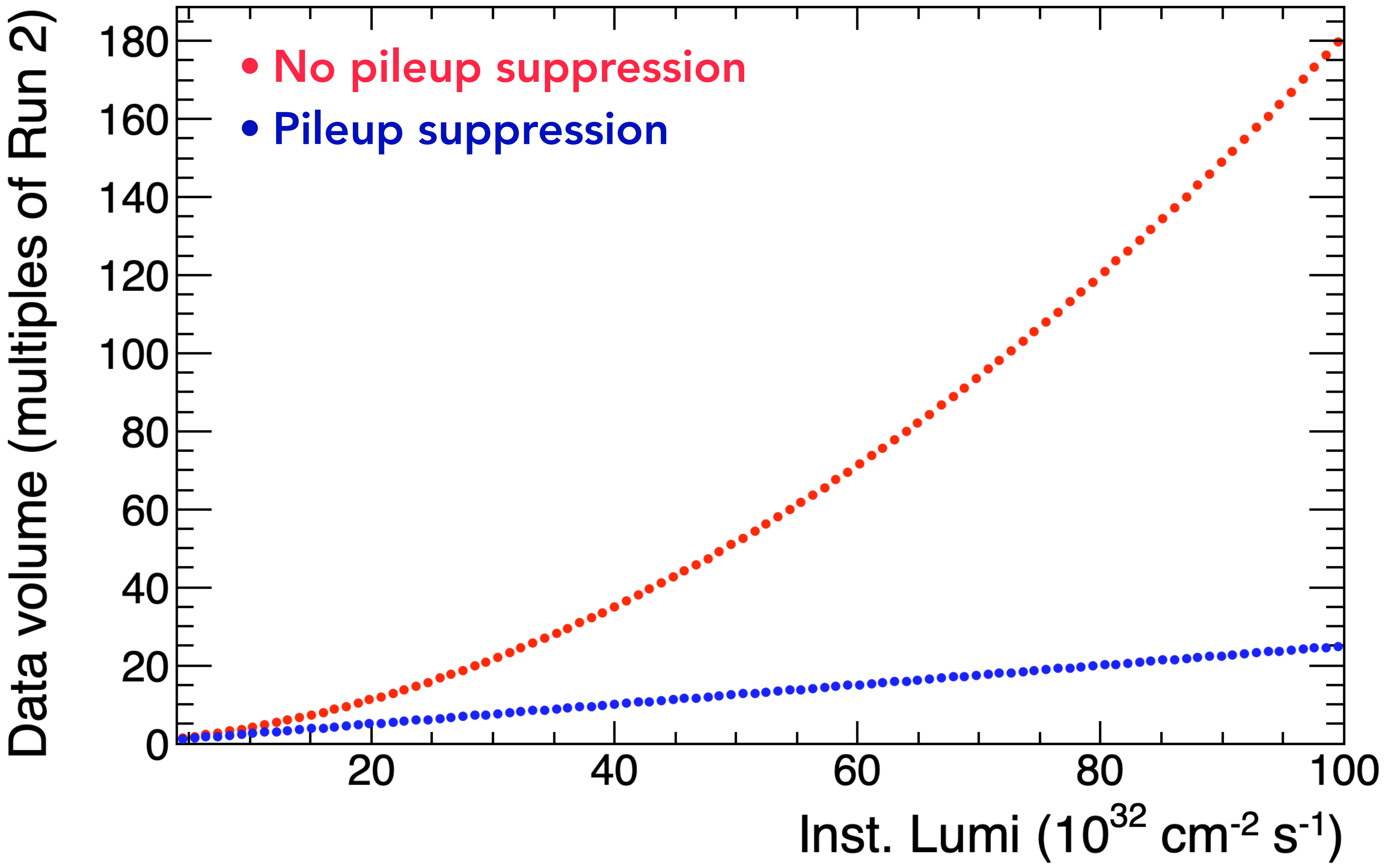
Use simulation to ensure robustness if timing estimates wrong

Optimization of the Run 2 LHCb cascade buffer



Use simulation to ensure robustness if LHC overperformed

And what about data volumes?



Data volume increases quadratically even with 0 background.
Select pp collisions, not bunch crossings, in real time!