Allen: LHCb’s HLT1 on GPUs

Dorothea vom Bruch

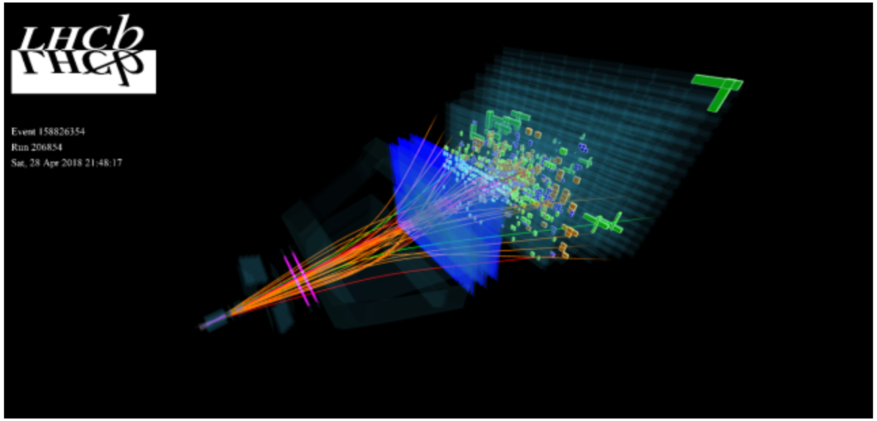
FCC-LHCb software discussion

September 22nd 2020

European Research Council
Established by the European Commission

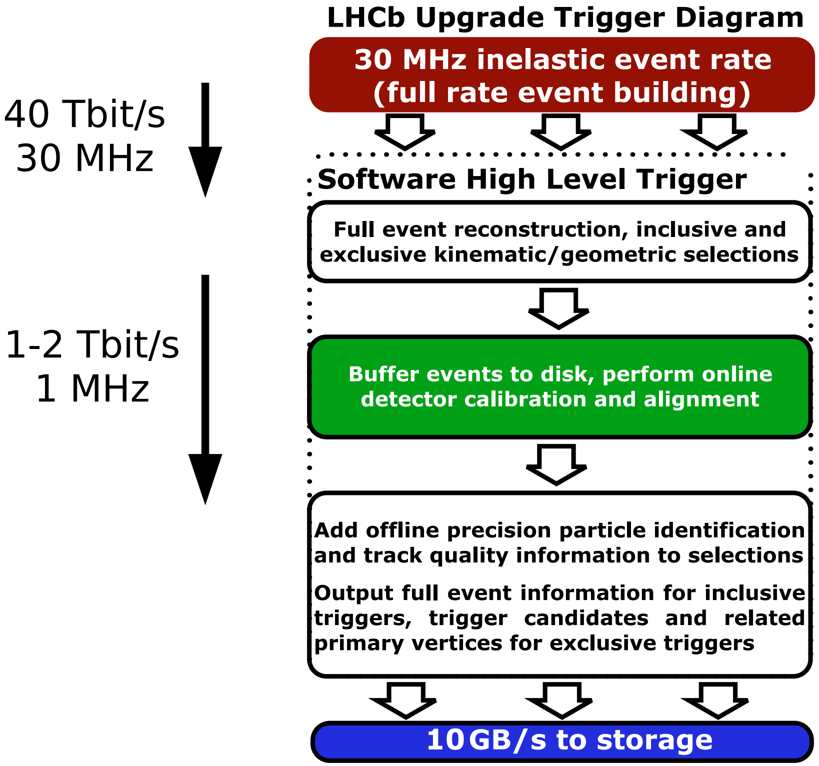
Outline

- Challenge of LHCb’s Run 3 trigger
- Why were GPUs considered?
- Performance: Physics & throughput
- Software framework



The LHCb Trigger in Run 3 of the LHC

Trigger in Run 3

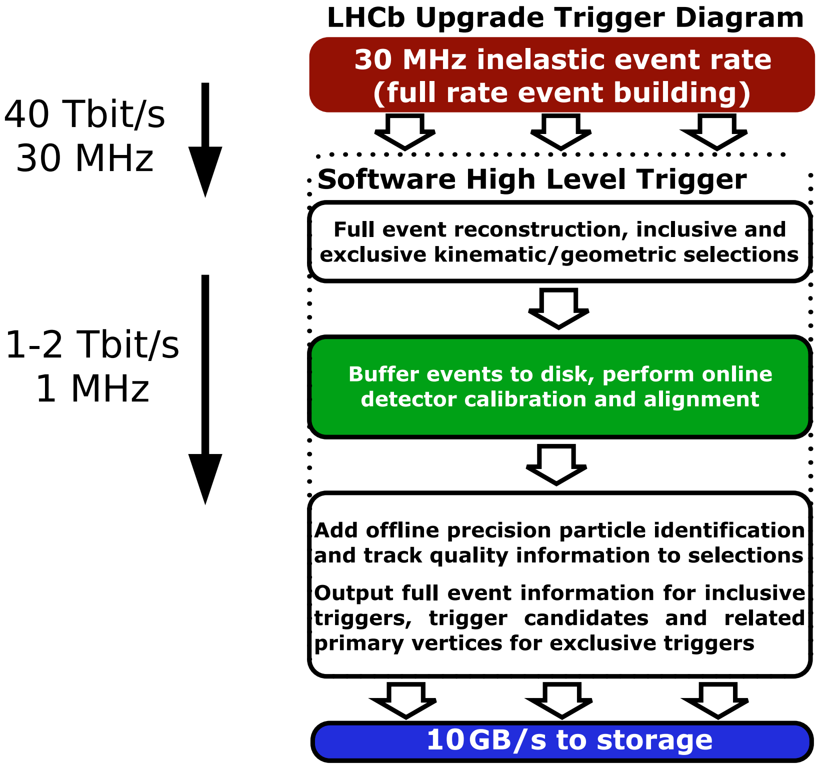


High Level Trigger 1 (HLT1)

- Full charged particle track reconstruction
- Few inclusive single or two-track selections
- Reduce event rate by roughly factor 30

- **High Level Trigger 2 (HLT2)**
- Aligned and calibrated detector
- Offline-quality track reconstruction
- Particle identification
- Full track fitting

Trigger in Run 3



Track reconstruction @ 30 MHz is a huge computing challenge

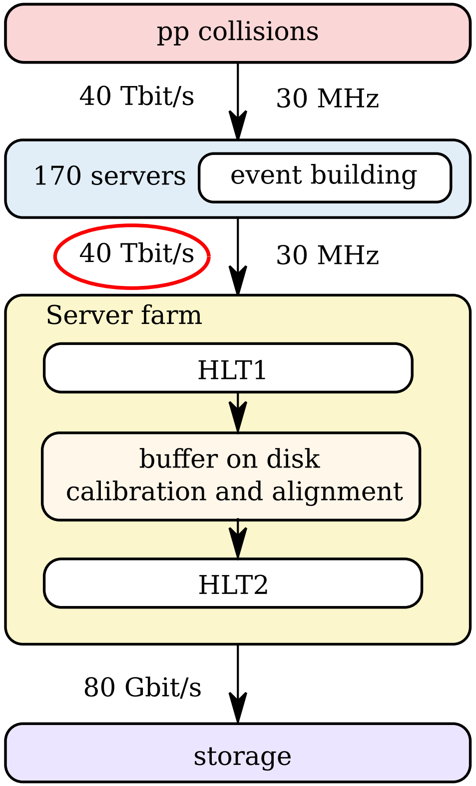
- **...igger 1 (HLT1)**
- Full ... ...rack reconstruction
- Few inclusive ... ...ck selections
- Reduce event rate b... ...30

- **High Level Trigger 2 (HLT2)**
- Aligned and calibrated detector
- Offline-quality track reconstruction
- Particle identification
- Full track fitting

HLT1 architecture choice

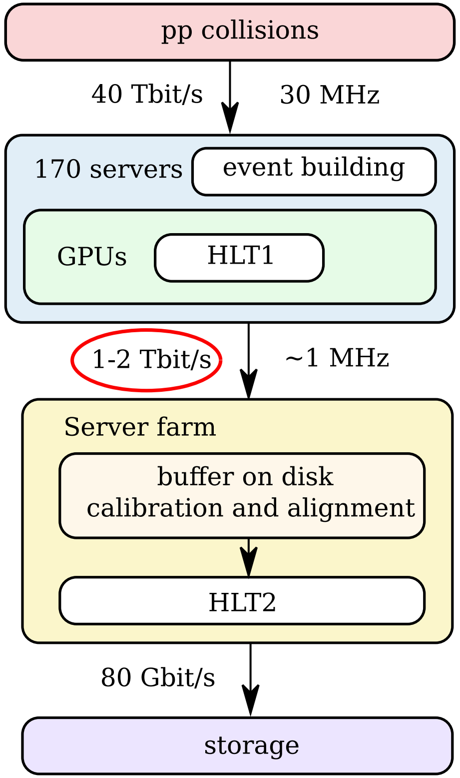
Proposal in TDR (2014)

CERN-LHCC-2014-016



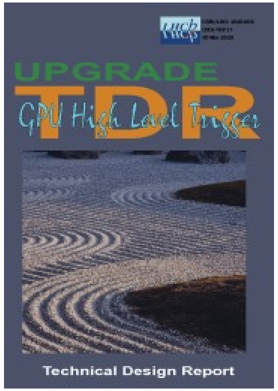
Updated strategy (as of 5/2020)

CERN-LHCC-2020-006



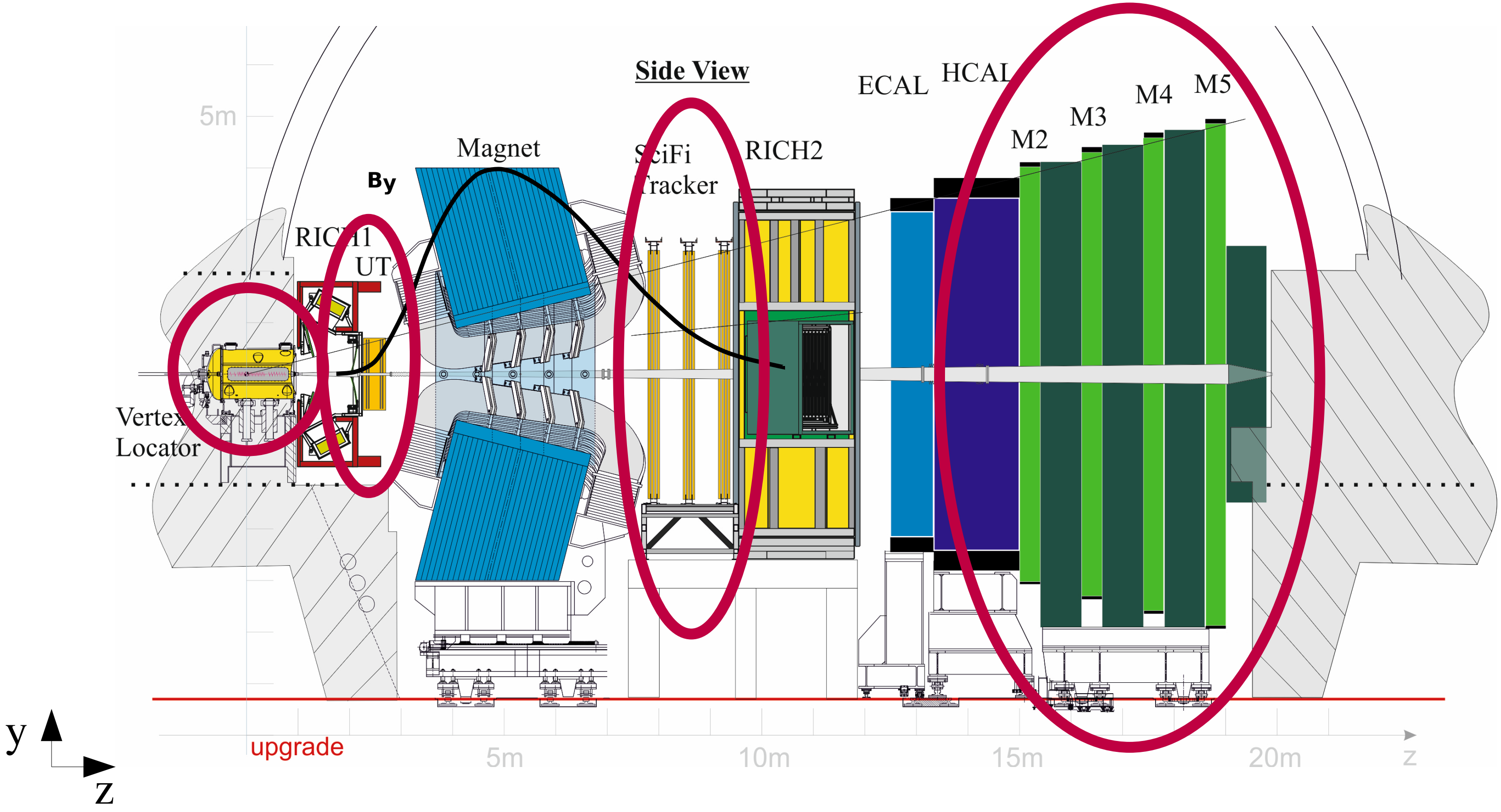
- Developed two solutions simultaneously
- Both the multi-threaded CPU & the GPU HLT1 fulfilled the requirements from the 2014 TDR
- LHCb was in the luxury situation to choose among them
- Compared physics performance & price-performance

  → decided for GPU solution



Why were GPUs considered for the HLT1?

LHCb HLT1 detectors

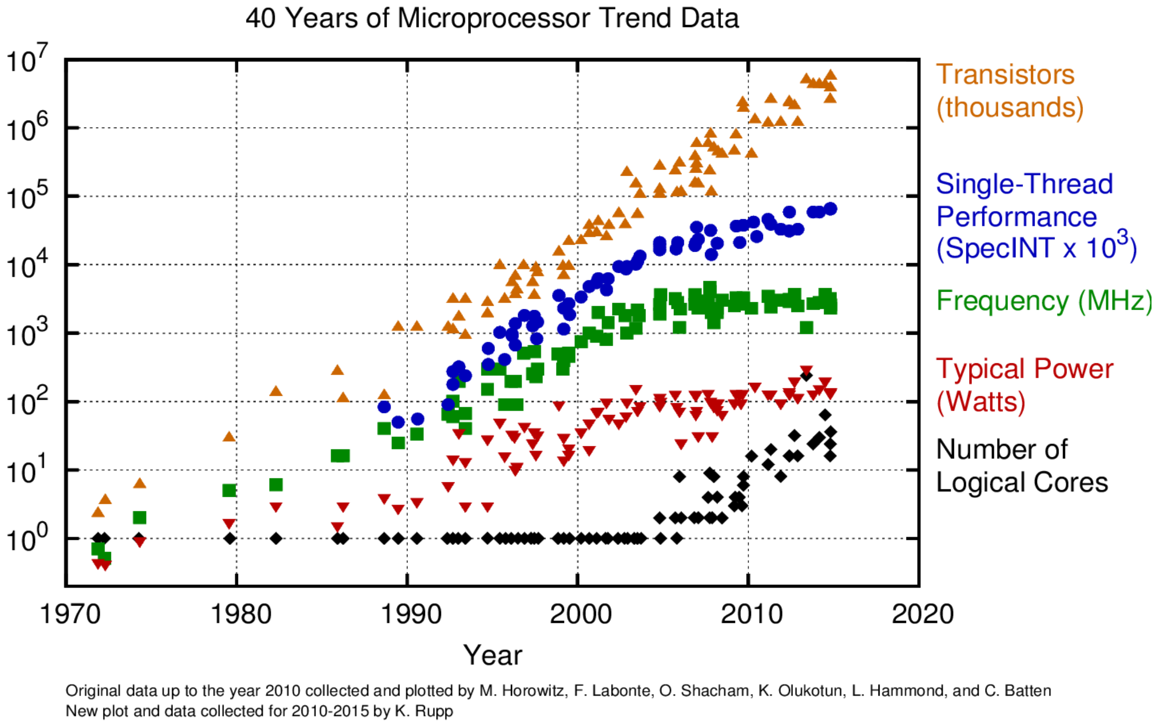


How does the HLT1 map to GPUs?

Characteristics of LHCb HLT1	**Characteristics of GPUs**
Intrinsically parallel problem: - Run events in parallel - Reconstruct tracks in parallel	Good for - Data-intensive parallelizable applications - High throughput applications
Huge compute load	Many TFLOPS
Full data stream from all detectors is read out → no stringent latency requirements	GPUs have higher latency than CPUs, not as predictable as FPGAs
Small raw event data (~100 kB)	Connection via PCIe → limited I/O bandwidth
Small event raw data (~100 kB)	Thousands of events fit into O(10) GB of memory

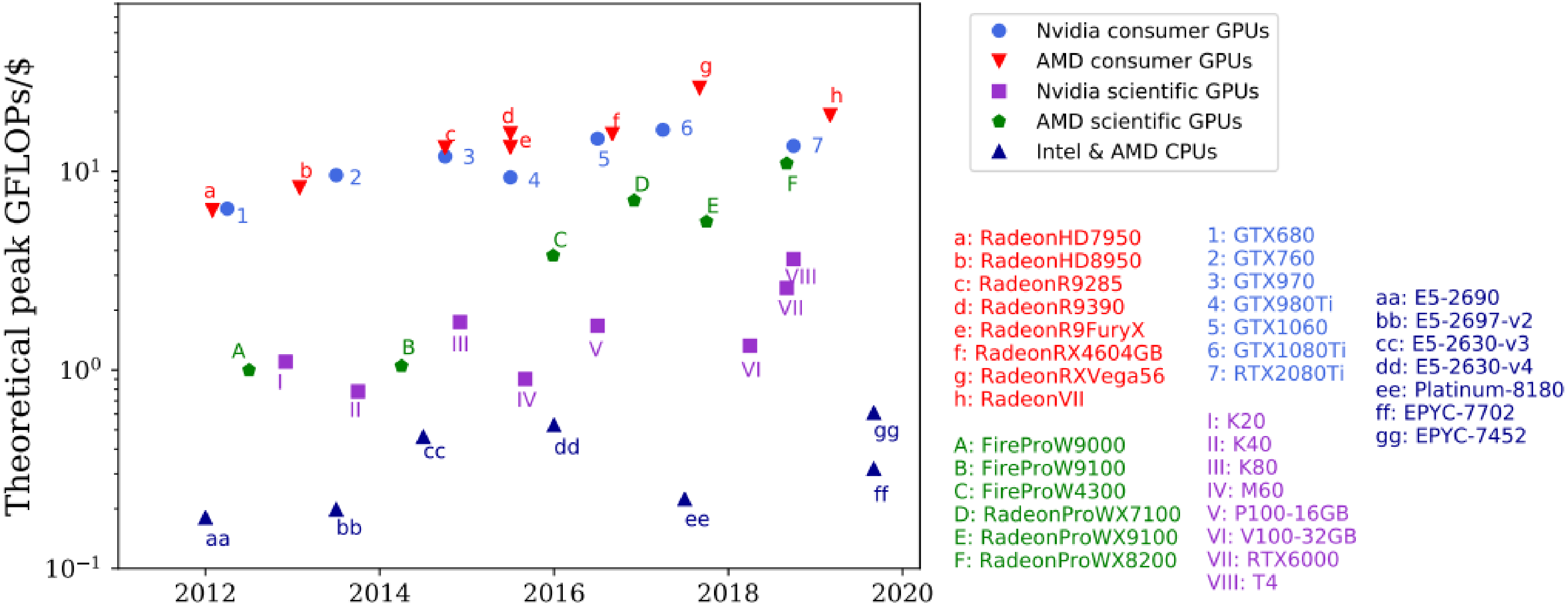
Perfect fit!

Moore’s law today



Need to go massively parallel to make best use of today’s processors

Theoretical FLOPs/$: GPUs & CPUs



JINST 15 C06010 (2020)

GPUs offer the most FLOPs / $

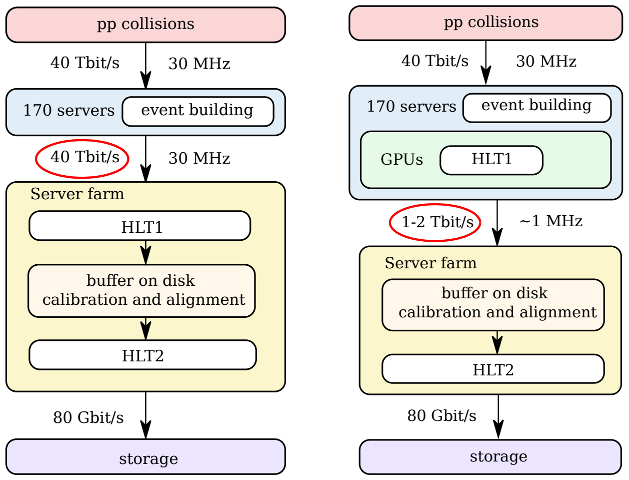
Benefits of a GPU HLT1 for LHCb

- Prepare LHCb for the era of heterogeneous computing
- Future DAQ systems will most certainly be heterogeneous in one way or another
- Reduce data rate early

  → financial benefit due to reduced network cost:

  100 Gbit/s → 10 Gbit/s
- Baseline HLT1 reconstruction fits comfortably into throughput budget

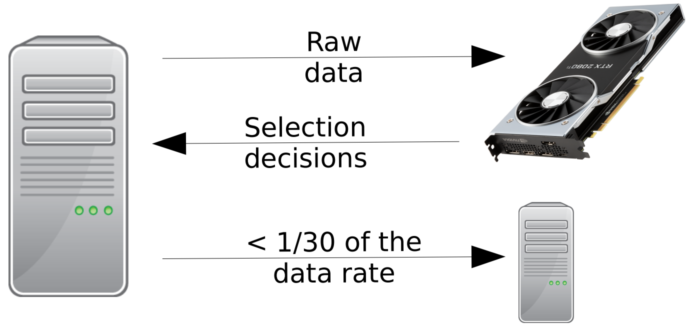
  → can increase physics reach by adding further algorithms to the HLT1 sequence
- Train HEP physicists in GPU computing

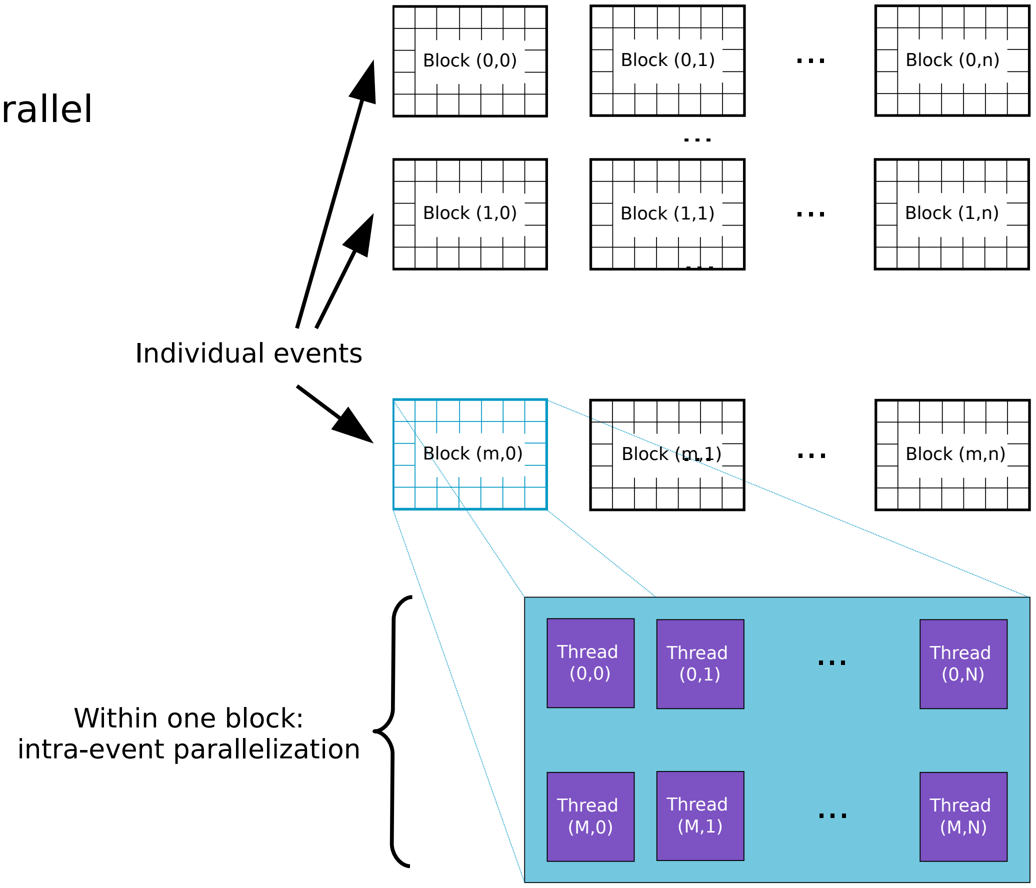


Performance: Physics & throughput

HLT1 on GPUs

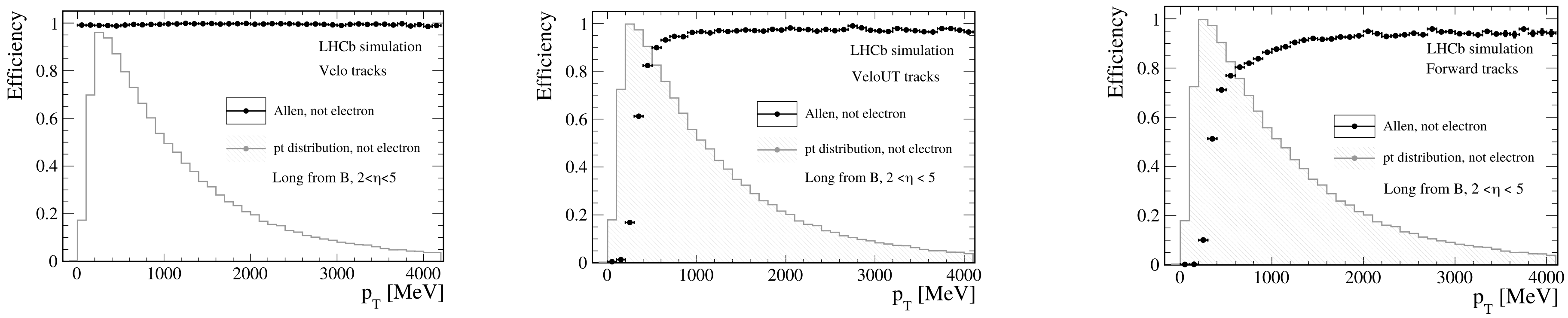
- Thousands of events are processed in parallel
- In addition: intra-event parallelization
- Only single precision is used





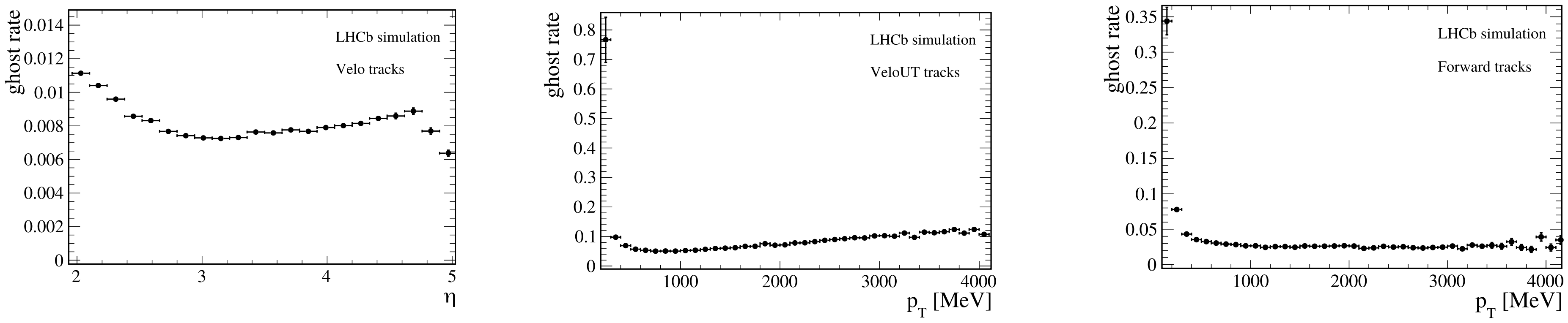
Physics performance: Track reconstruction

Track reconstruction efficiency



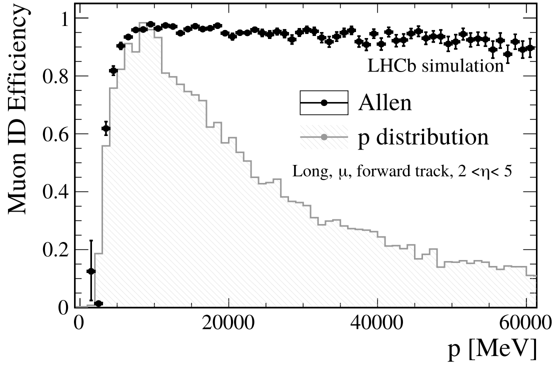
LHCb-FIGURE-2020-014

Fake rate

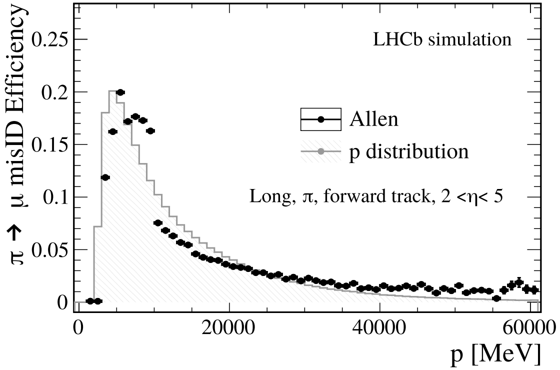


Physics performance: Muon ID, PVs, resolution

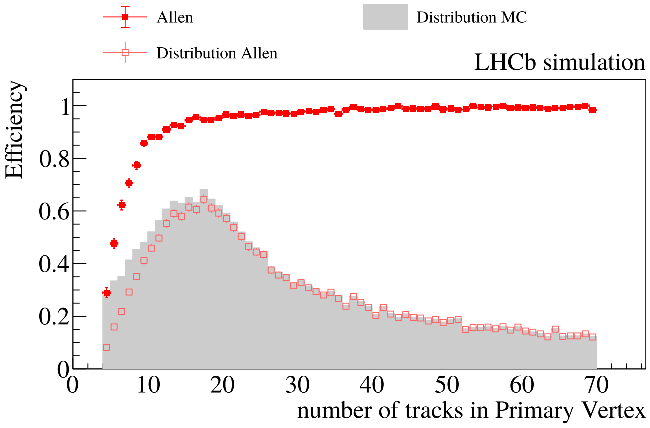
Muon ID efficiency



π→μ mis-ID efficiency

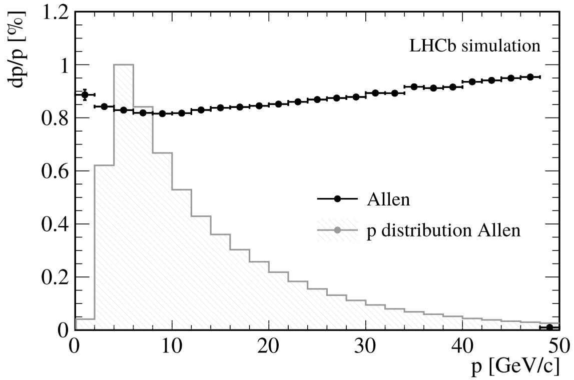


PV reconstruction efficiency



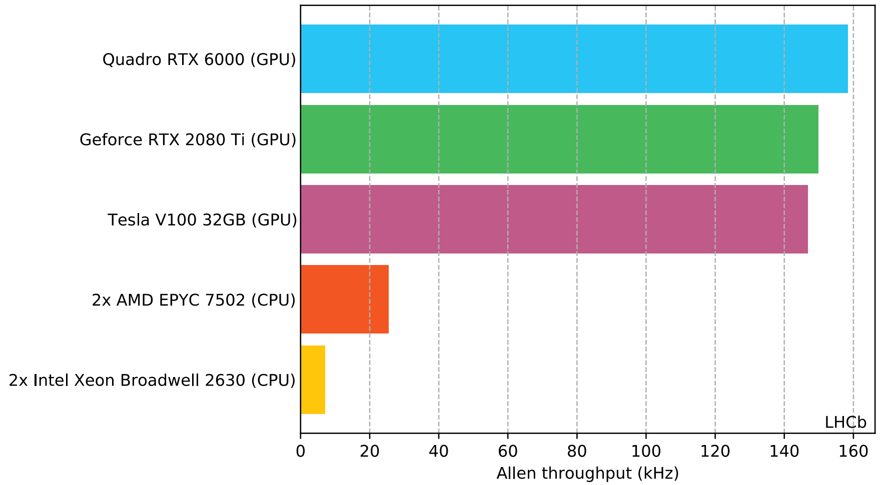
LHCb-FIGURE-2020-014

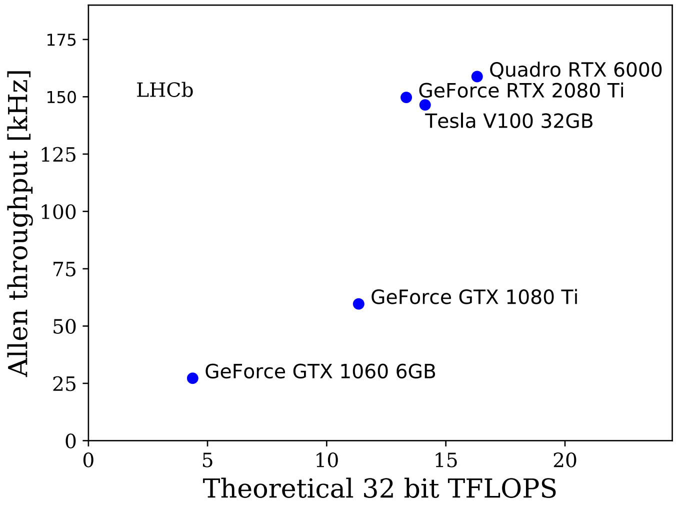
Momentum resolution



Computing performance

LHCb-FIGURE-2020-014





- Require about 215 GPU cards to process full HLT1 @ 30 MHz
- Have slots for 500 cards
- Computational performance scales well with GPU generations → expect improvements with next generation cards (coming out this year)

The software framework

The Allen project

- Fully standalone software project: https://gitlab.cern.ch/lhcb/Allen
- Lightweight framework
- Only requirements:
  C++17 compliant compiler, CUDA v10, boost, ZeroMQ
- Cross-architecture compatibility (HIP/ROCm, x86) via macros
- Standalone framework for fast development & easy entry-point for new users
- Compilation with Gaudi to integrate Allen with other LHCb software

- Named after Frances E. Allen

Framework requirements

Support various architectures

Low entry point for user

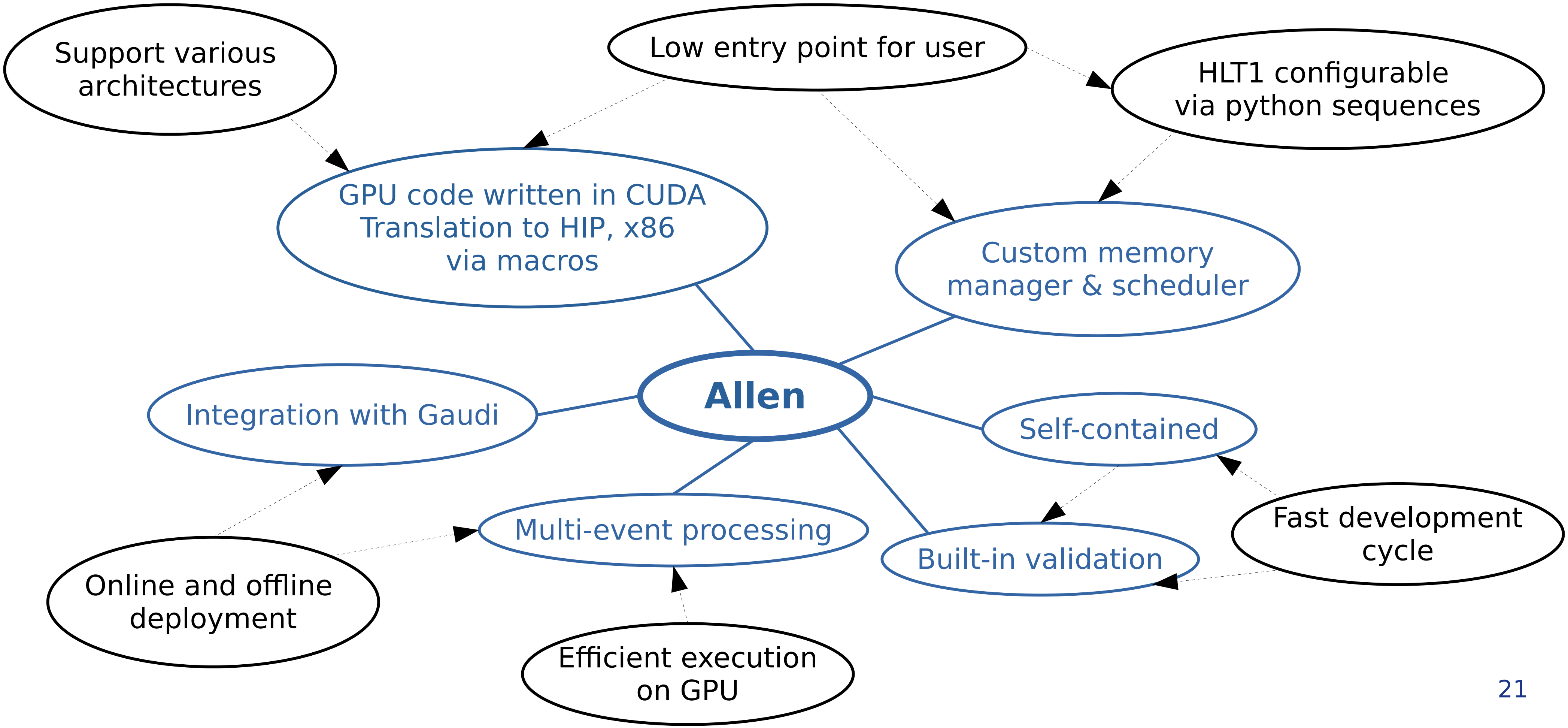
HLT1 configurable via python sequences

Fast development cycle

Online and offline deployment

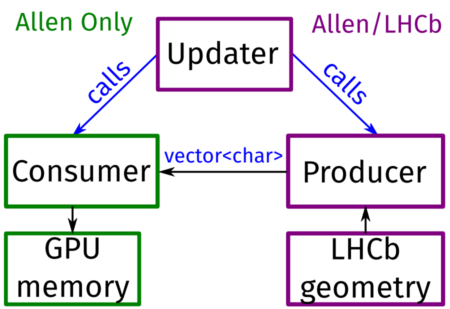
Efficient execution on GPU

Framework design



Online integration

- Event-loop steered by Allen in multi-event batches
- Non-event data requested from Gaudi upon run change
    - Aligned & calibrated detector description
    - Magnet polarity
    - Special running conditions
- Raw data from selected events + decision reports sent to HLT2



Offline integration

- For simulation & offline studies
- Use x86 compilation of Allen → can run on the WLCG
- Event loop steered by Gaudi
- Allen called one event at a time

Summary

- LHCb will commission the first complete high-throughput GPU trigger for an HEP experiment
- With a heterogeneous trigger LHCb can benefit from future industry developments
- Allen framework provides various tools that are independent of LHCb software: scheduler, memory manager, event loop, cross-architecture macros
- Interest from other experiments, possible integration with ACTS in the near future

Further information:

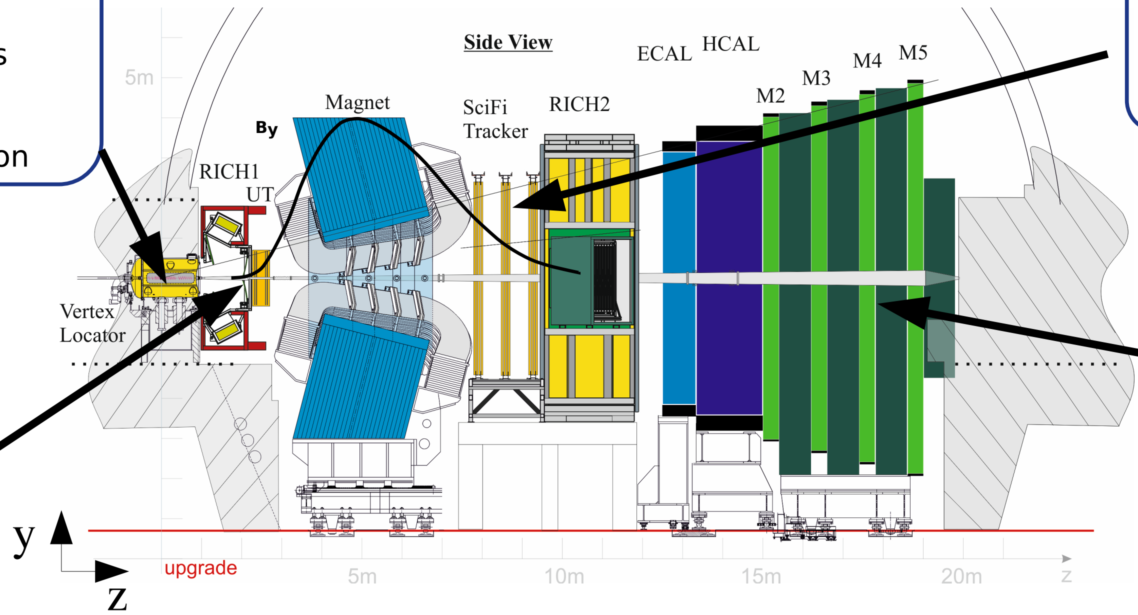
- Allen TDR
- Allen publication in CSBS
- Latest public results: LHCb-FIGURE-2020-014
- Allen gitlab repository

Backup

LHCb HLT1 elements

Velo
- Decode raw data
- Clustering of measurements
- Track reconstruction
- Primary vertex reconstruction

SciFi
- Decode raw data
- Track reconstruction



Muons
- Decode raw data
- Match hits to tracks

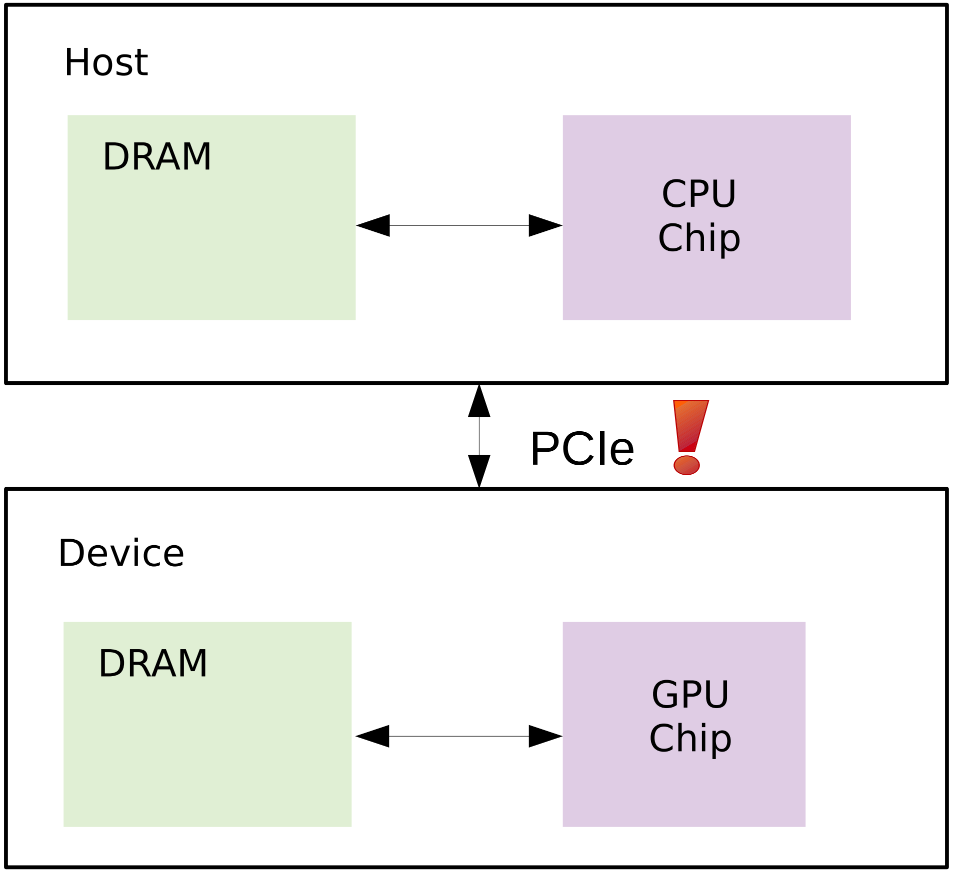
UT
- Decode raw data
- Track reconstruction

Track fit: Kalman filter

Find secondary vertices

Selections
- 1-track selection
- 2-track selection
- Based on p, p_t, displacement, vertex criteria and muon identification

A GPU’s natural habitat



PCIe generation	1 lane	16 lanes	Year
3.0	985 MB/s	15.75 GB/s	2010
4.0	1.97 GB/s	31.5 GB/s	2017

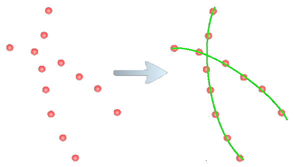
Common parallelization techniques

Raw data decoding

- Transform binary payload from subdetector raw banks into collections of hits (x,y,z) in LHCb coordinate system
- Parallelize over all subdetectors and readout units

Track reconstruction

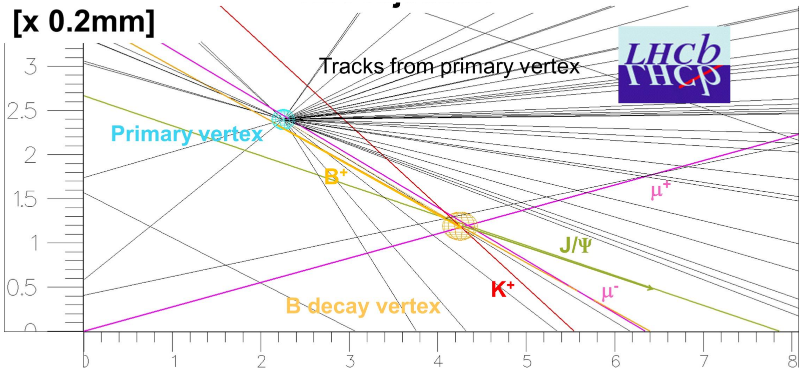
- Consists of two steps:
  - Pattern recognition: Which hits belong to which track?
  - Track fitting: Done for every track
- Parallelize over combinations of hits and tracks



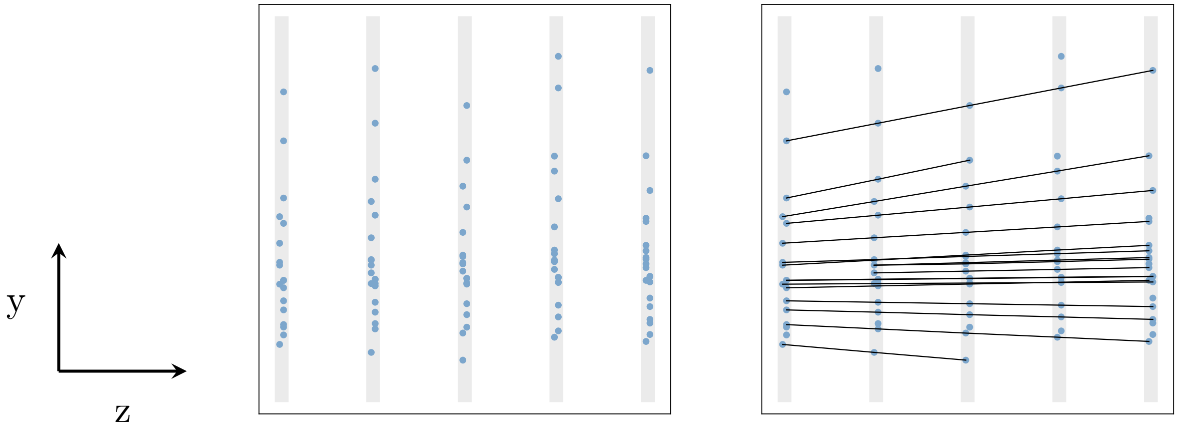
f(x) = ... +/- ...

Vertex finding

- Reconstruct primary and secondary vertices
- Parallelize across combinations of tracks and vertex seeds

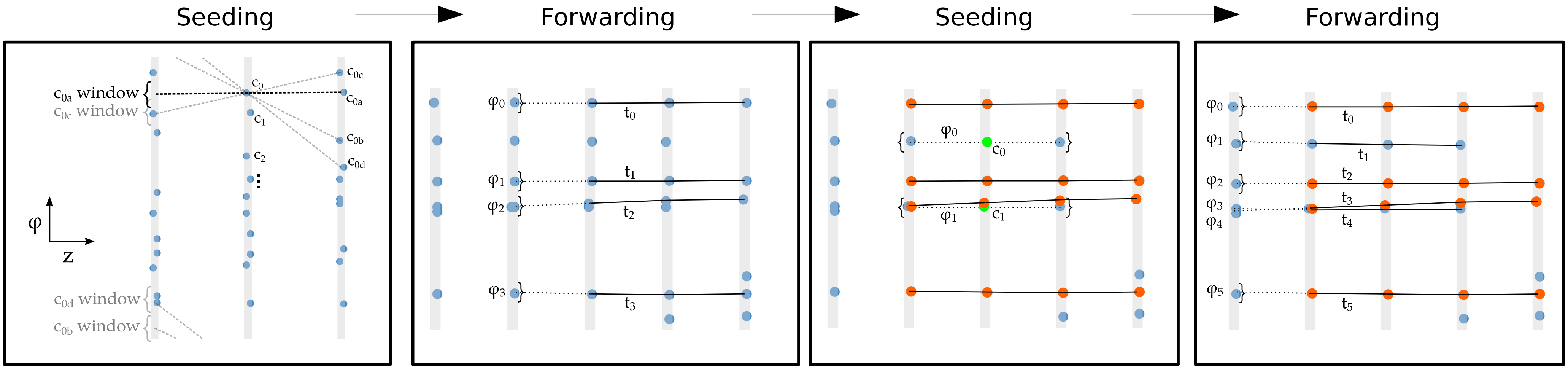


Example: Velo track reconstruction



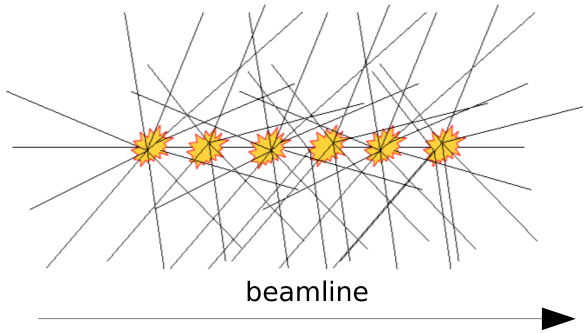
- No magnetic field in the Velo detector
- → straight line tracks
- Tracks from origin traverse detector in line of constant phi

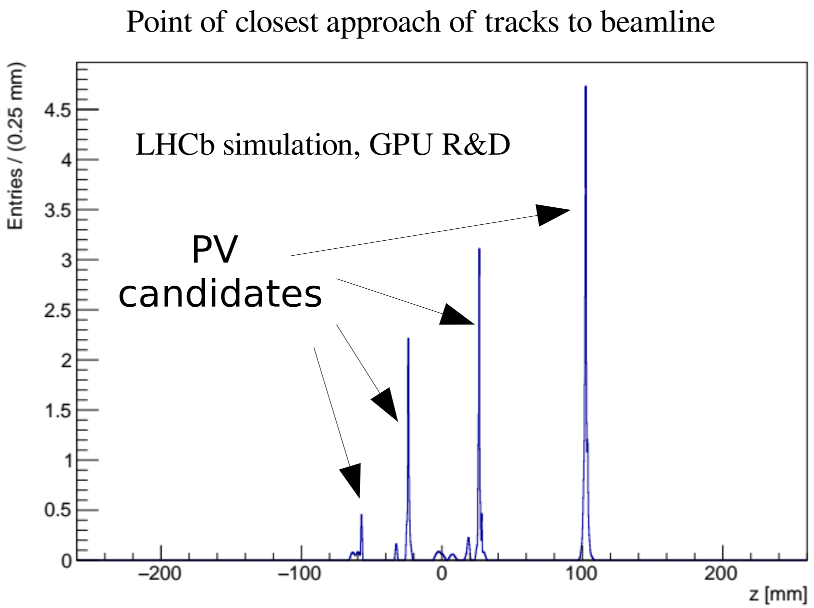
Example: Velo track reconstruction on GPUs



- Hits sorted by phi → memory accesses as contiguous as possible
- Seeding / forwarding separate kernels (parallelized algorithms)
    - Highly parallelized (across 3-hit combinations and track seeds)
    - Branching minimized

Example: Primary vertex reconstruction





- Histogram of track z-positions at beamline
- Clusters in histogram → PV candidates
- Fill histogram in parallel
- Every track contributes to every PV candidate with a weight → no inter-dependence among PV candidates
- PV candidate fitting parallelized across
  - PV candidates
  - Tracks