AN INTRODUCTION TO TRIGGERING IN HEP

Don’t Miss the Good Stuff

Some of the most ingenious engineering in experimental HEP has nothing to do with the detector — it's about deciding, in microseconds, whether what just happened is worth keeping.

HOOK & MOTIVATION

The rare vs. the mundane

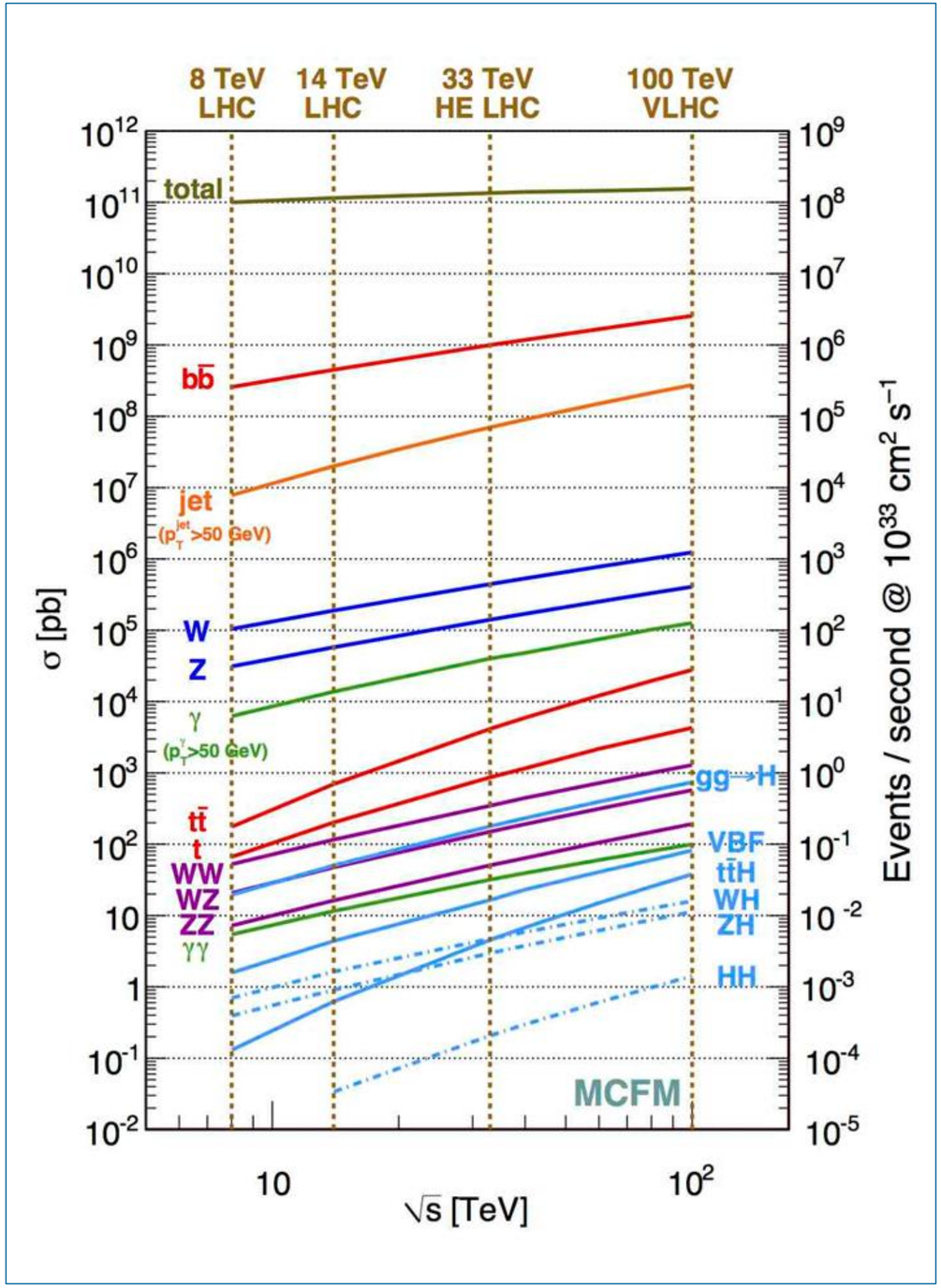
Proton bunches cross **40 MHz** — forty million times per second

Each crossing produces ~40 simultaneous pp collisions (**pileup**)

The vast majority are soft QCD — elastic scattering, minimum bias — uninteresting

Processes we care about (Higgs, SUSY, rare B decays…) are **orders of magnitude rarer**

> *"The LHC is like a firework factory where 99.9999999% of what comes out is confetti. The trigger is your eye trained to spot the one rocket."*



THE STORAGE PROBLEM

Why you can’t just record everything

EVENT SIZE

1–2

MB

per ATLAS/CMS event, after zero-suppression

RAW THROUGHPUT

60

TB/s

40 MHz × 1.5 MB — you'd fill all world disk in hours

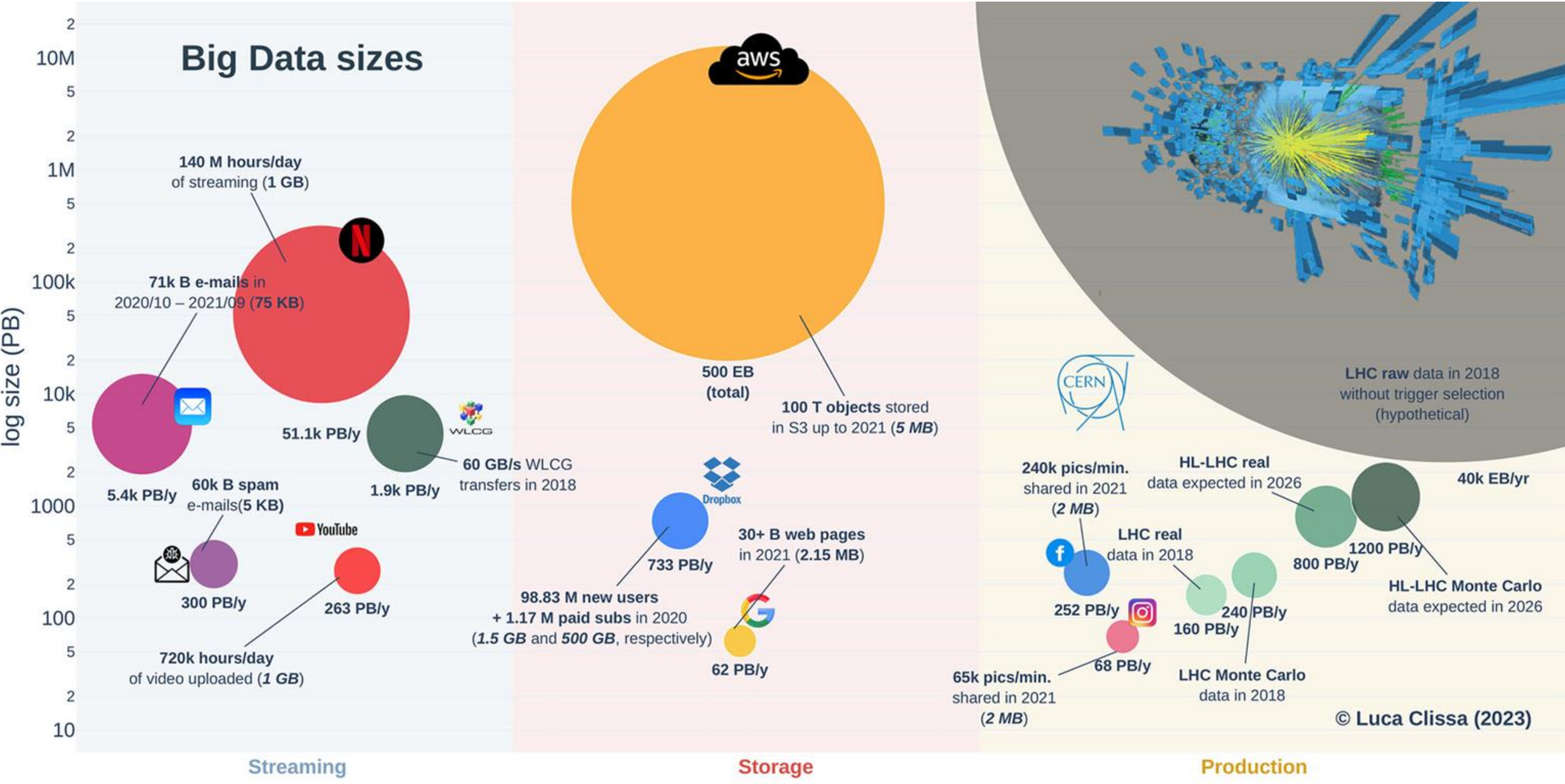
STORAGE BUDGET

1–2

GB/s

Required reduction factor: ~10^5

This is not solvable with more money or faster disks — it is a fundamental constraint.


Big Data sizes
log size (PB)
2
10M
5
2
1M
5
2
100k
5
2
10k
5
2
1000
5
2
100
5
2
10
140 M hours/day
of streaming (1 GB)
71k B e-mails in
2020/10 – 2021/09 (75 KB)
5.4k PB/y
60k B spam
e-mails(5 KB)
300 PB/y
720k hours/day
of video uploaded (1 GB)
263 PB/y
51.1k PB/y
1.9k PB/y
60 GB/s WLCG
transfers in 2018
500 EB
(total)
100 T objects stored
in S3 up to 2021 (5 MB)
733 PB/y
98.83 M new users
+ 1.17 M paid subs in 2020
(1.5 GB and 500 GB, respectively)
30+ B web pages
in 2021 (2.15 MB)
62 PB/y
LHC raw data in 2018
without trigger selection
(hypothetical)
240k pics/min.
shared in 2021
(2 MB)
252 PB/y
65k pics/min.
shared in 2021
(2 MB)
68 PB/y
LHC real
data in 2018
160 PB/y
LHC Monte Carlo
data in 2018
240 PB/y
HL-LHC real
data expected in 2026
800 PB/y
40k EB/yr
1200 PB/y
HL-LHC Monte Carlo
data expected in 2026
© Luca Clissa (2023)
Streaming
Storage
Production
data sources

THE TRIGGER

Definition and goals

A trigger is a **real-time decision system** that selects events to record.

`Trigger = real-time classification under extreme resource constraints`

Fast — keep up with the beam crossing rate

Selective — high background rejection

Efficient — retain signal events

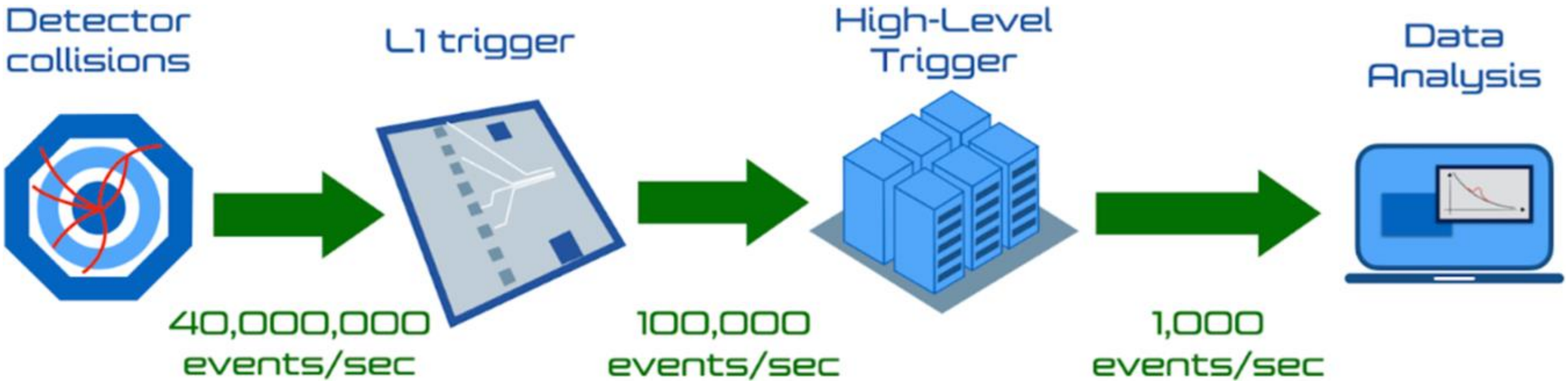
Flexible — configurable for many simultaneous physics goals

TWO DISTINCT TECHNICAL CHALLENGES

CHALLENGE	REQUIREMENT
Rate reduction	`40 MHz → O(kHz)`
Latency	Decision must fit within front-end readout buffer

ARCHITECTURE

Divide and conquer: trigger levels



The problem is too hard to solve in one step — split into **levels**, each doing what it does best:

L1: coarse, fast, custom silicon

HLT: full reconstruction, commodity CPUs

LEVEL-1 TRIGGER

The hardware gatekeeper

L1 must decide in **fixed latency** — CMS: **3.8 µs** from collision to decision

Data is held in **pipeline buffers** in detector front-end electronics during this time

Uses **coarse, fast** information: calorimeter clusters (e, γ, jets, MET) + muon track stubs

Implemented in **custom FPGAs and ASICs** — no commercial off-the-shelf hardware

Output: list of "trigger objects" — e.g. e^{-} candidate, E_T > 30 GeV at (η, φ)

L1 accept rate: **~100 kHz** (ATLAS/CMS)

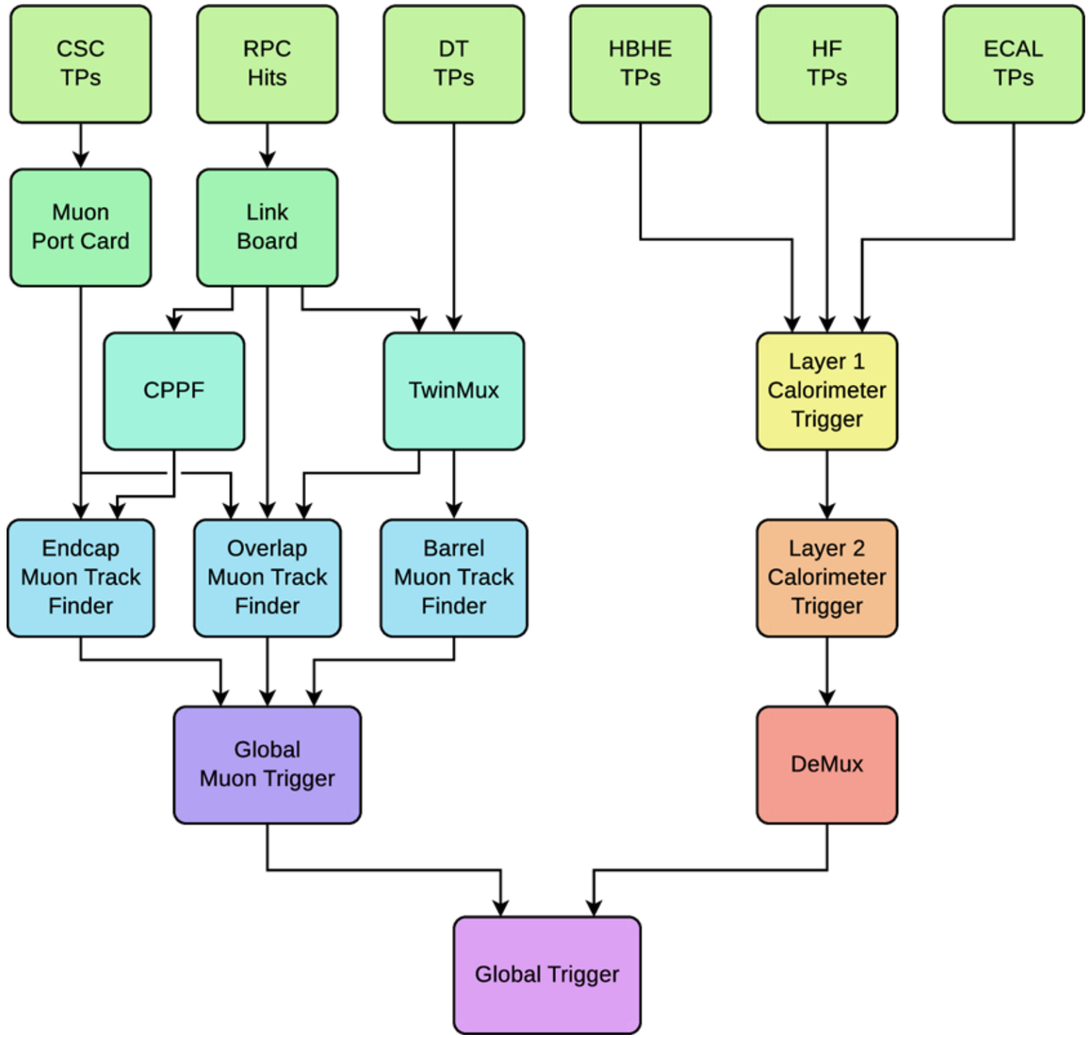
CMS L1 LATENCY BUDGET

3.8 µs

From collision to L1 accept signal

> *"L1 is like a bouncer at a club who has 4 microseconds to look at your shoes and decide if you get in. No time to check your ID."*

How does L1 actually work?



EXAMPLE MENU CONDITIONS (CMS-STYLE)

SEED NAME	CONDITION
SingleEle32	Single electron, $E_T > 32$ GeV
DoubleMu7	Two muons, each $p_T > 7$ GeV
MET200	Missing $E_T > 200$ GeV

Implemented in **custom FPGAs** — logic evaluated in nanoseconds

All ~500 seeds evaluated **simultaneously** in parallel hardware

Any condition satisfied → L1 ACCEPT → readout all subdetectors

TRIGGER MENU

The physics program in a lookup table

The trigger menu **is the experiment's physics programme** encoded in real-time logic.

~O(500) trigger seeds / paths in current LHC experiments

Each path has a **prescale** — keep 1-in-N events to manage rates

What you don't trigger on, you **lose forever**

The menu must be agreed before data-taking — a physics/political process

TRIGGER MENU

The physics program in a lookup table

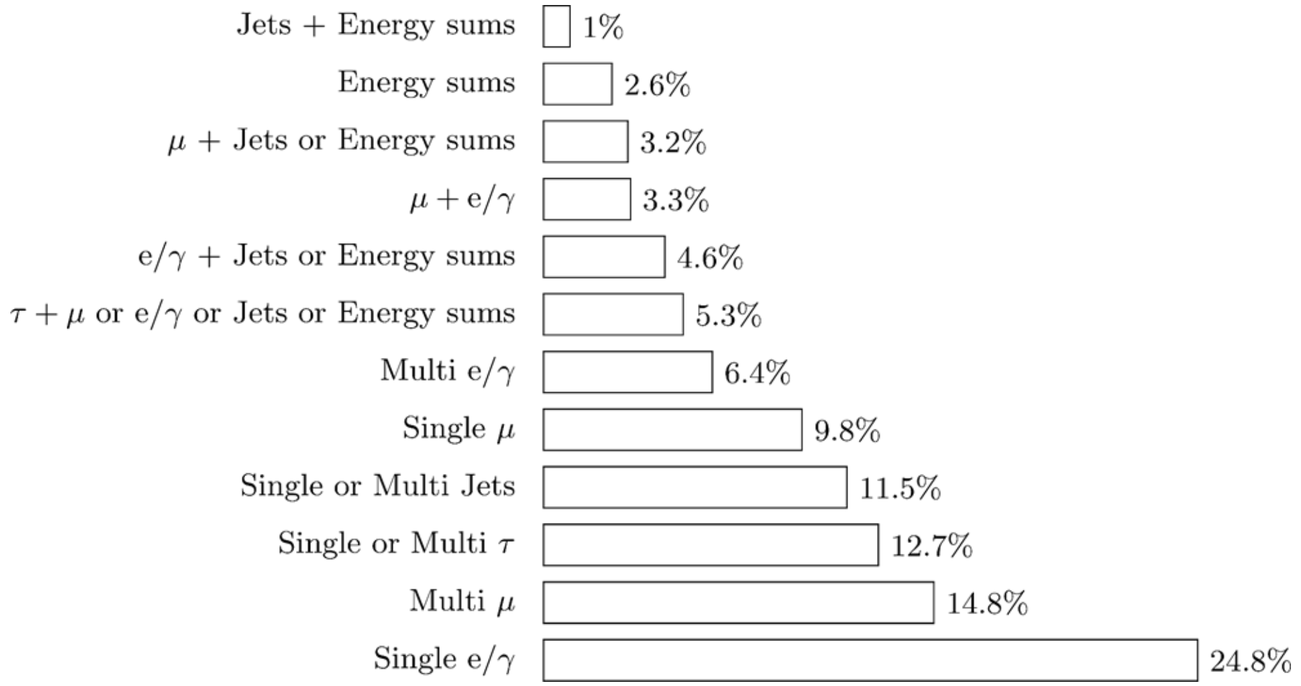
The trigger menu **is the experiment's physics programme** encoded in real-time logic.

~O(500) trigger seeds / paths in current LHC experiments

Each path has a **prescale** — keep 1-in-N events to manage rates

What you don't trigger on, you **lose forever**

The menu must be agreed before data-taking — a physics/political process



Software reconstruction at 100 kHz input

HLT runs on a CPU/GPU farm: CMS `~30,000 cores`, ATLAS similar scale

At 100 kHz input → `~O(450 ms)` per event on average

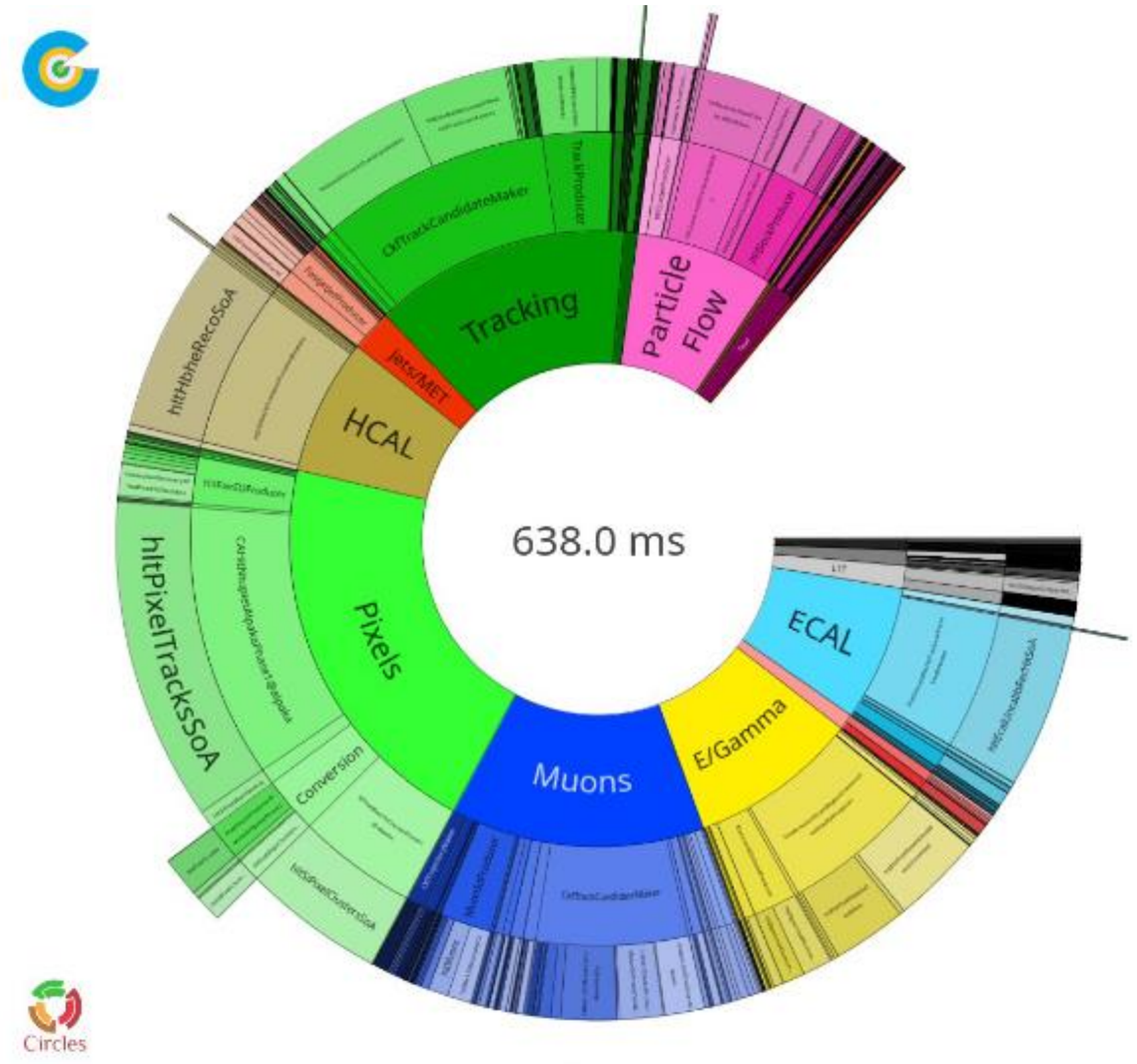
Runs a **partial version of offline reconstruction** : tracking, clustering, jet finding, b-tagging

Using L1 inputs as seeds for the reconstruction and defining ROIs.

Input 100 kHz

↓

Storage 1–2 kHz



Software reconstruction at 100 kHz input

HLT runs on a CPU/GPU farm: CMS `~30,000 cores`, ATLAS similar scale

At 100 kHz input → `~O(450 ms)` per event on average

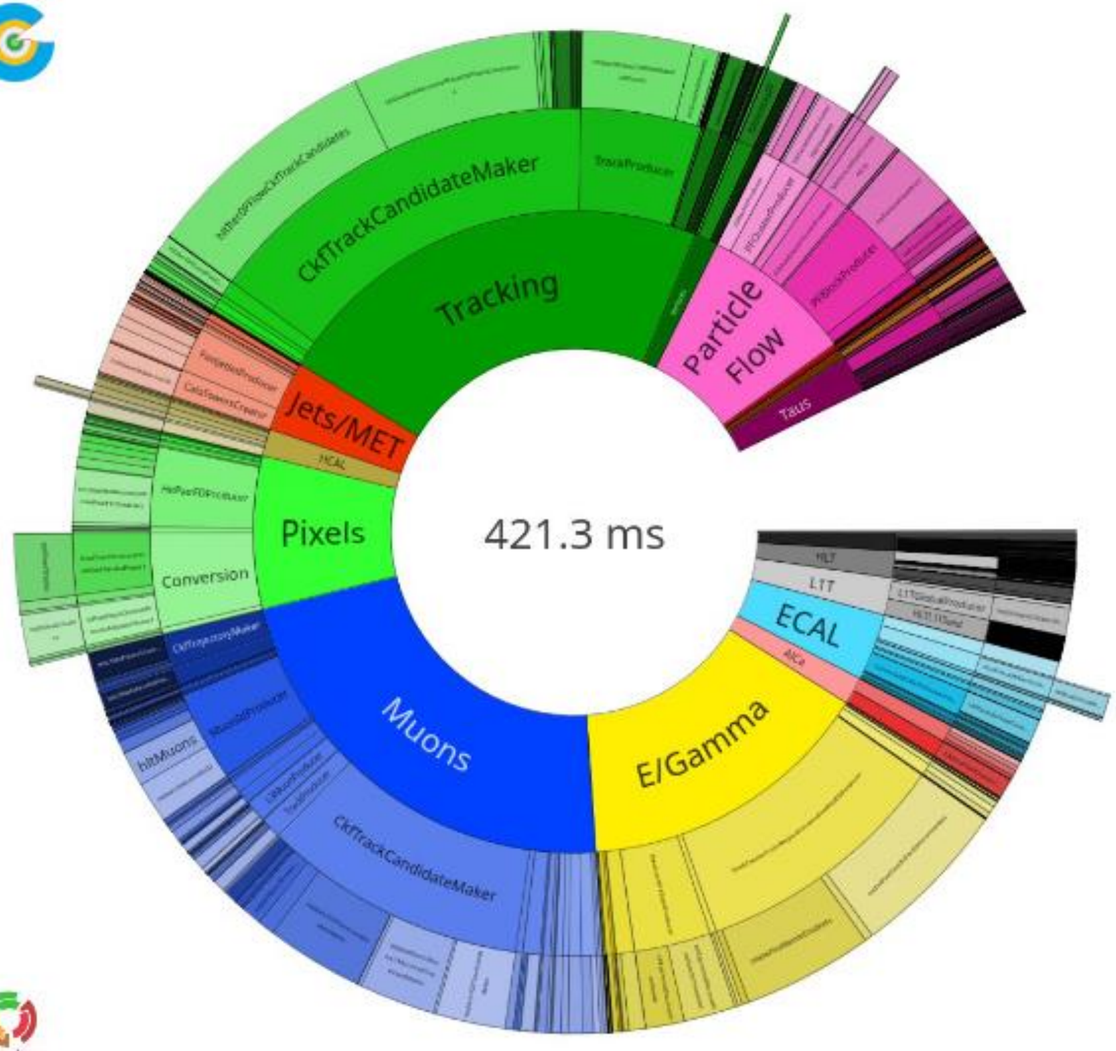
Runs a **partial version of offline reconstruction** : tracking, clustering, jet finding, b-tagging

Using L1 inputs as seeds for the reconstruction and defining ROIs.

Input 100 kHz

↓

Storage 1–2 kHz



Software reconstruction at 100 kHz input

HLT runs on a CPU/GPU farm: CMS ~30,000 cores, ATLAS similar scale

At 100 kHz input → ~O(450 ms) per event on average

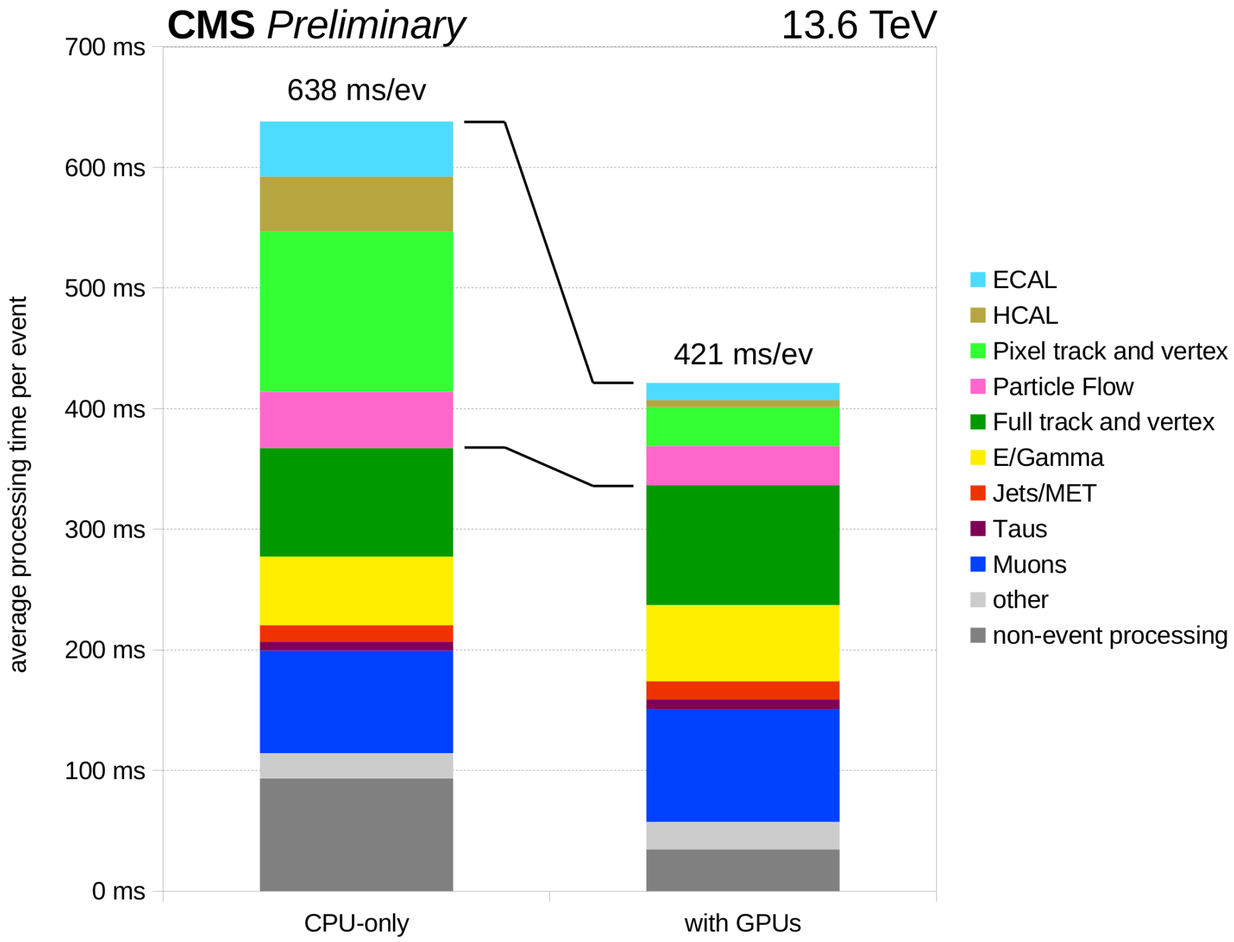
Runs a **partial version of offline reconstruction** : tracking, clustering, jet finding, b-tagging

Using L1 inputs as seeds for the reconstruction and defining ROIs.

Input 100 kHz

↓

Storage 1–2 kHz



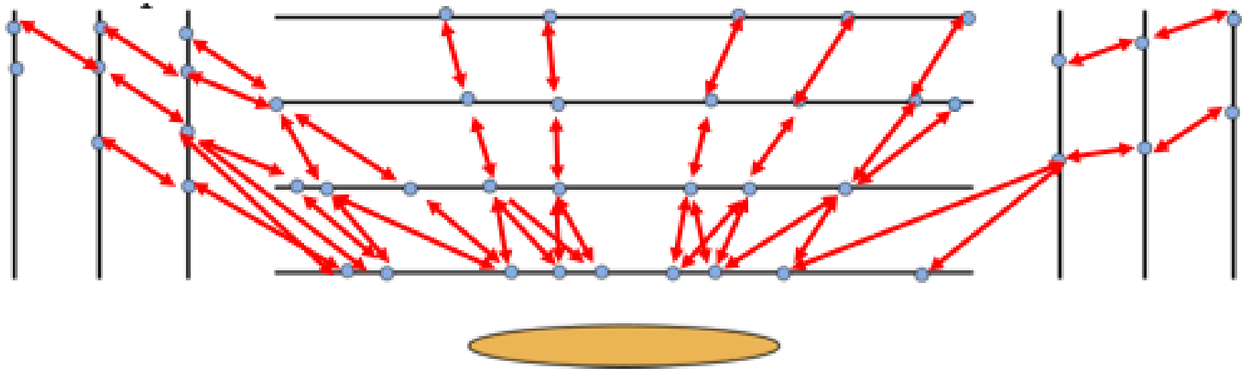
The most expensive step — and the most powerful

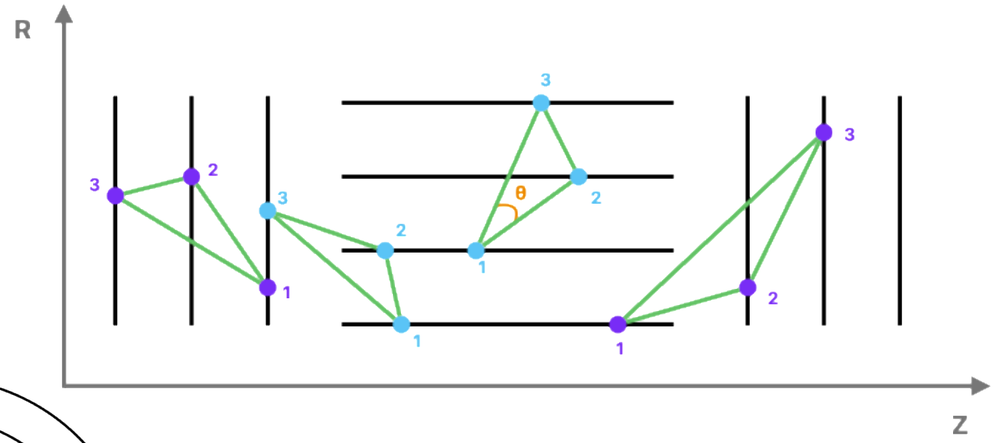
Track reconstruction is a **combinatorial problem** — scales badly with pileup

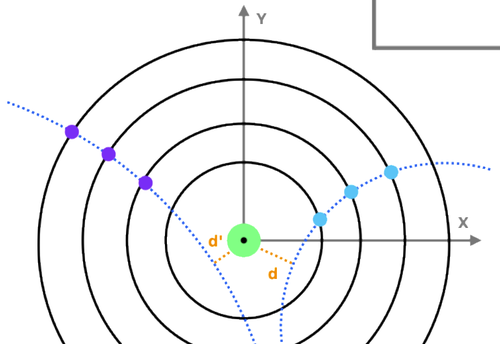
Inner tracker: ~O(10⁸) channels → billions of hits per event at high pileup

HLT uses **fast track finding** : road-based, Hough transform, cellular automaton

Tracking unlocks: **impact parameter** → b-tagging, τ-tagging, displaced vertices







> *"Tracking at the HLT is the difference between knowing an electron exists and knowing where it came from."*

PHYSICS IMPACT

What the trigger does to your measurement

Every trigger introduces **efficiency** (signal kept) and **bias** (shape distortion)

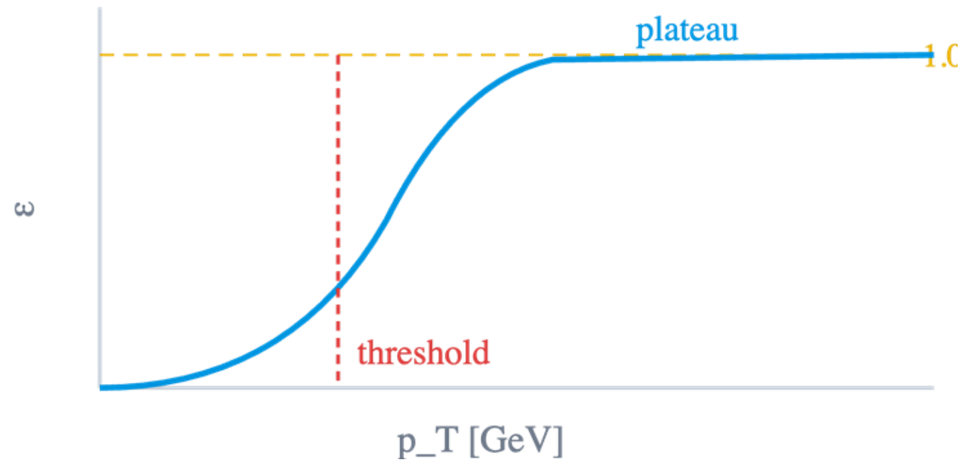
A p_T threxshold creates a sharp **turn-on curve** — must be measured in data

Tag-and-probe method: use clean samples (J/ψ→μμ, Z→ℓℓ) to measure trigger efficiency in situ

Efficiency corrections applied as **per-event weights** in analyses

Trigger bias can distort distributions: e.g. isolation requirement changes jet activity near lepton

SCHEMATIC TURN-ON CURVE

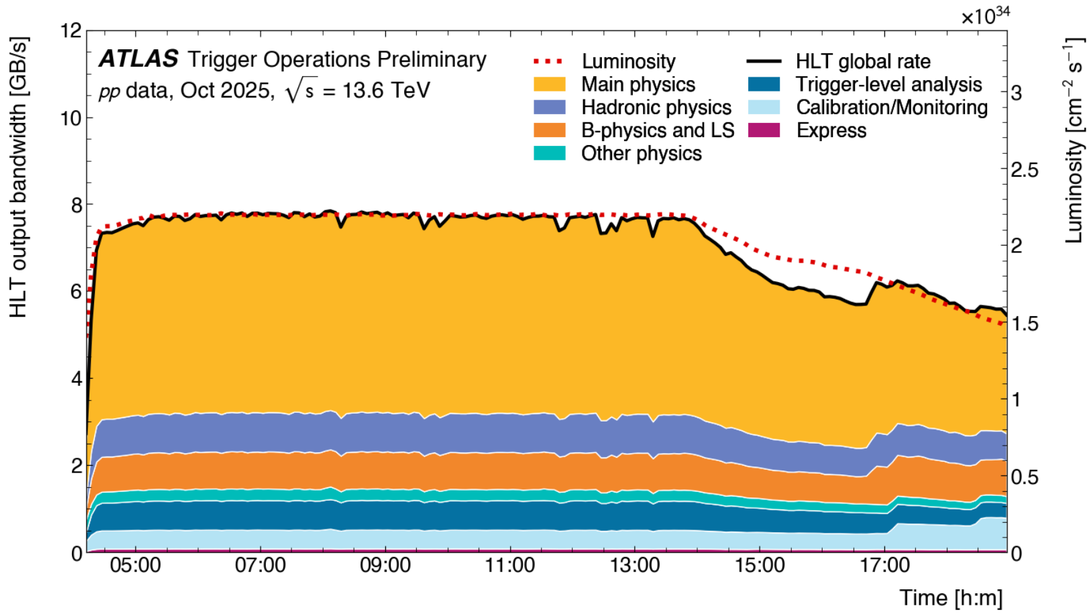


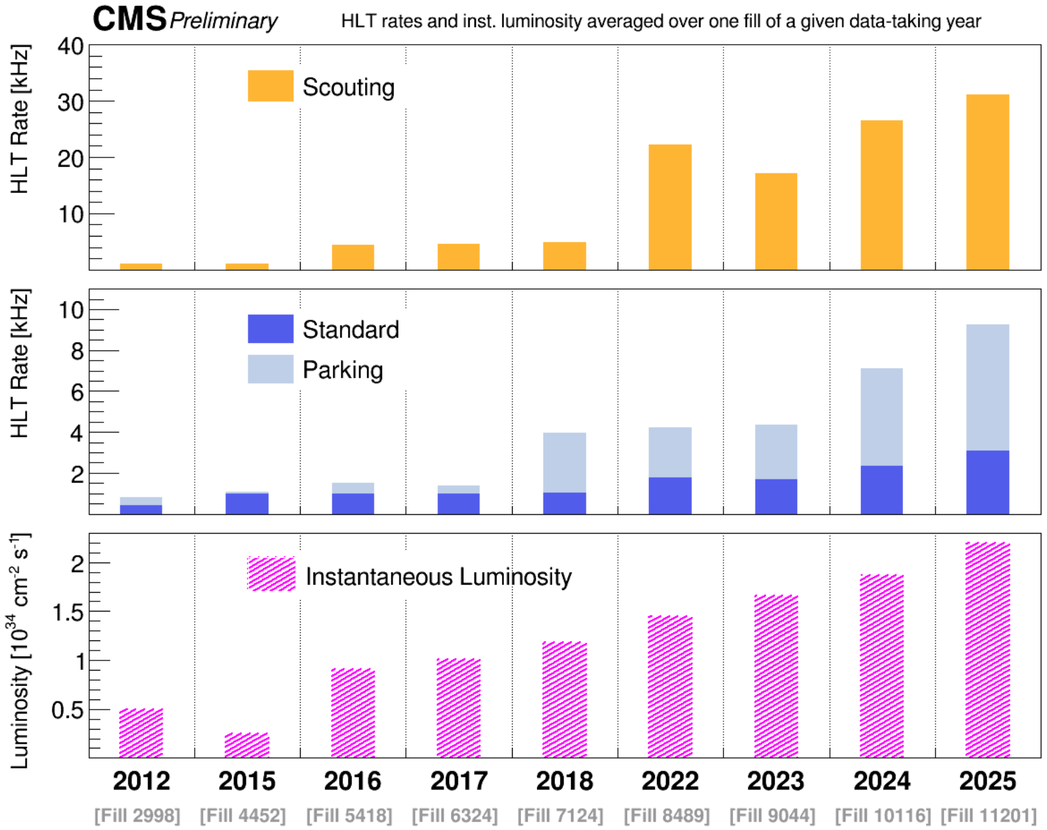
You cannot publish a cross-section measurement without understanding your trigger efficiency. It's not optional.

CASE STUDY 1

General purpose — keeping all options open

Both designed to discover the **unexpected** → broad menu philosophy





CASE STUDY 2

LHCb— a completely different philosophy

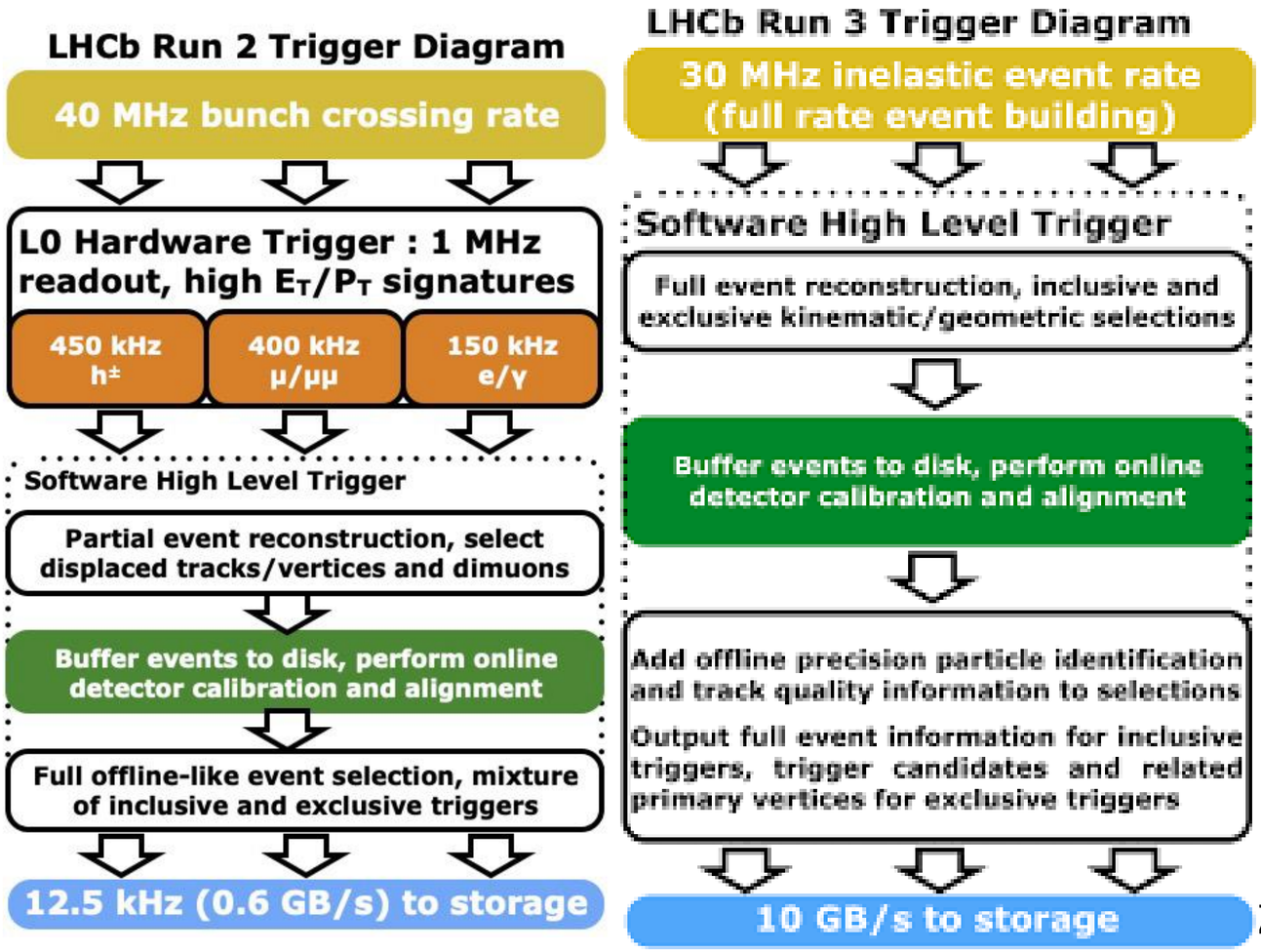
LHCb is a **forward spectrometer** optimized for B-physics and CP violation

Key signature: **displaced vertices** from B decay ($c\tau \sim 500\ \mu m$)

Lower luminosity than ATLAS/CMS → different constraints on trigger design

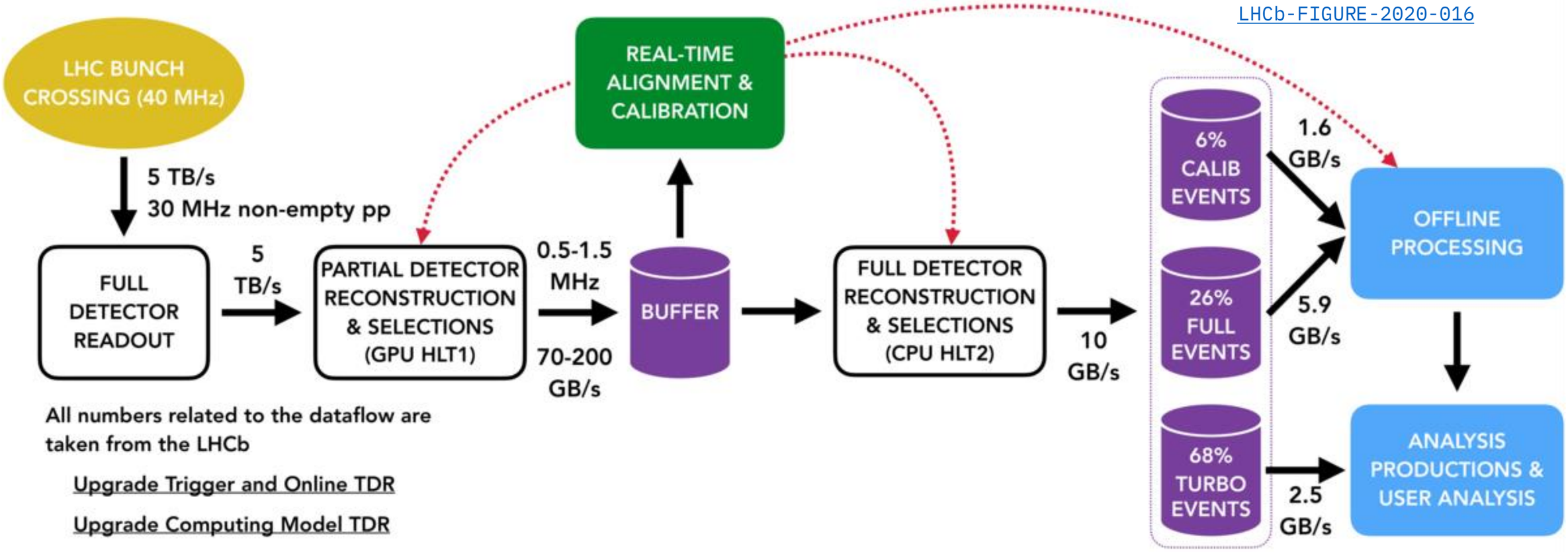
GPU-based real-time reconstruction (**Allen framework**): First LHC experiment with a fully software trigger.

EVOLUTION OF THE LHCB TRIGGER



Heterogeneous computing comes to the trigger

Allen is LHCb's GPU-based first-stage software trigger (HLT1)



Runs on ~500 NVIDIA GPUs (RTX class)

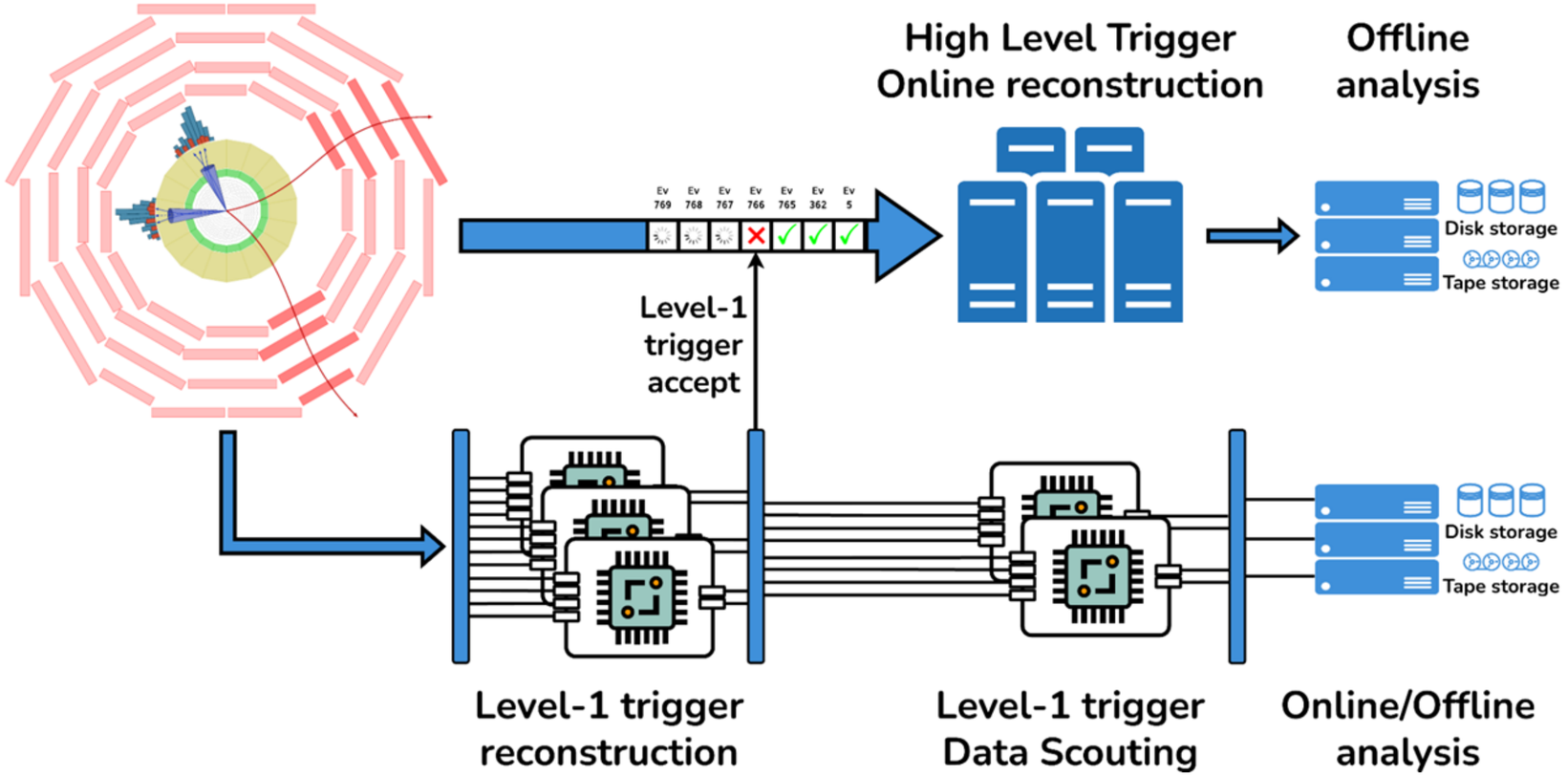
Reconstructs VELO tracks in ~500 µs per event

Finds primary vertices, selects displaced tracks

Open-source framework, influential beyond LHCb

CASE STUDY 3

When you can’t afford the full event



Level-1 Data Scouting rack

For some physics (e.g. dijet resonances at low mass), rates are too high to store full events. Solution: store only trigger-level objects.

~few kB
per event (scouting)

VS

~1.5 MB
per event (full)

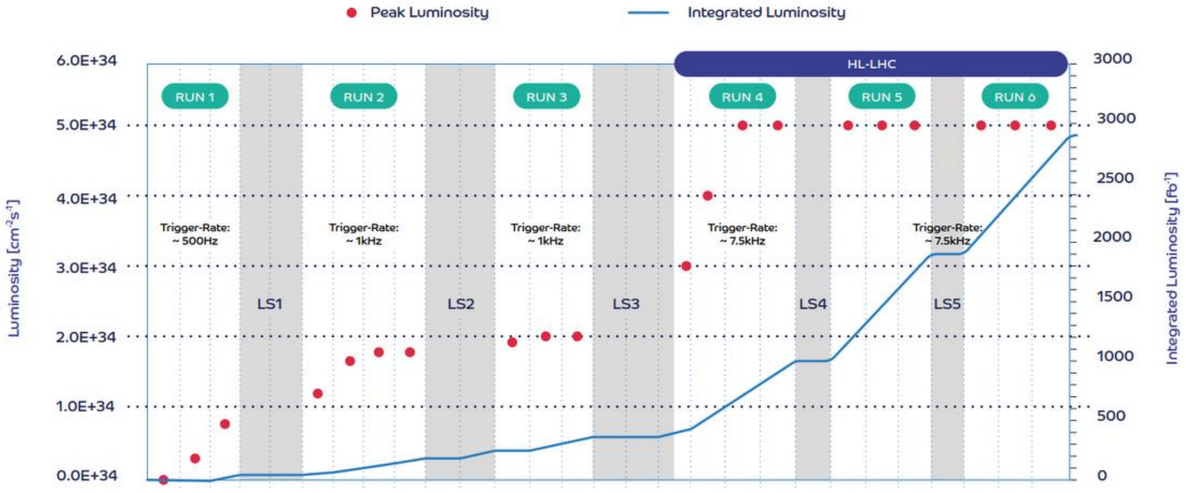
The trigger problem gets harder

HL-LHC (STARTING ~2029)

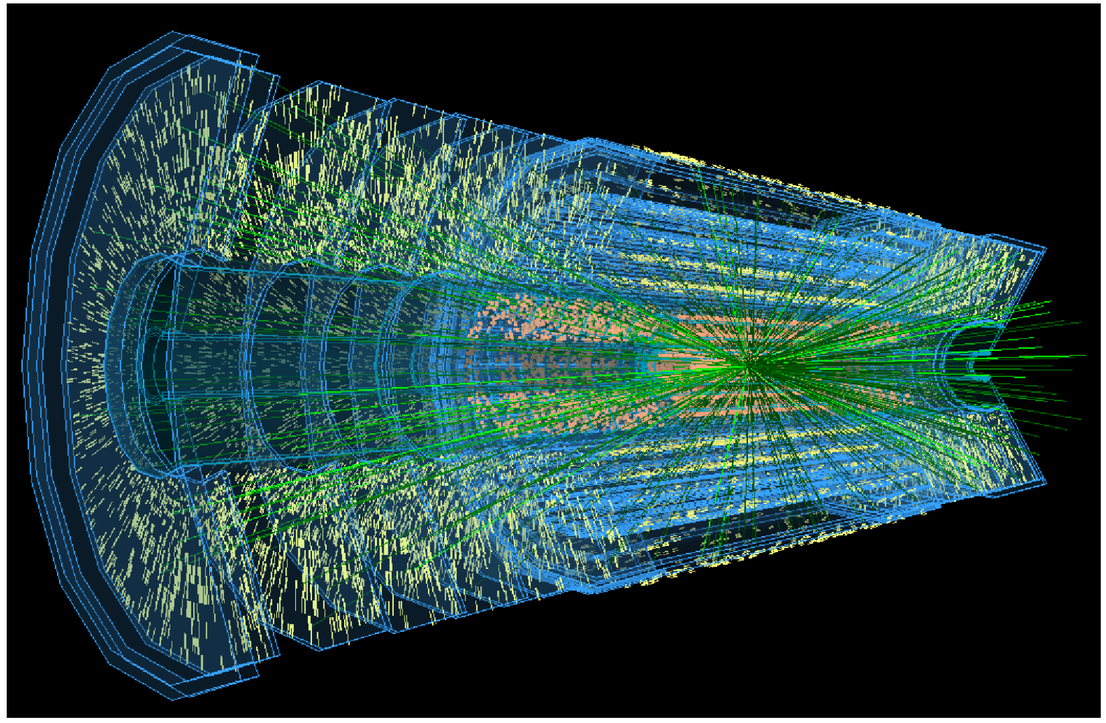
5–7× luminosity increase

~200 pileup interactions

Current trigger architectures break — L1 rates and HLT CPU time both explode.



PHASE-2 UPGRADES (ATLAS + CMS)



DAQ bandwidth: O(100 Tb/s) — requires entirely new network infrastructure

ML & FRONTIER

The new frontier

ML has transformed the HLT in the last ~5 years:

b-tagging at HLT: deep NNs (ParticleNet, DeepJet variants) deployed in production

τ identification: BDTs and DNNs → crucial for H→ττ triggers

Anomaly detection: model-agnostic triggers — CMS AXOL1TL autoencoder at L1 , running on FPGAs

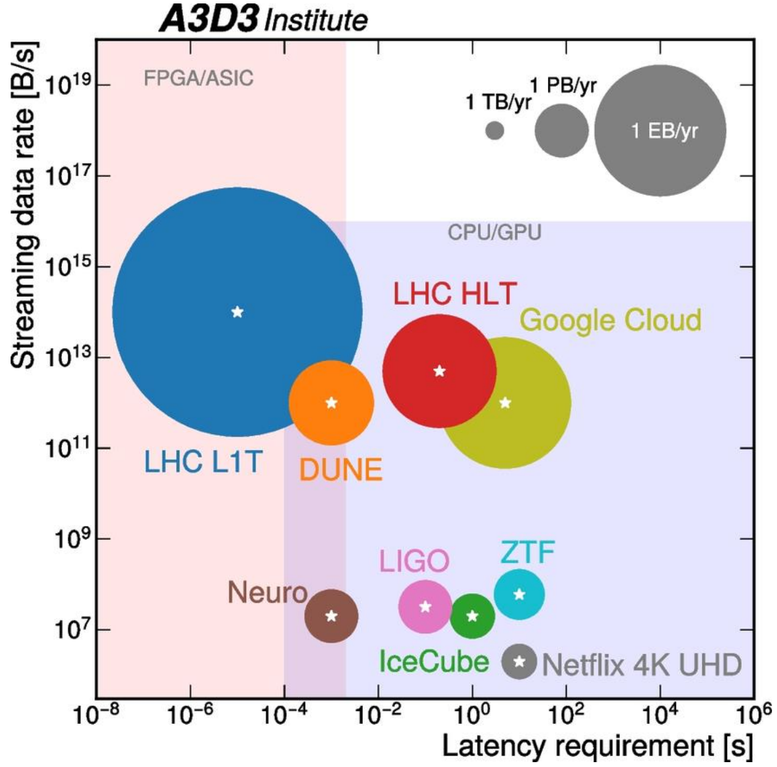
Open question: Can we trigger on "something unusual" without knowing what we're looking for?

HLS4ML — FAST ML ON FPGAS

Framework to compile NNs directly to FPGA firmware

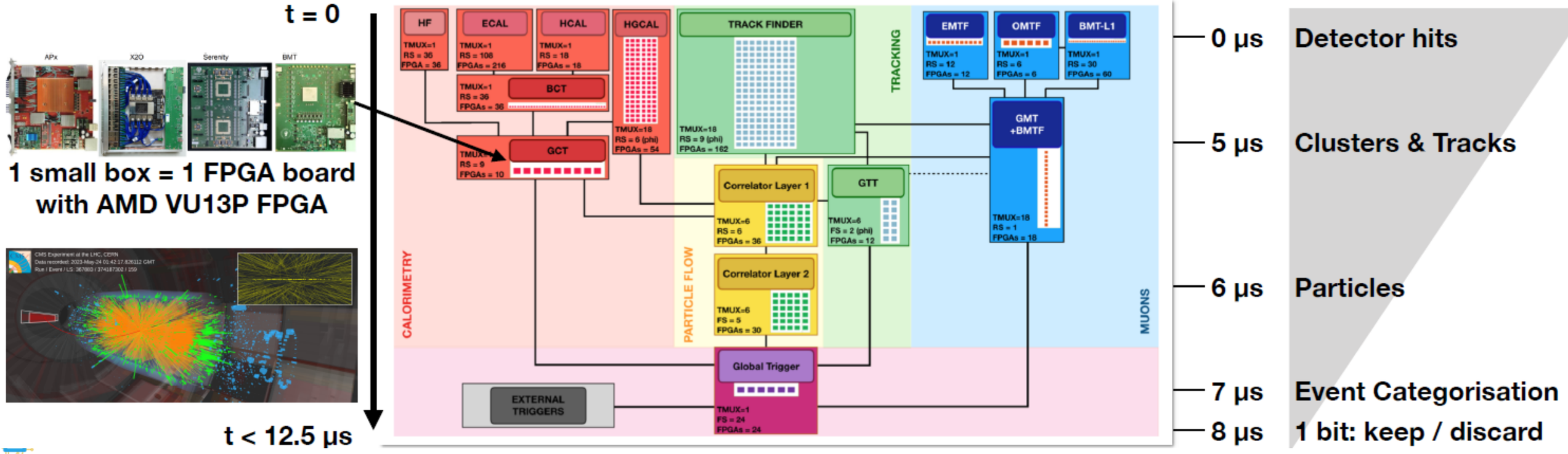
~50 ns inference latency

Compatible with Level-1!



The new frontier

- **Benefit from the upgrade** of the CMS detector: high granularity information and tracking information
- The system allows a throughput of > +64 Tb/s using top-of-the-line FPGAs and ultra-fast optical links (25 Gbps).
- Increased bandwidth to 750 kHz at increased latency of < 12.5 μs
- Incorporate sophisticated algorithms and advanced techniques to extend CMS physics acceptance



Hardware choices

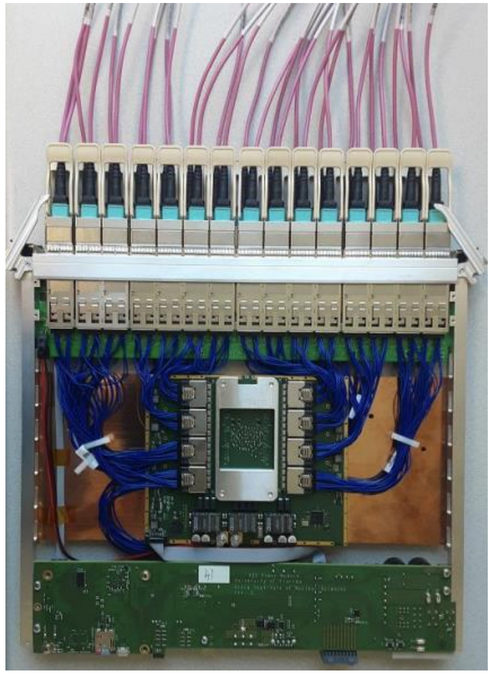
Design philosophy: Custom ATCA-boards. Generic Processing Engines → I/O, FPGA → sophisticated algo, arch flexibility

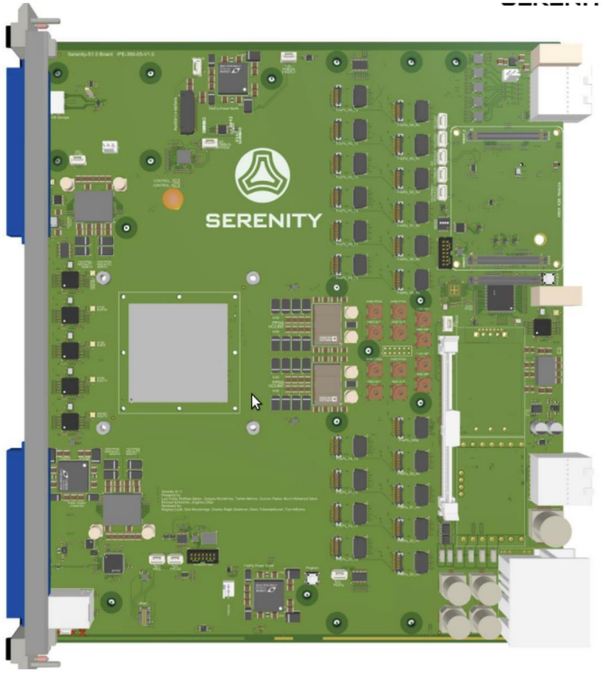
Design evolution: increased I/O and computing power

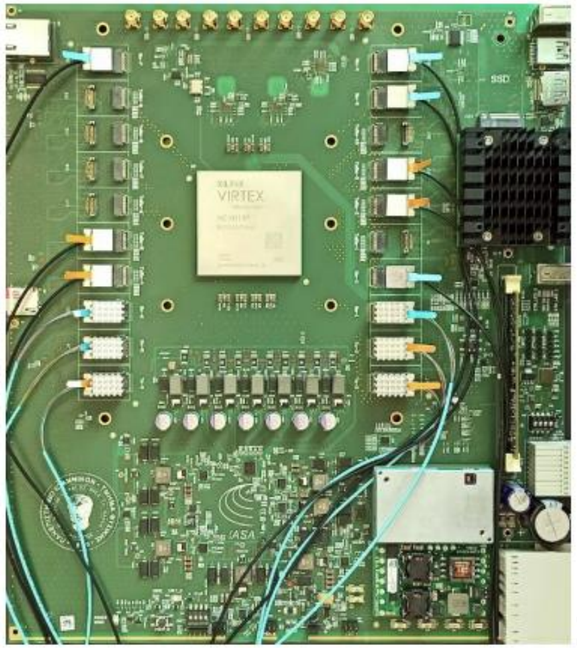
FPGA: larger A2577 pin package, Xilinx Virtex Ultrascale VU13P

Optics: New denser version of on-board fly over Samtec Firefly & QSFP

Processors on board running commercial linux for flexible configuration and monitoring







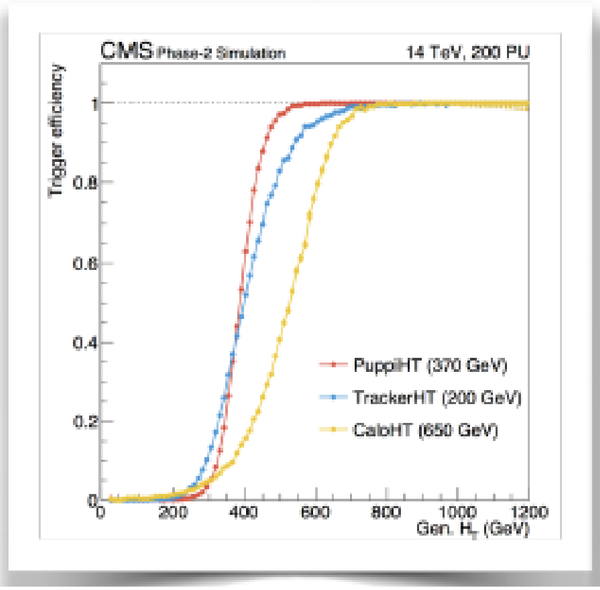
Algorithms for the Level-1 trigger

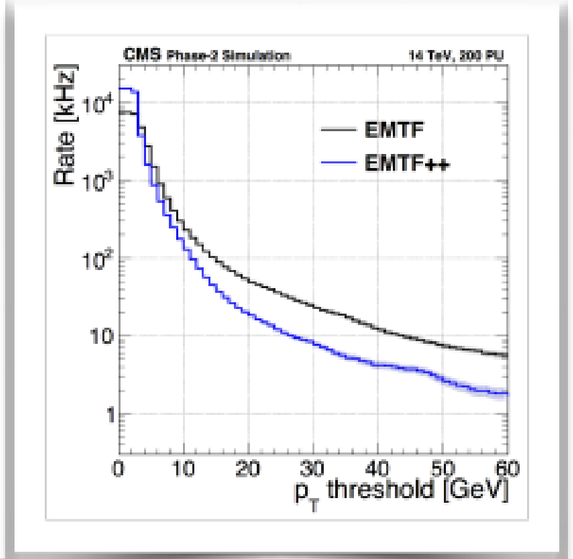
Extensive use of tracking to reach near offline performance (sharper efficiency turn-on curves) + reconstruction of Primary Vertex.

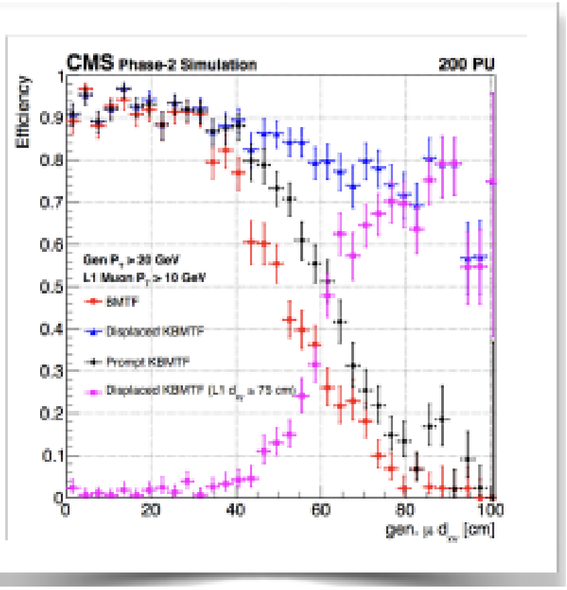
Standalone objects: robust triggers based on independent sub-detectors

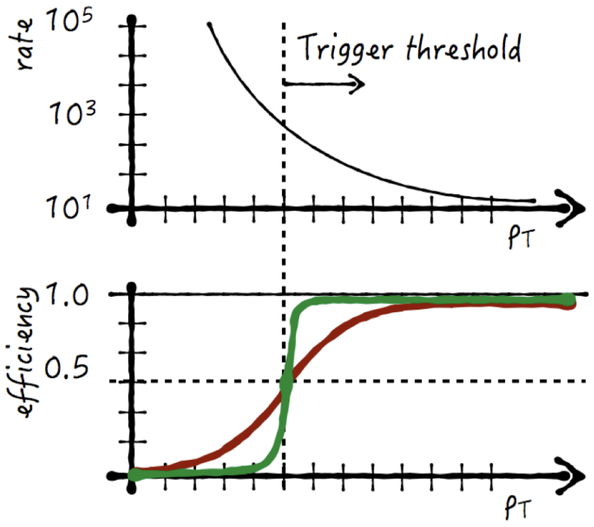
Track-matched objects: tracking used to confirm standalone Muon and Calo objects, significant improvement with simple design

Particle-flow objects: ultimate performance improvement, combine all information to match offline algorithms, require most processing time and resources for calculation









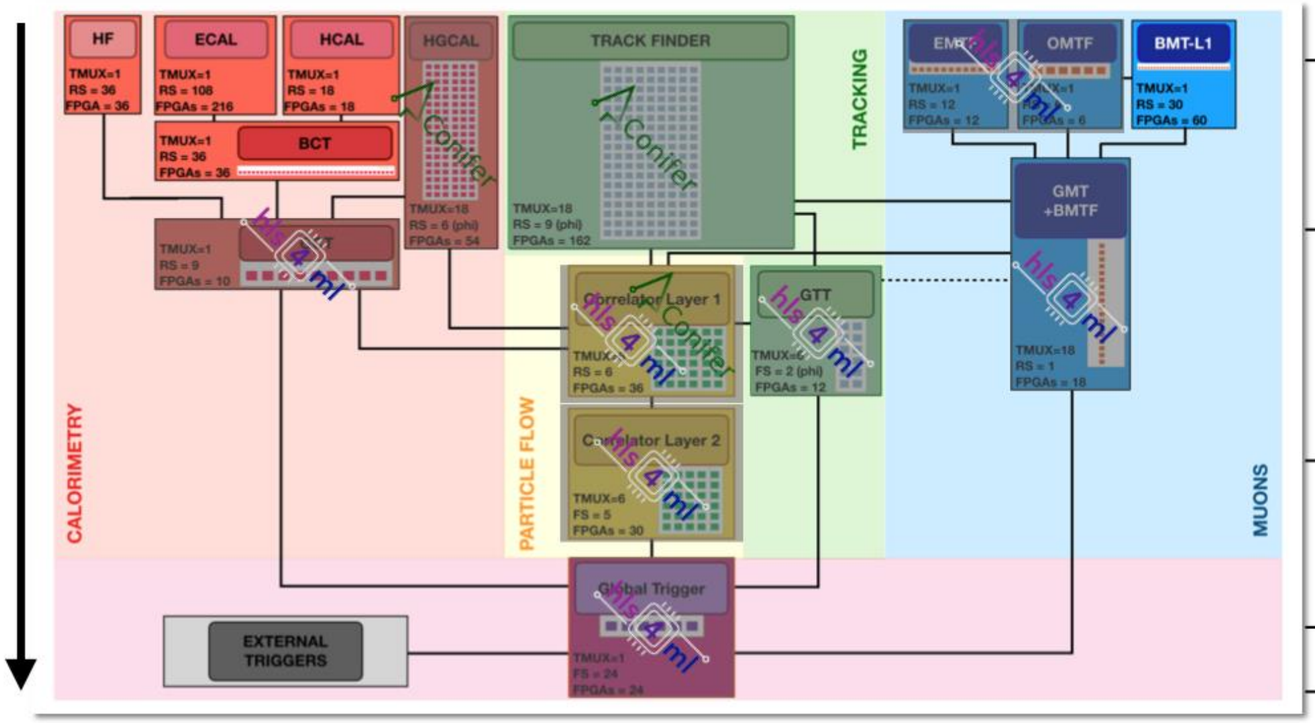
Extensive use of ML algorithms

t = 0

Each small box is one Xilinx Ultrascale+ FPGA

These have NNs and/or BDTs inside

hls4ml Conifer



t = 12.5 µs

Detector hits

Clusters & Tracks

Particles

Event Categorisation

1 bit: keep / discard

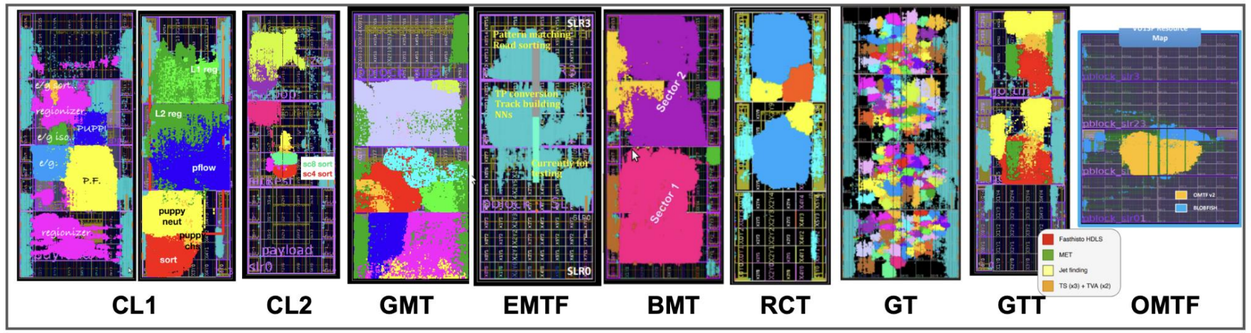
From algorithms to FW: latency and resources

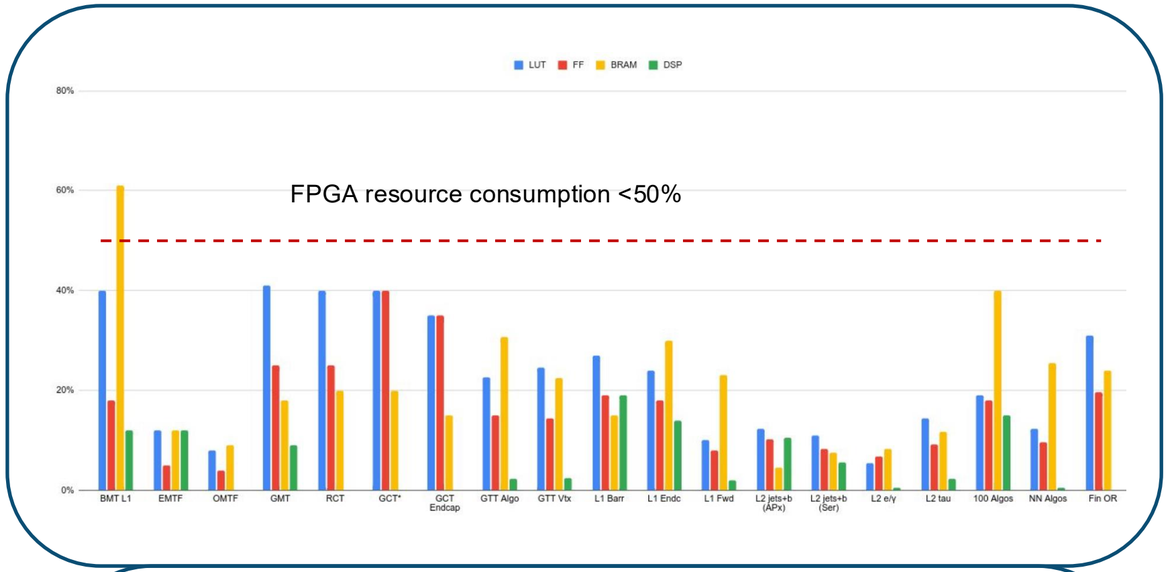
Firmware and design integration

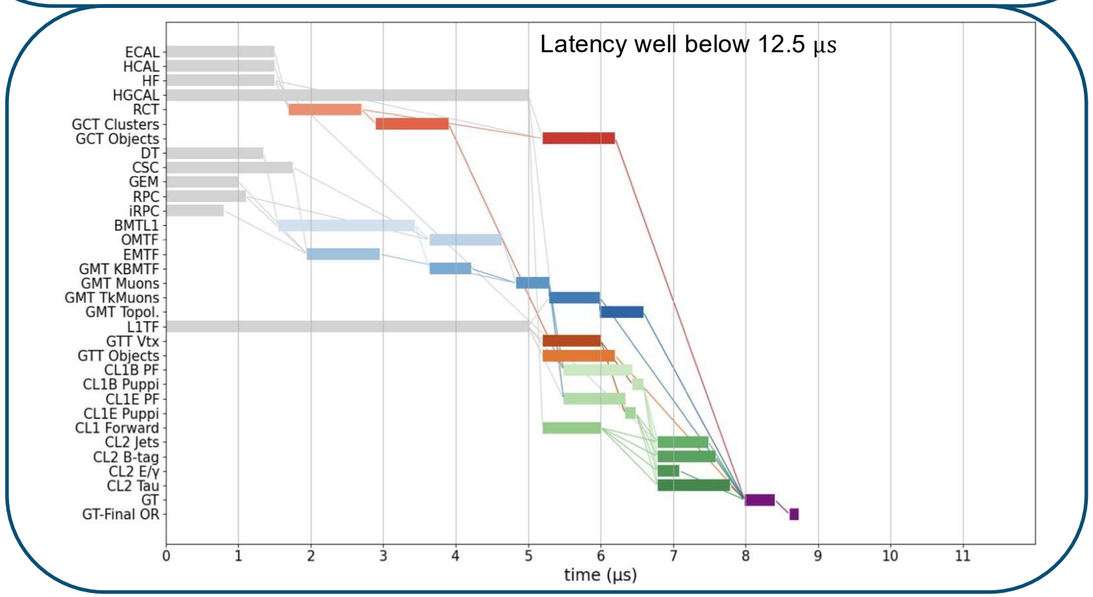
Algorithm developed mostly in C → High Level Synthesis (HLS). Using Vivado HLS, Vitis HLS

Many tools available for Machine Learning inference: hls4ml, Conifer for BDT evaluation

Continuous integration of the FW repository: resources utilization, latency.







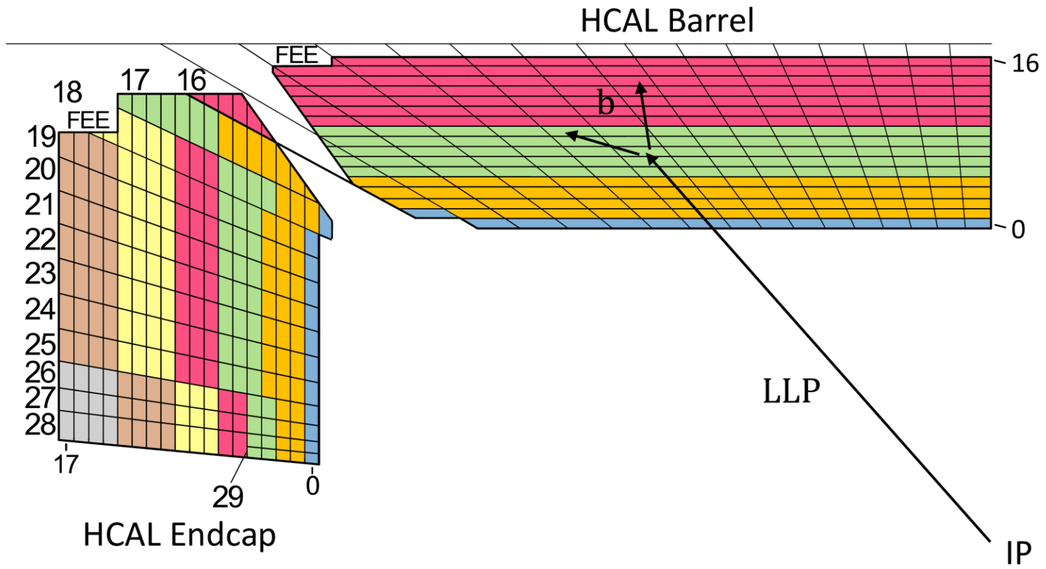
Detecting long-lived particles (testing ideas in Run-3)

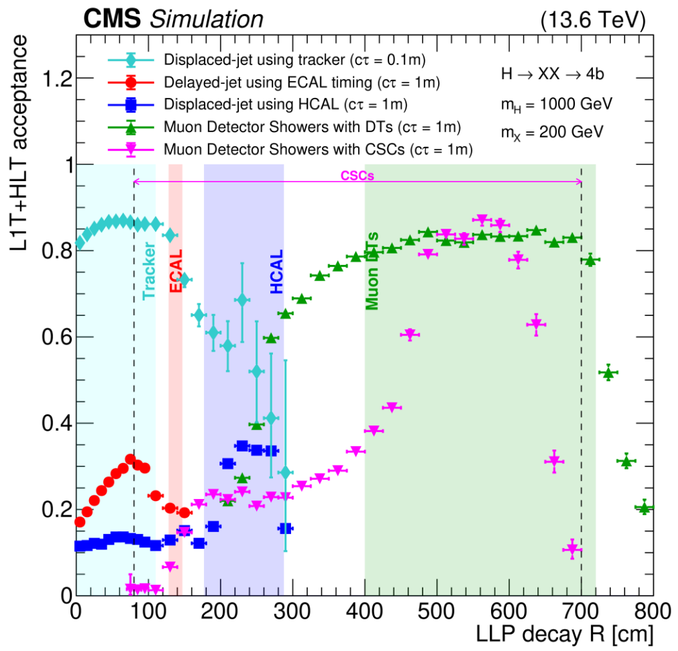
ECAL measures arrival time of objects with precision of ~200 ps (for energy deposits >50 GeV). Tau seeding at L1 and trackless jets at HLT

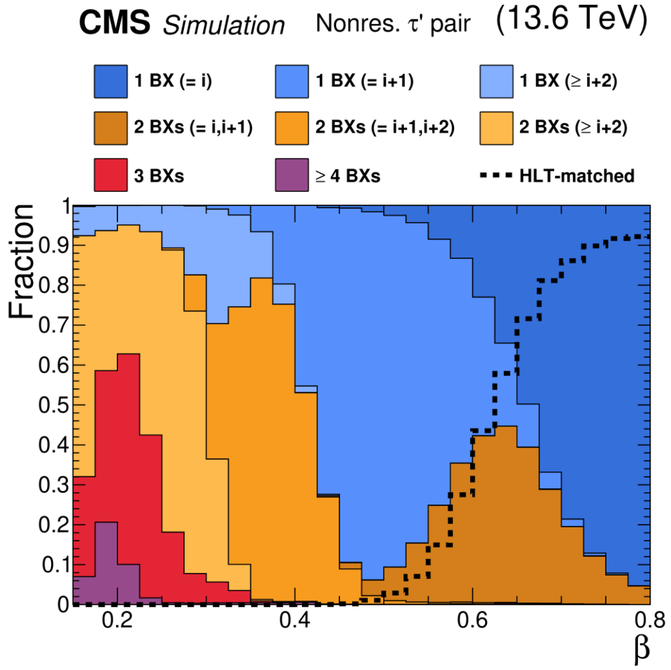
Use **HCAL** time information at the L1 trigger level to identify delayed jets (>6ns). Prompt veto applied

High multiplicity at the muon system for long-lifetimes

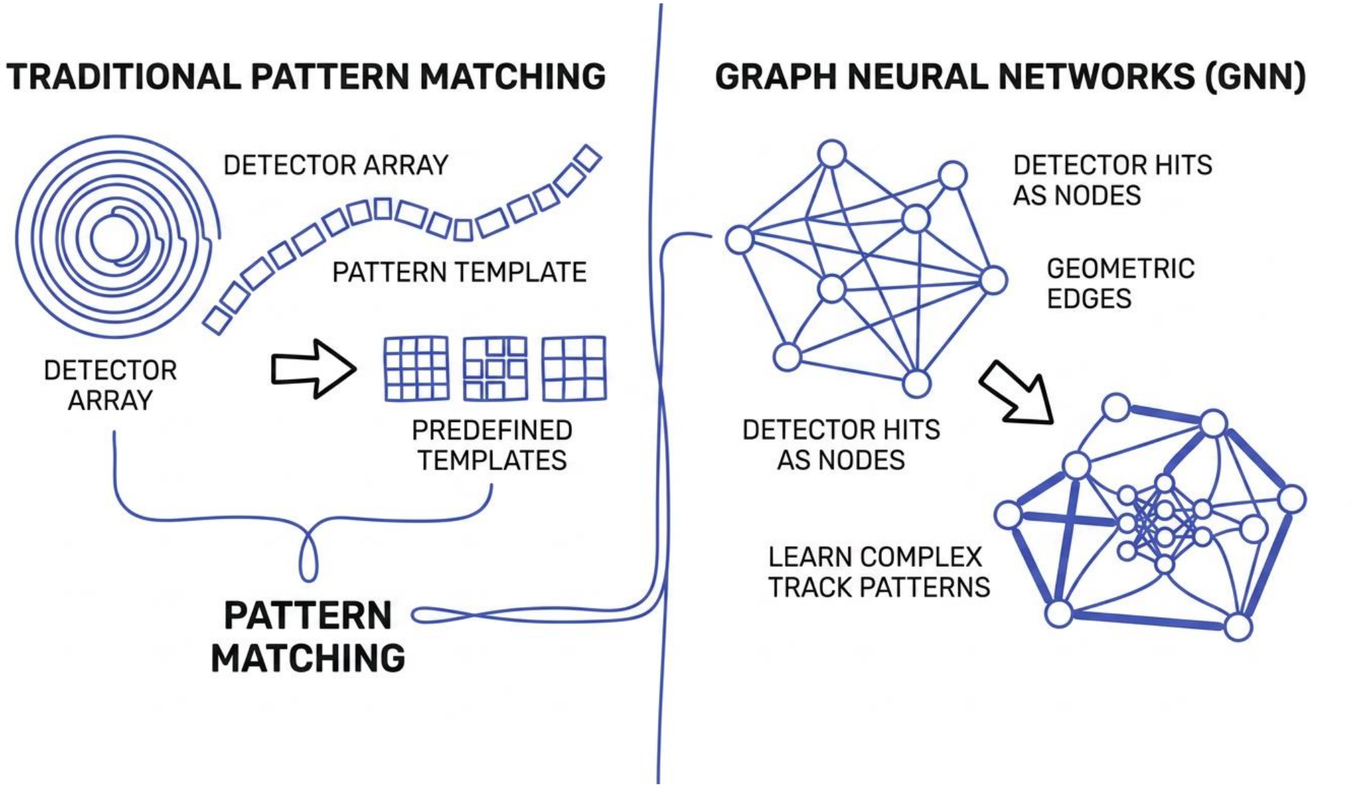
Multi-BX timing at the muon system for slow particles







Graph Neural Networks for muon track-finding



NODES

Each detected particle hit becomes a node in the graph — preserving the detector's spatial structure.

EDGES

Potential connections between hits — based on proximity and expected trajectories — form the graph edges.

LEARNING

The GNN learns patterns of genuine muon tracks, filtering out background noise while leveraging detector geometry natively.

Trade-off: data-driven recognition outperforms fixed algorithms — but variable graph sizes and tight FPGA latency budgets are the engineering challenge.

ML & FRONTIER

Anomaly detection

Where’s the new physics? To find anything, you need a trigger. If we knew what we were looking for, we’d build a trigger for it!

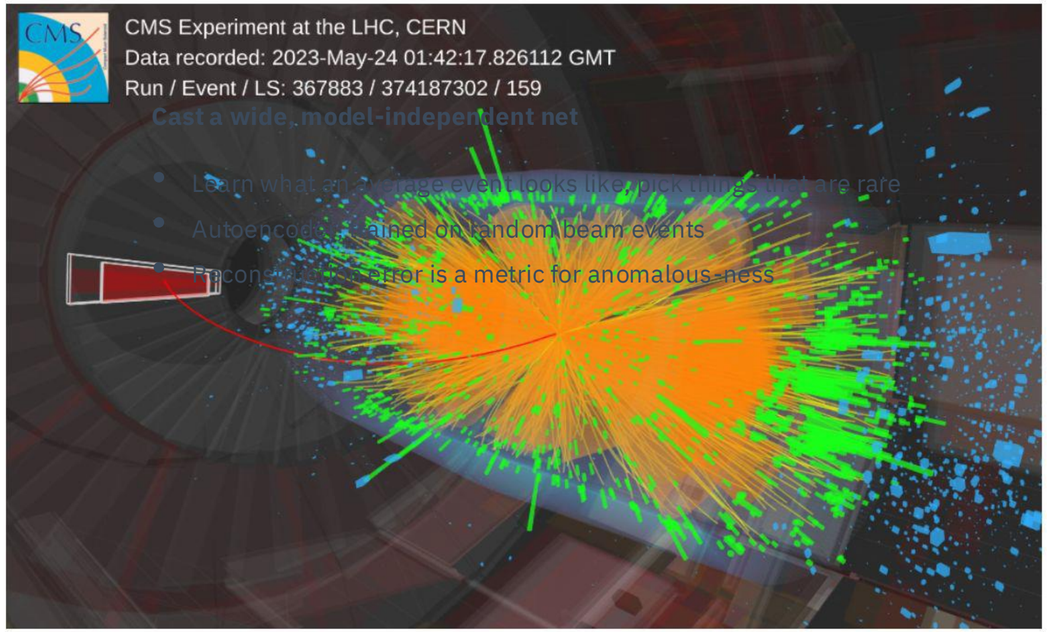
Cast a wide, model-independent net that:

Learns what an average event looks like and pick rare things

Autoencoder trained on random beam events

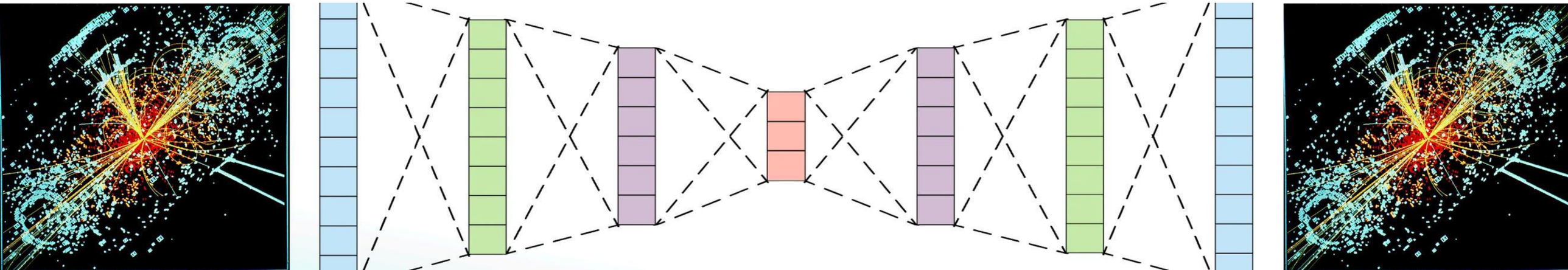
Reconstruction is a metric for anomalousness

CICADA AXOL1TL



AXOL1TL & CICADA

Low-level variables (L1T or Calorimeter objects)
Outputs an anomaly metric to keep the event or not



SUMMARY

What to remember

The trigger is **not optional** — LHC data rates exceed recordable capacity by ~10^5

The solution is **layered**: fast hardware (L1) + slower software (HLT), each doing what it does best

Different experiments have **radically different** trigger philosophies driven by their physics goals

The trigger **shapes your physics**: efficiency, bias and thresholds matter as much as detector resolution

ML and heterogeneous computing (GPUs, FPGAs) are **transforming** real-time data processing

HL-LHC will require **another generation** of trigger innovation

DISCUSSION

Things worth arguing about

"If you had infinite bandwidth, would you still want a trigger?"

→ online calibration, data quality, computing models

"LHCb removed the hardware trigger. Should everyone?"

→ luminosity dependence, detector geometry, GPU scalability

"Should the trigger be physics-model-agnostic?"

→ anomaly detection — what does "unusual" mean at a collider?

"Who owns the trigger menu — the detector team or the physics analysts?"

→ governance, prescale decisions, rare process protection

Thank you

Questions, comments, and arguments about trigger menus all welcome.

ATLAS Trigger/DAQ TDR: CERN-LHCC-2017-020 · Allen: Comput.Softw.Big Sci. 4 (2020) 7 · hls4ml: JINST 13 (2018) P07027

From interface YAMLs to verified DUTs

PLUGIN-AUTHORED CONTRACTS

`modules.yml` `interfaces/*.interface.yaml` `designs/design.yml` `design.verification.yml`

consumes

ARC FRAMEWORK

topgen
load contracts · derive wiring · generate the stitched DUT

HLS helpers
generate TCL · run HLS stages · collect metrics

fw_verify
per-flow runtime · run csim / xsim · preflight

generates

GENERATED ARTEFACTS

`algo_top.v` `build_manifest.json` `port_map.yaml` `tb_*.sv` `verify.flow.yml` `wave.tcl`