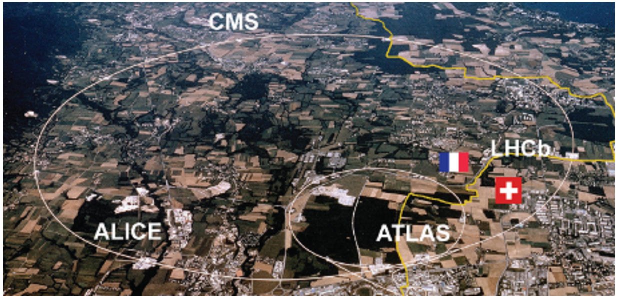
The Allen Project: a GPU trigger for LHCb

Daniel Craik (MIT)
on behalf of the LHCb collaboration

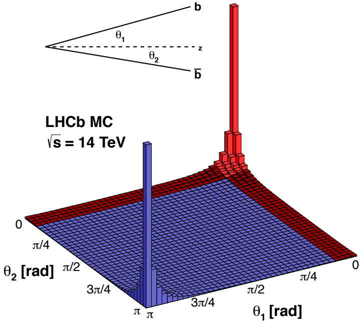
2020-01-27

The LHCb detector

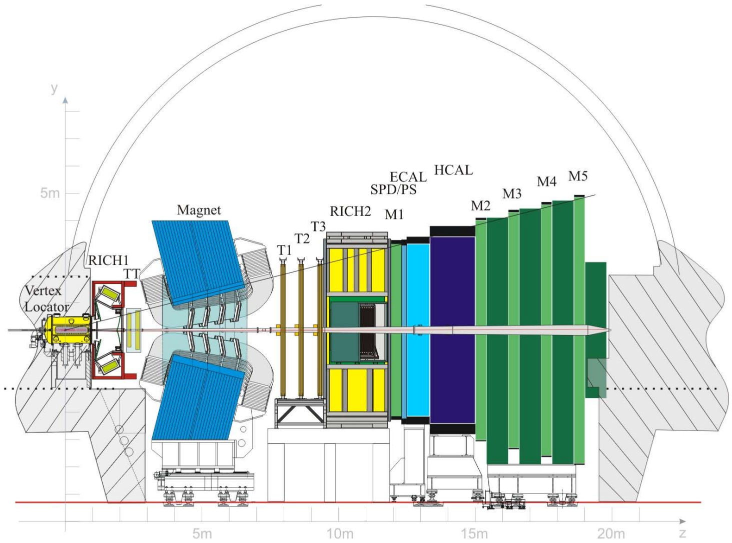


- Located at point 8 of the LHC
- General-purpose detector in the forward region
- Specialised in studying *b*- and *c*-decays

- Instrumented in the forward region to exploit forward-production of *c*- and *b*-hadrons

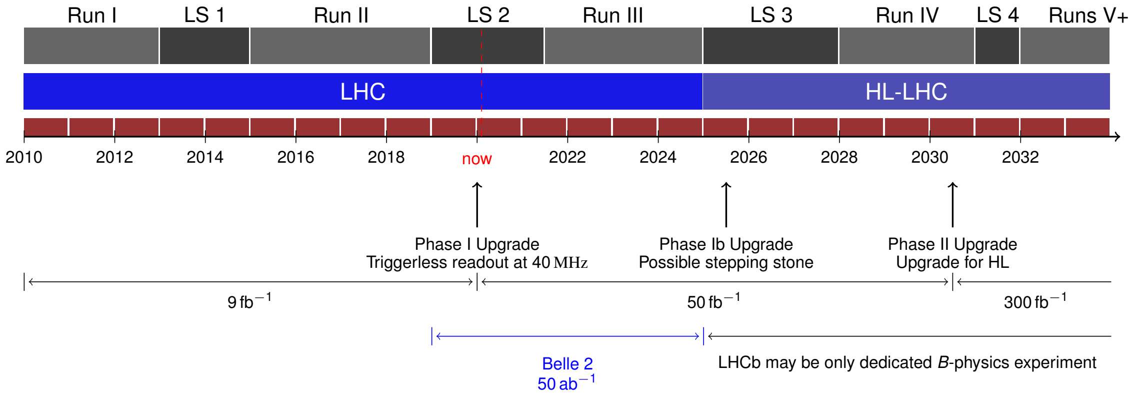


The LHCb detector



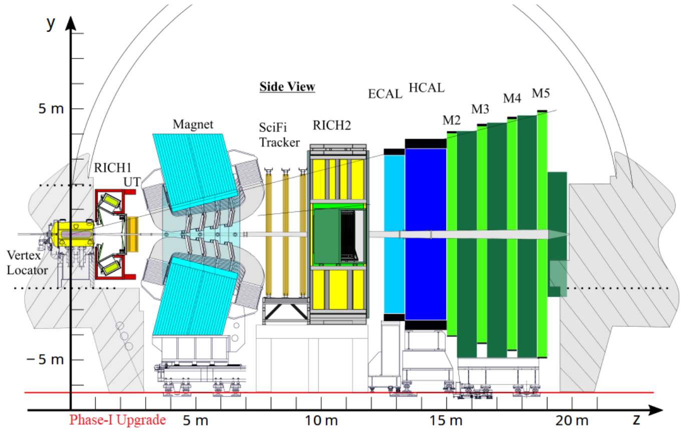
- Instrumentation in the forward region ($2 < \eta < 5$)
- Excellent secondary vertex reconstruction
- Precise tracking before and after magnet
- Good PID separation up to $\sim 100\,\mathrm{GeV}/c$

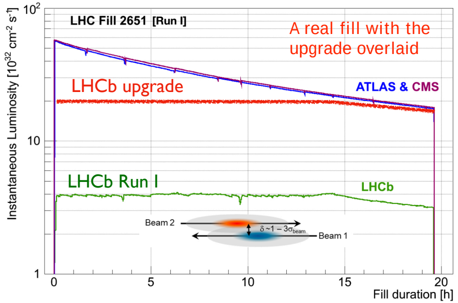
LHCb timeline



The LHCb detector: Run III upgrade

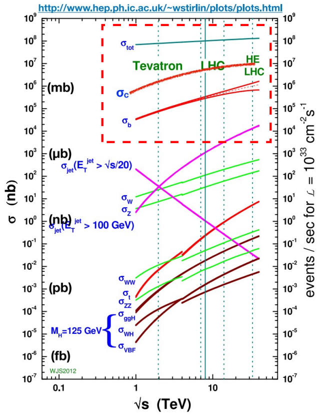
CERN-LHCC-2012-007





- New vertex locator
- New tracking (UT, SciFi)
- New front-end electronics
- Run at 5$\times$ higher luminosity

Challenges in Run III

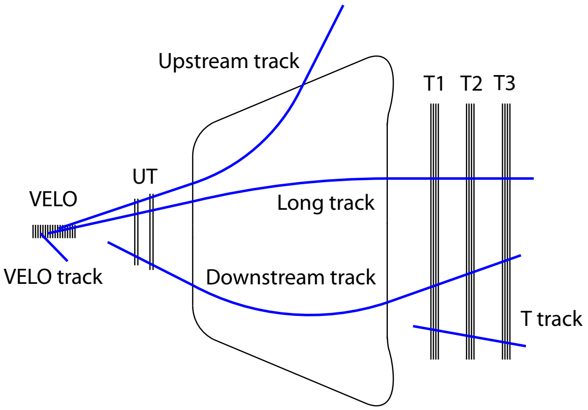


- At increased luminosity, charm (beauty) in 24 % (2 %) of bunch crossings
  - Cannot write out charm at 7 MHz
- Trigger must distinguish signal from less-interesting signal as well as from background
- No longer feasible to have first trigger based on calorimeters and muon detectors alone
- Need as much information about an event as soon as possible $\rightarrow$ run tracking

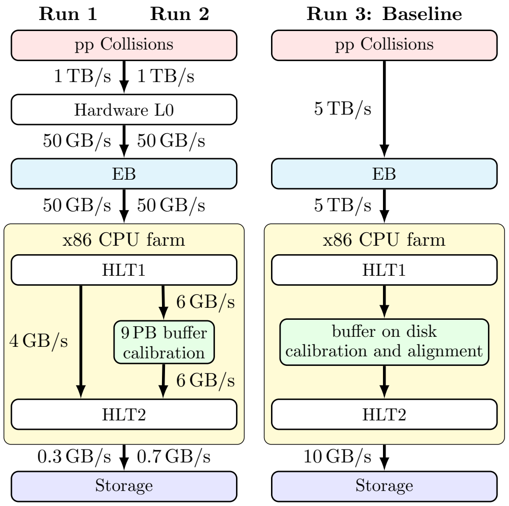
$\longrightarrow$

Tracking at LHCb

- Tracking requires readout of several sub-detectors
- Tracks must be extrapolated between VELO, UT and SciFi (T1–T3)
- Also match to muon stations for muon particle ID

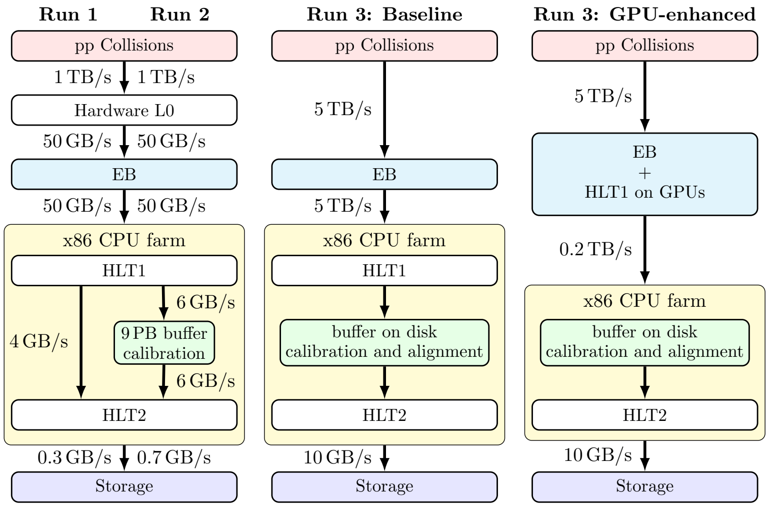


LHCb trigger in Run III



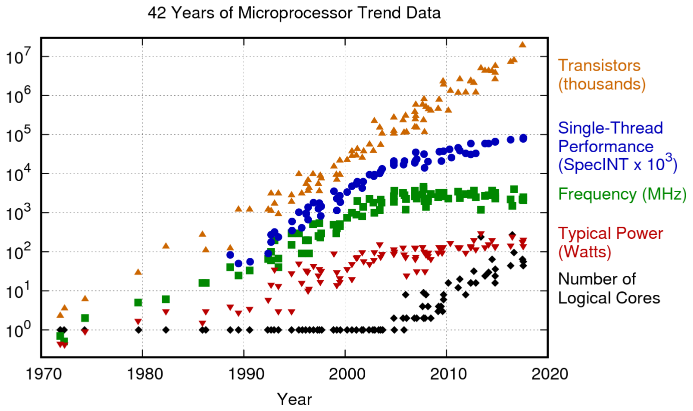
- Hardware trigger to be removed from Run III
- HLT1 software trigger must perform at 30$\times$ higher rate with 5$\times$ the pileup
- Buffer will reduce from $\mathcal{O}$(weeks)$\rightarrow$ $\mathcal{O}$(days)
- Significant increase in data transfer rates
- New trigger setup offers up to $\sim$ 10$\times$ efficiency improvement for some physics channels

Alternative trigger for Run III?



- Option to move to a GPU-based HLT1 with GPUs installed on the Event Builder servers
- Free up full CPU farm for HLT2 and save on networking between event builders and CPU farm
- Demonstrated technical feasibility
- Decision due next few months

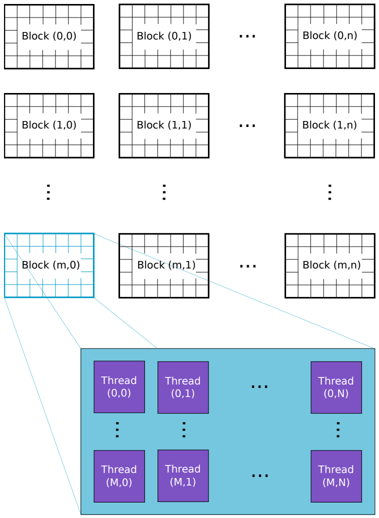
Why GPUs?



Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2017 by K. Rupp

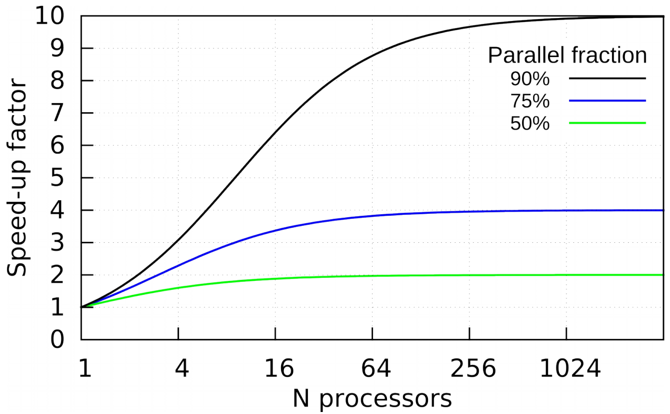
- Moore's law still holds but single-thread performance has levelled off
- Gains now to be made through parallelisation
- GPUs specialised for massively parallel operations (100s–1000s of cores)

GPU architecture



- Kernel executed in many threads
- Threads run same algorithm on different parts of the data
- Threads arranged within blocks within a grid
- Threads within a block share memory and synchronised
- Block and grid dimensions optimised for each kernel

Amdahl's law



Speedup $= \frac{1}{S+P/N}$

- $S =$ sequential fraction
- $P =$ parallel fraction
- $N =$ number of processors

Significant gains require large fraction of sequence to be parallelised

External constraints

- $\sim$ 250 Event Builder servers $\rightarrow$ $\sim$ 500 GPUs
- 16 – 32 GB/s PCIe rate $\rightarrow$ sufficient for 5 TB/s input
- Small raw event size $\sim$ 100 kB $\rightarrow$ process several 1000 events at once per GPU

The Allen project

arXiv:1912.09161

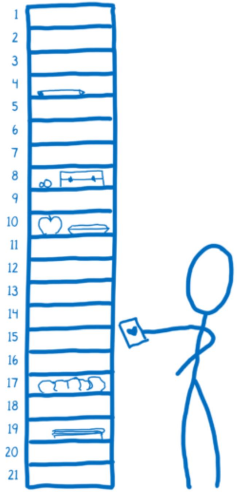
- Generic configurable framework for GPU-based execution of an algorithm sequence
- Stand-alone software package: https://gitlab.cern.ch/lhcb/Allen
- Dependencies: C++17 compiler, CUDA v10.2, boost, ZeroMQ
- Built-in validation and monitoring (requires ROOT)
- Process thousands of events in a single sequence
  - Opportunity for massive parallelisation
- Cross-platform compatibility with CPU architectures
- Named for Frances E. Allen
- Implement HLT1 on GPUs

Algorithm sequeunce

- Multiple “streams” on each GPU process “slices” of $\mathcal{O}(1000)$ events
- Single transfer of data to GPU device
  - Data passed to device
  - All algorithms executed in order
  - Results passed back to the host
- Configurable sequences at compile time
- Configurable algorithms at run time via JSON

Memory management

- No dynamic memory allocation
- Data dependencies and memory assignments resolved at compile time
- Host and device memory handled by custom memory manager
  - All memory allocated on startup
  - Assigned on demand
- Failsafe mechanism to sub-divide data slices with unusually large memory requirements and pass through problematic events

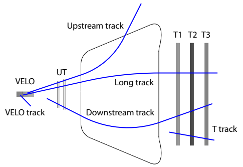


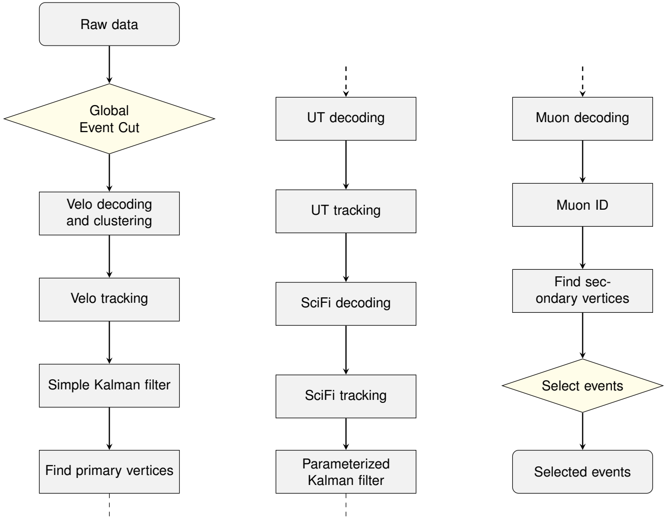
Integration

- I/O performed asynchronously by separate CPU thread
    - Input data banks may be read from binary files or decoded from MDF or MEP formats
    - Only selected events sent to output
    - Selection decisions and reconstructed objects added to output data
- Monitoring also performed in dedicated thread

HLT1

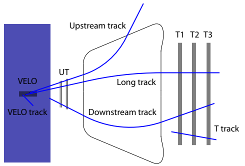
- HLT1 involves decoding, clustering and track reconstruction for all tracking subdetectors
- Algorithms also perform Kalman filter and trigger selection
- All stages of the process may be parallelised

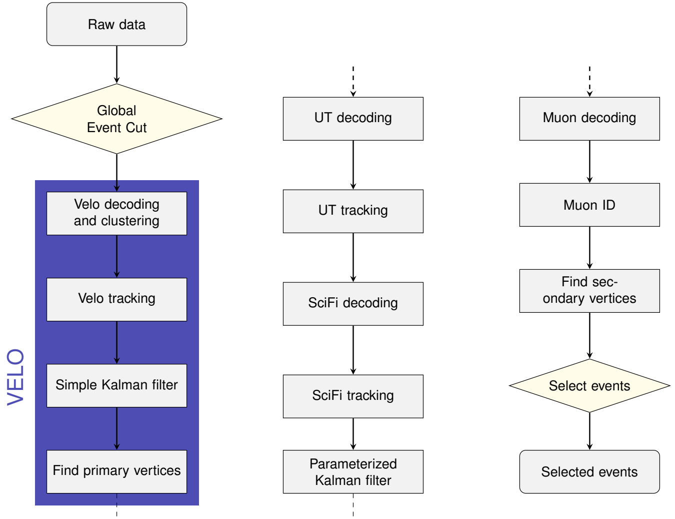




HLT1

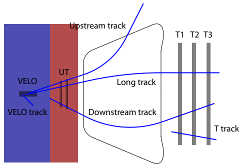
- HLT1 involves decoding, clustering and track reconstruction for all tracking subdetectors
- Algorithms also perform Kalman filter and trigger selection
- All stages of the process may be parallelised

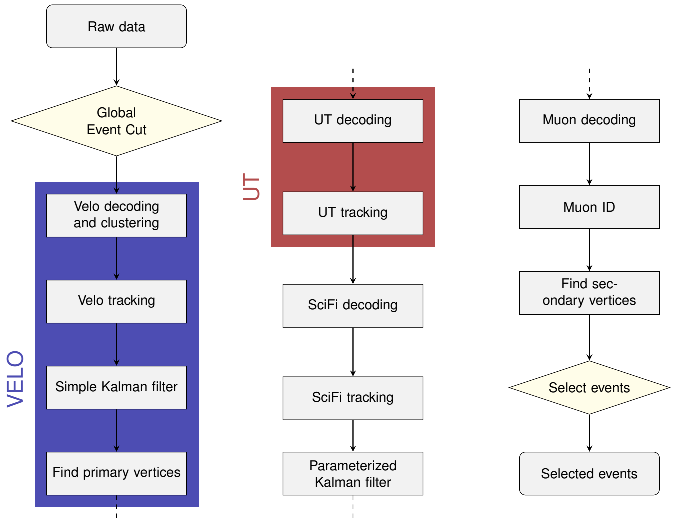




HLT1

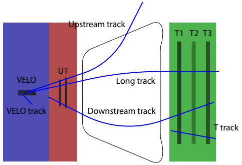
- HLT1 involves decoding, clustering and track reconstruction for all tracking subdetectors
- Algorithms also perform Kalman filter and trigger selection
- All stages of the process may be parallelised

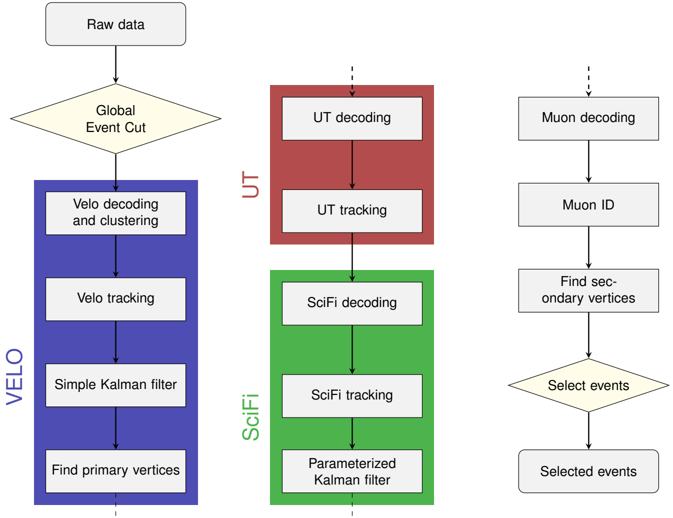




HLT1

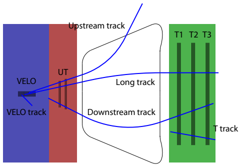
- HLT1 involves decoding, clustering and track reconstruction for all tracking subdetectors
- Algorithms also perform Kalman filter and trigger selection
- All stages of the process may be parallelised

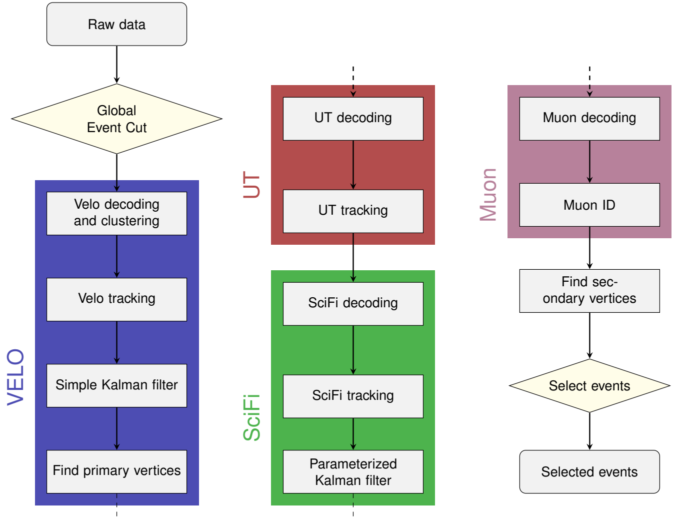




HLT1

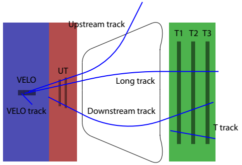
- HLT1 involves decoding, clustering and track reconstruction for all tracking subdetectors
- Algorithms also perform Kalman filter and trigger selection
- All stages of the process may be parallelised

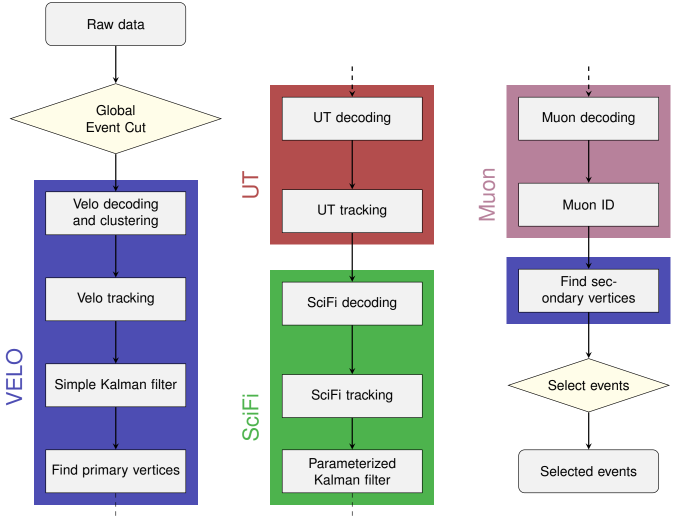




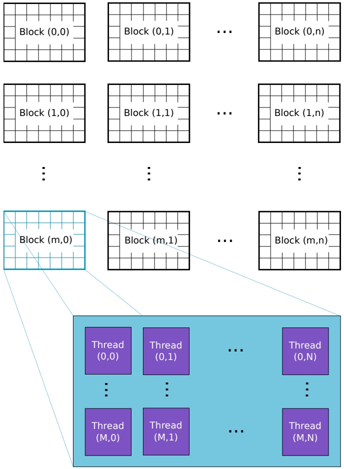
HLT1

- HLT1 involves decoding, clustering and track reconstruction for all tracking subdetectors
- Algorithms also perform Kalman filter and trigger selection
- All stages of the process may be parallelised



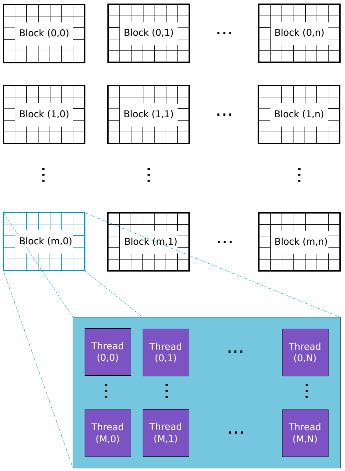


Parallelisation



Run each event in one block

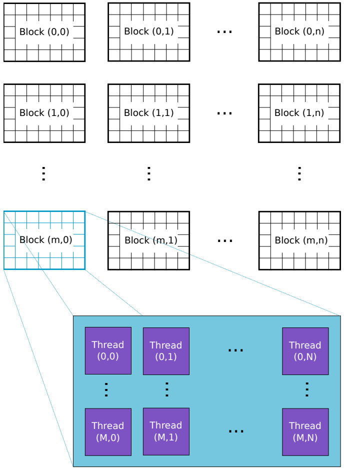
Parallelisation



Run each event in one block

Decoding $\rightarrow$ parallelise by readout unit

Parallelisation

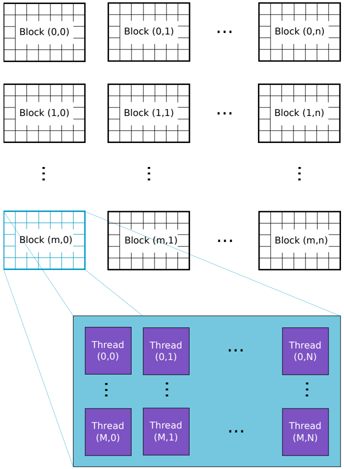


Run each event in one block

Decoding $\rightarrow$ parallelise by readout unit

Clustering $\rightarrow$ parallelise in (overlapping) detector regions

Parallelisation



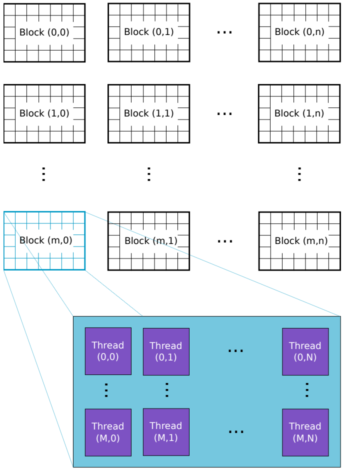
Run each event in one block

Decoding → parallelise by readout unit

Clustering → parallelise in (overlapping) detector regions

Tracking → parallelise by track

Parallelisation



Run each event in one block

Decoding → parallelise by readout unit

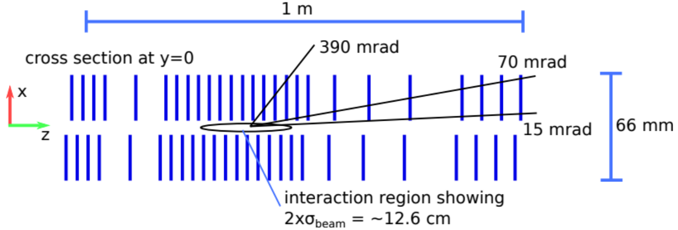
Clustering → parallelise in (overlapping) detector regions

Tracking → parallelise by track

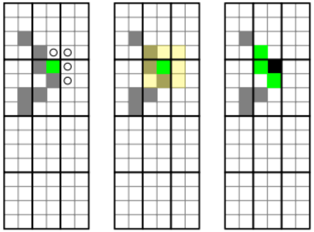
Vertexing → parallelise by combination

Example: velo clustering

26 planes of silicon pixel detectors

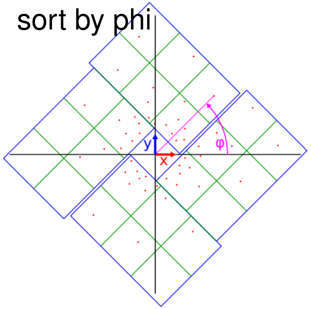


Clustering with bit masks



Example: velo tracking

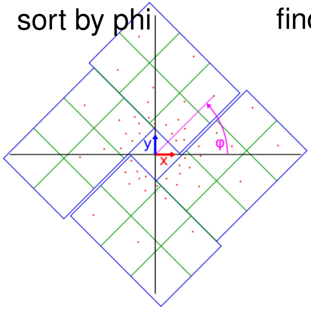
sort by phi

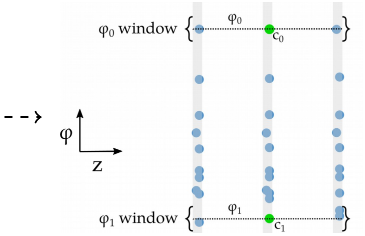


Example: velo tracking

sort by phi

find windows in phi





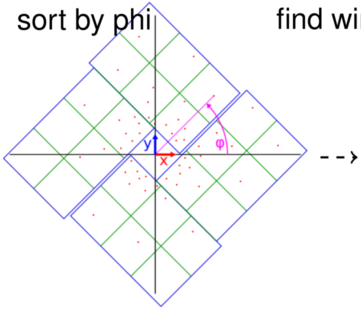
Example: velo tracking

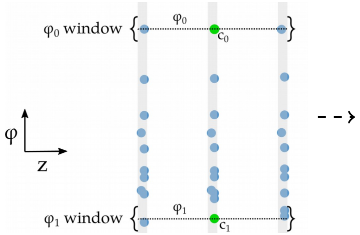
IPDPSW (2019) 00118

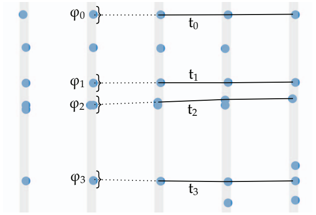
sort by phi

find windows in phi

forward tracks



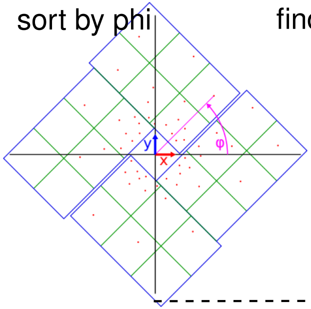




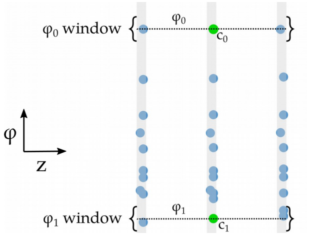
Example: velo tracking

IPDPSW (2019) 00118

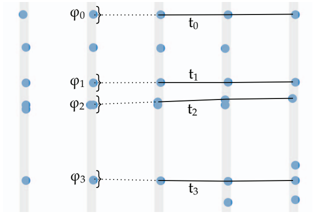
sort by phi

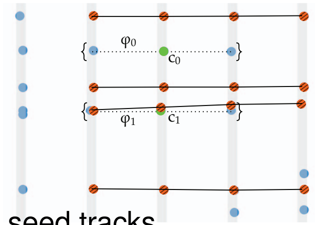


find windows in phi



forward tracks



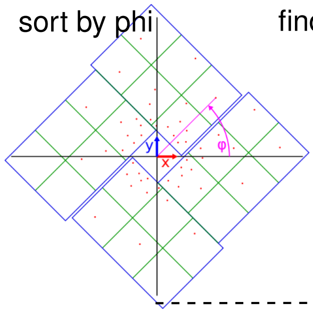


seed tracks

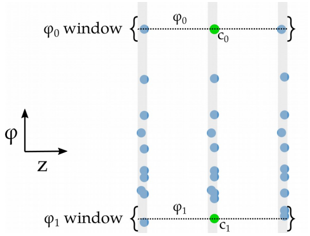
Example: velo tracking

IPDPSW (2019) 00118

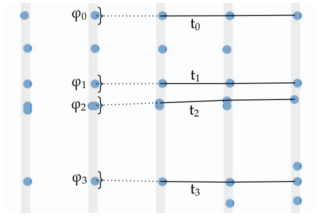
sort by phi

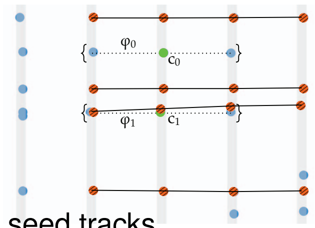


find windows in phi

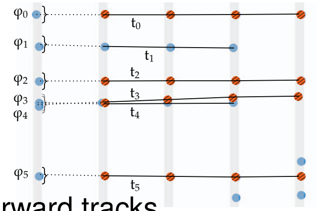


forward tracks





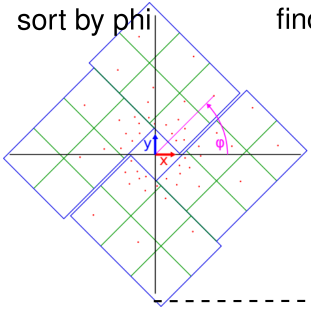
seed tracks



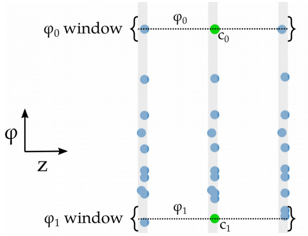
forward tracks

Example: velo tracking

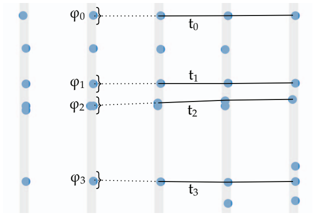
sort by phi

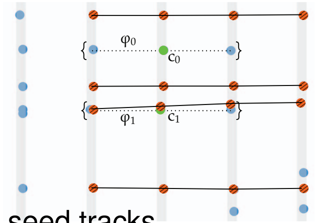


find windows in phi

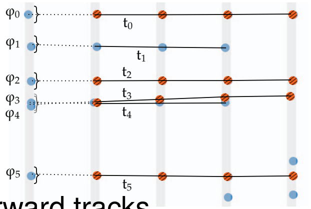


forward tracks

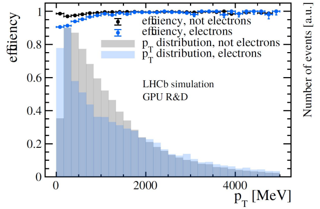




seed tracks



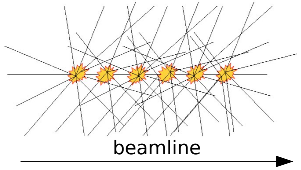
forward tracks

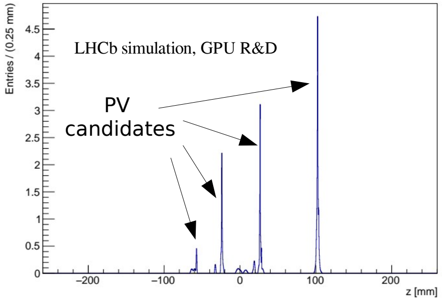


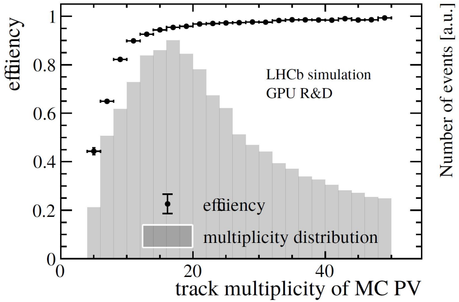
Example: velo vertexing

Record *z* of closest approach to beamline for each track

Peaks in distribution identify PVs



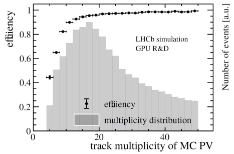




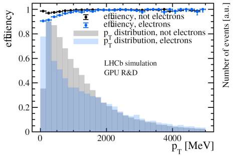
Allen selection ingredients

LHCB-FIGURE-2019-009

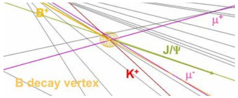
Primary vertices



Tracks



Secondary vertices

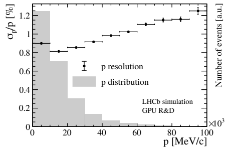


Selections

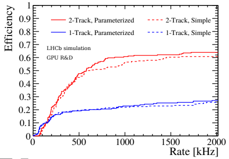
- One track
- Two tracks
- Single muon
- Two muons (displaced)
- Two muons (high-mass)

...

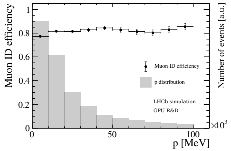
Momentum



Impact parameter



Muon ID



Allen performance

LHCB-FIGURE-2019-009

Trigger	Rate [kHz]
1-Track	215 ± 18
2-Track	659 ± 31
High-p_T muon	5 ± 3
Displaced dimuon	74 ± 10
High-mass dimuon	134 ± 14
Total	999 ± 38

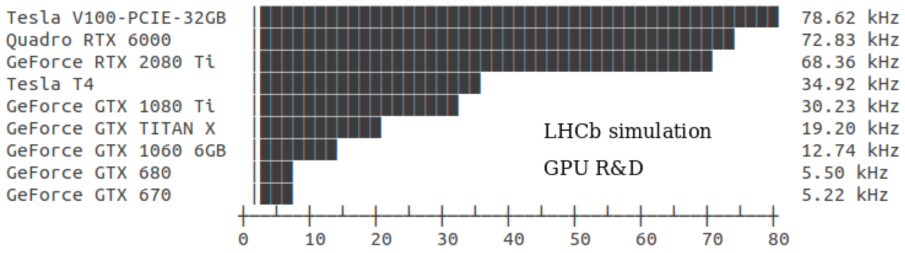
- Total rate reduced $30 \rightarrow 1\,\mathrm{MHz}$
- Physics performance consistent with x86 baseline

Signal	GEC	TIS -OR- TOS	TOS	$\mathrm{GEC} \times \mathrm{TOS}$
$B^0 \rightarrow K^{*0}\mu^+\mu^-$	89 ± 2	91 ± 2	89 ± 2	79 ± 3
$B^0 \rightarrow K^{*0}e^+e^-$	84 ± 3	69 ± 4	62 ± 4	52 ± 4
$B_s^0 \rightarrow \phi\phi$	83 ± 3	76 ± 3	69 ± 3	57 ± 3
$D_s^+ \rightarrow K^+K^-\pi^+$	82 ± 4	59 ± 5	43 ± 5	35 ± 4
$Z \rightarrow \mu^+\mu^-$	78 ± 1	99 ± 0	99 ± 0	77 ± 1

GEC = global event cut, TIS = trigger independent of signal, TOS = trigger on signal

Allen throughput

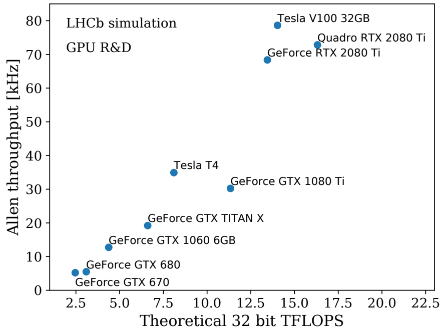
LHCB-FIGURE-2019-009



- Full HLT1 algorithm can be run on $\sim$ 500 current GPUs
- Buy GPUs instead of networking

Allen throughput

LHCB-FIGURE-2019-009



- Performance scales with GPU so can expect more from 2021 GPUs
  - Not yet limited by Amdahl's law
  - Potential to perform more tasks within HLT1

Summary

- Allen project offers a GPU-implementation of LHCb HLT1
- Full track reconstruction and selection performed
- Generic framework allows for configurable algorithm sequence
- Feasibility for possible use in Run III already demonstrated