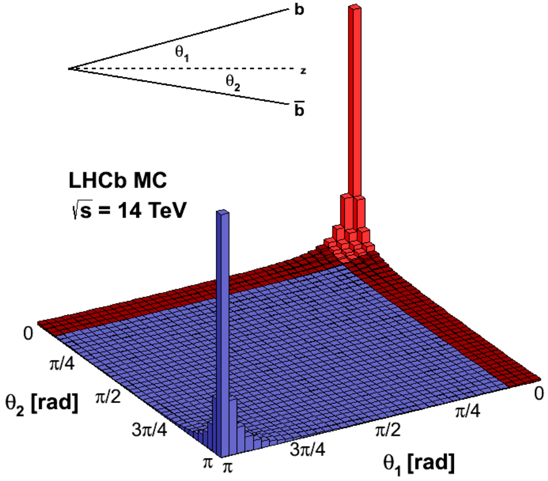
The Allen heterogeneous software framework and possible applications in ePIC

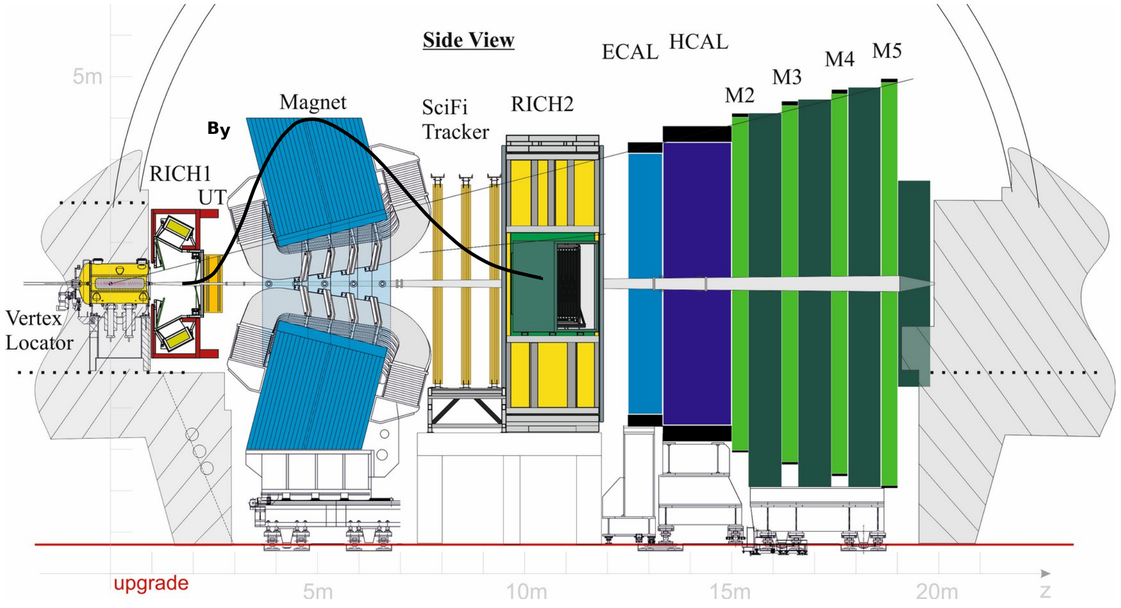
Dorothea vom Bruch
CPPM, Aix-Marseille Université, Marseille, CNRS/IN2P3

ePIC France Workshop
October 10th 2024

LHCb experiment

General purpose detector in the forward region specialized in beauty and charm hadrons

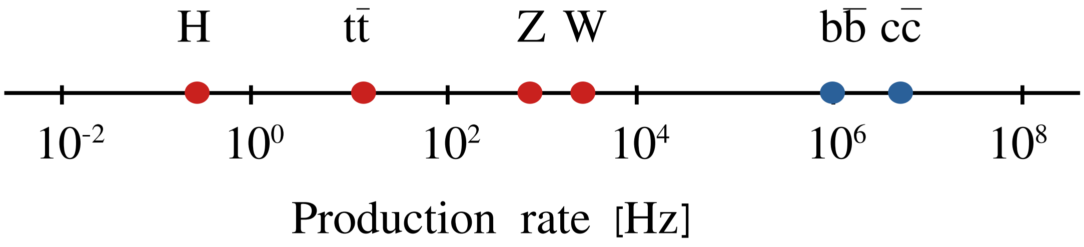


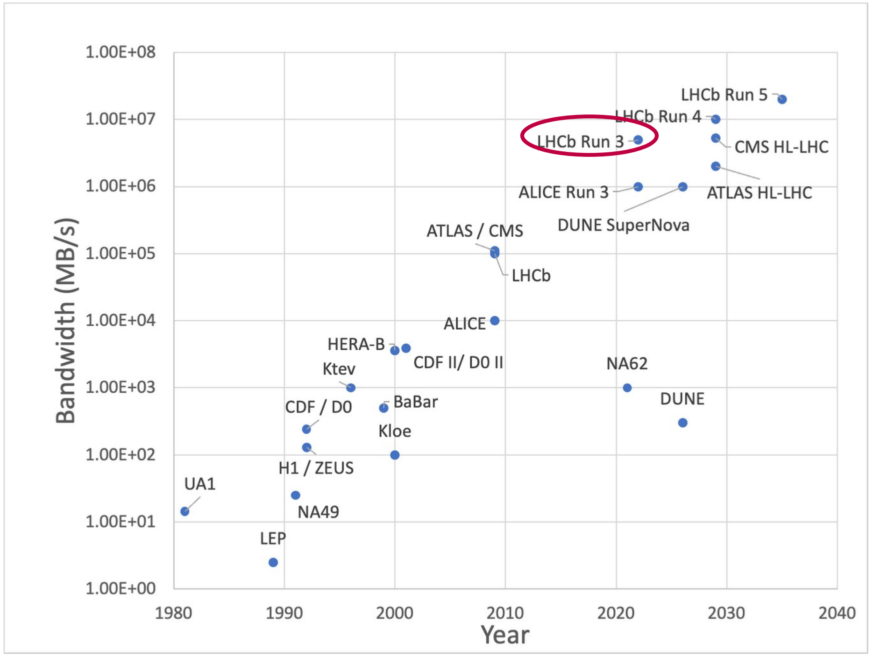


LHCb trigger challenge

$\mathcal{L} = 2\times10^{34}\ cm^{-2}s^{-1}$ (ATLAS/CMS) sqrt(s) = 13.6 TeV

$\mathcal{L} = 2\times10^{33}\ cm^{-2}s^{-1}$ (LHCb)

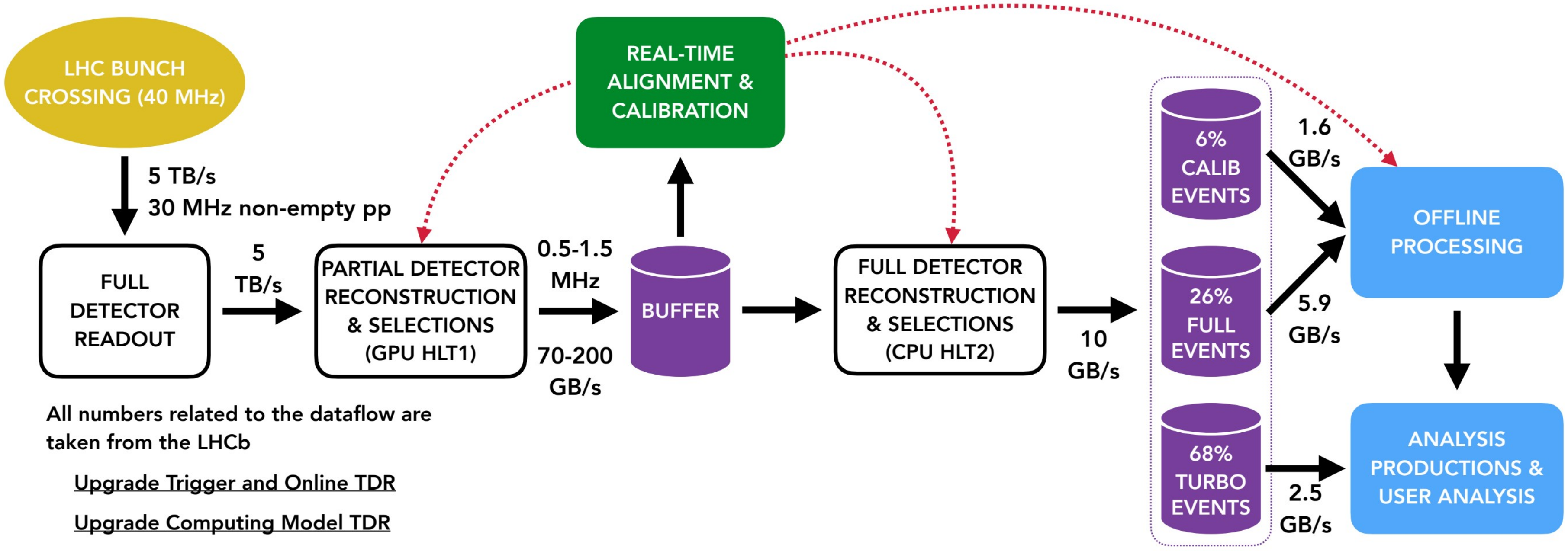




Key signature is a secondary vertex with significant transverse momentum and displacement from the pp collision
→ Charged particle reconstruction at 30 MHz in full detector is necessary

A. Cerri - University of Sussex

LHCb: Software-only real-time analysis since 2022

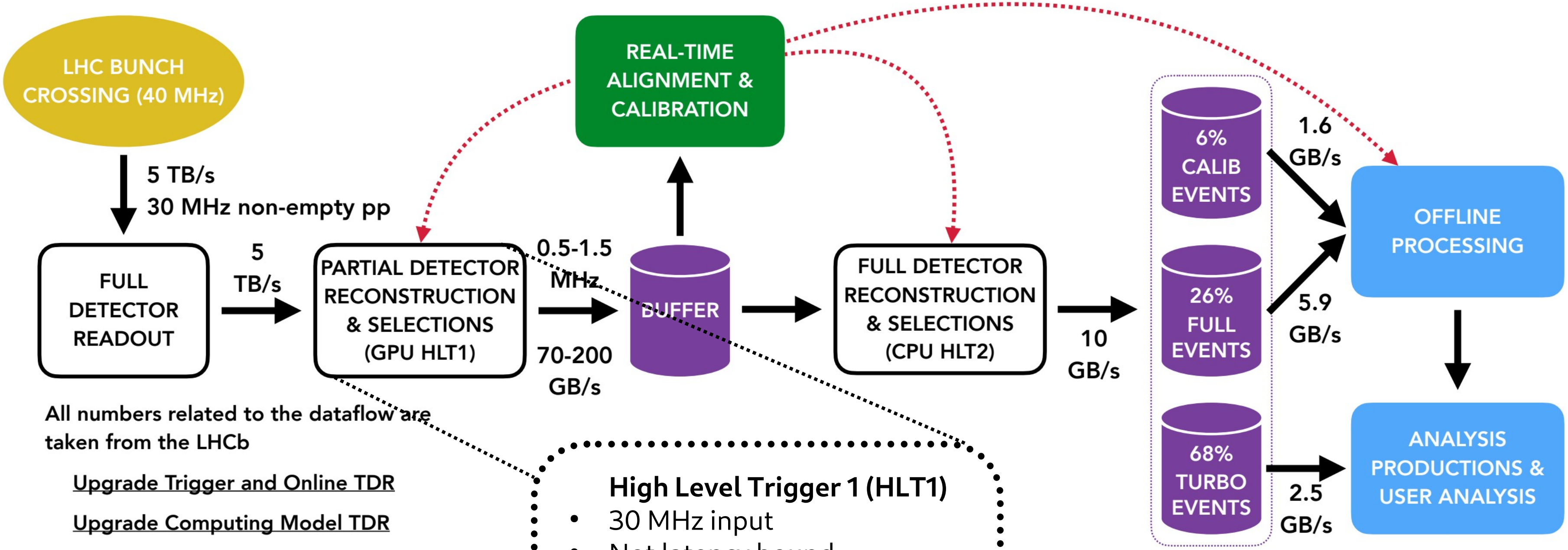


All numbers related to the dataflow are taken from the LHCb

Upgrade Trigger and Online TDR

Upgrade Computing Model TDR

LHCb: Software-only real-time analysis since 2022



All numbers related to the dataflow are taken from the LHCb

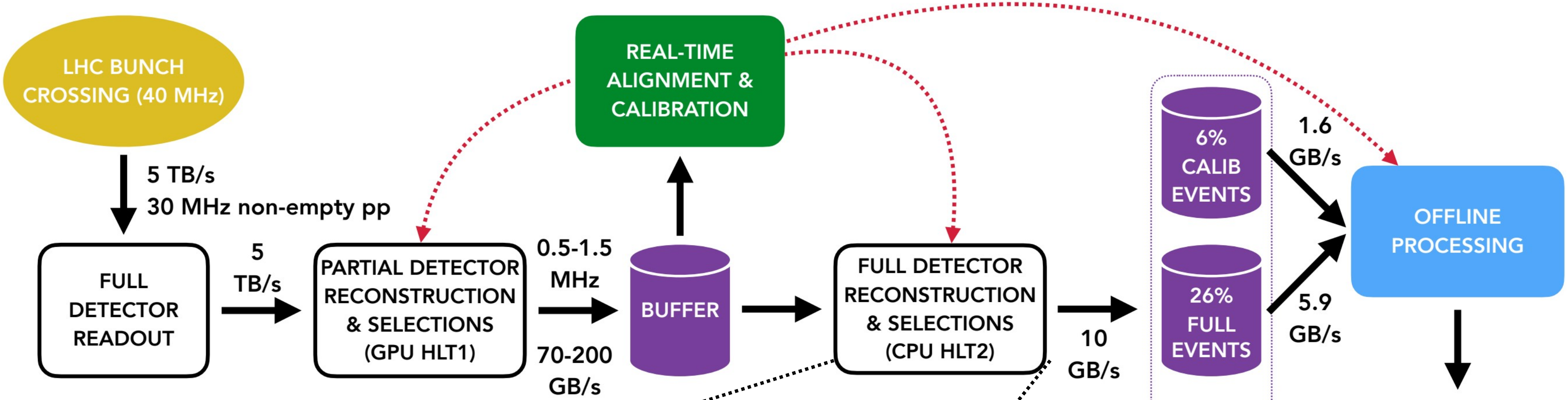
Upgrade Trigger and Online TDR

Upgrade Computing Model TDR

High Level Trigger 1 (HLT1)

- 30 MHz input
- Not latency bound
- 125k lines of code
- O(100) algorithms to maintain by O(10) developers
- Processed by Allen software
- On O(300) GPUs

LHCb: Software-only real-time analysis since 2022



All numbers related to the dataflow are taken from the LHCb

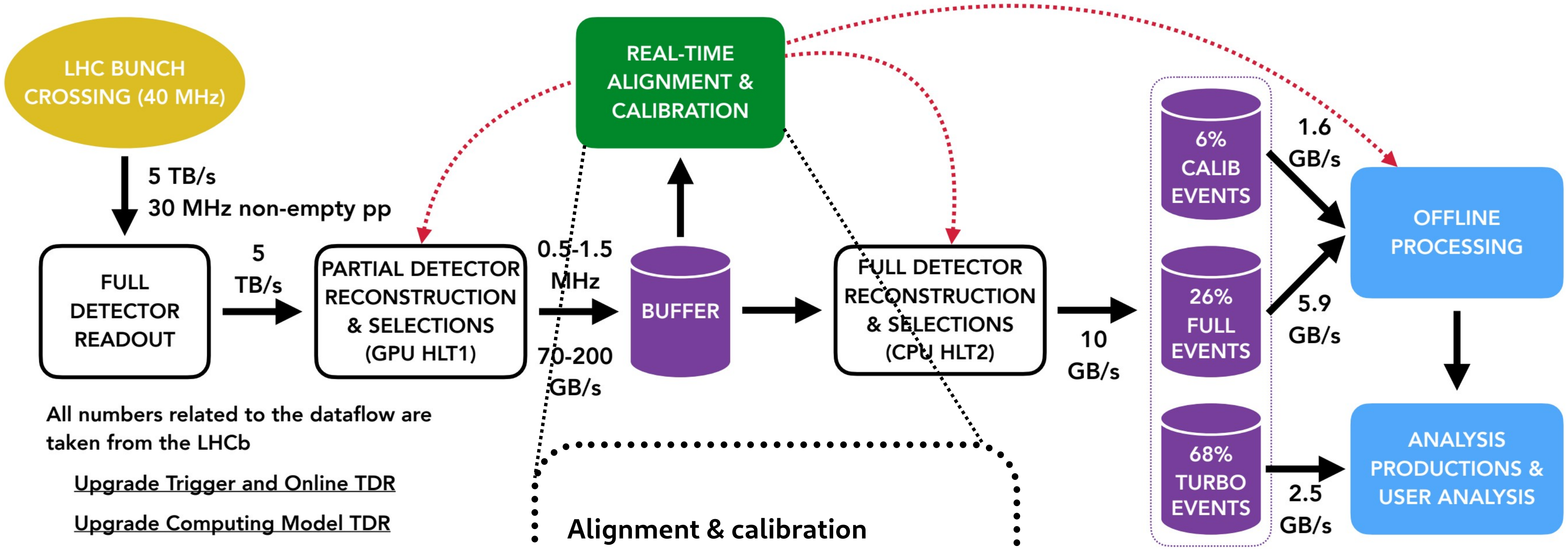
Upgrade Trigger and Online TDR

Upgrade Computing Model TDR

High Level Trigger 2 (HLT2)

- 1 MHz input
- Not latency bound
- 2 million lines of code
- O(2000) algorithms to maintain by O(100) developers
- Processed by Moore software

LHCb: Software-only real-time analysis since 2022



All numbers related to the dataflow are taken from the LHCb

Upgrade Trigger and Online TDR

Upgrade Computing Model TDR

Alignment & calibration
Determine detector position based on reconstructed quantities up to tens of micrometers

LHCb: Software-only real-time analysis since 2022

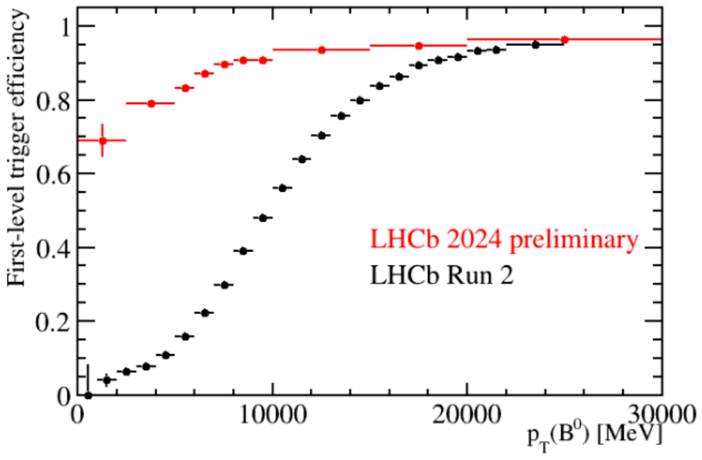
All numbers related to the dataflow are taken from the LHCb

[Upgrade Trigger and Online TDR]

[Upgrade Computing Model TDR]

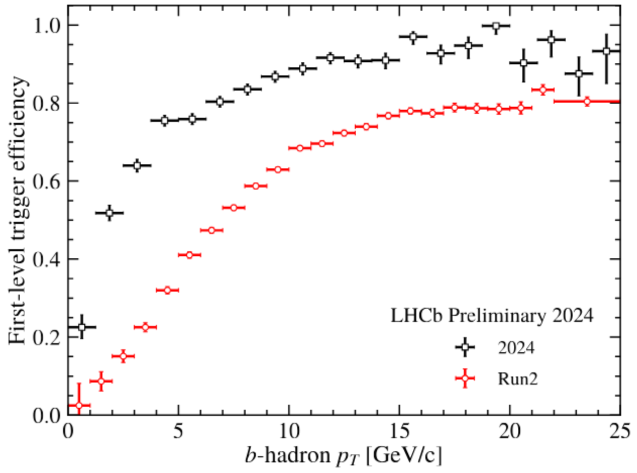
LHCb’s trigger performance in 2024

B -> Dπ

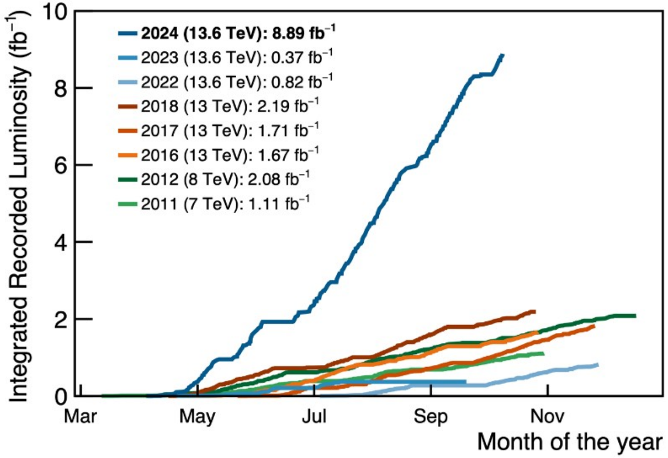


LHCb-Figure-2024-014

B -> Kee

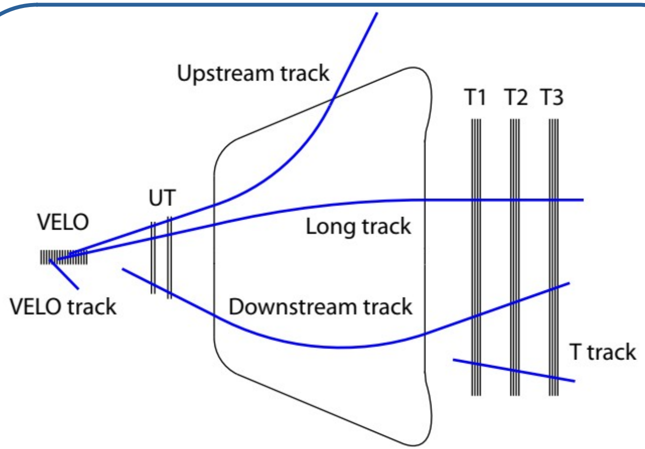


LHCb-Figure-2024-007

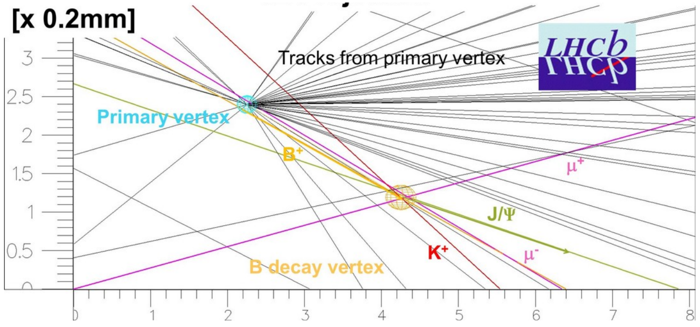


What do we reconstruct at LHCb?

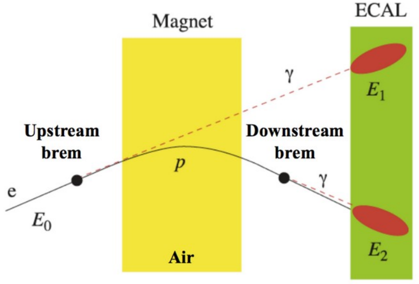
Tracks



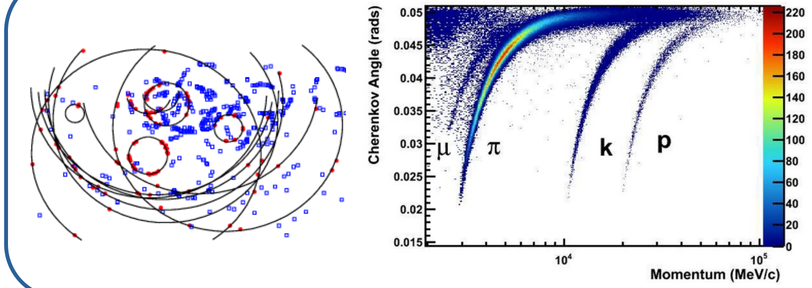
Vertices



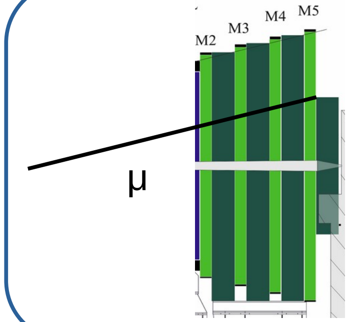
Electrons



Cherenkov rings



Muons

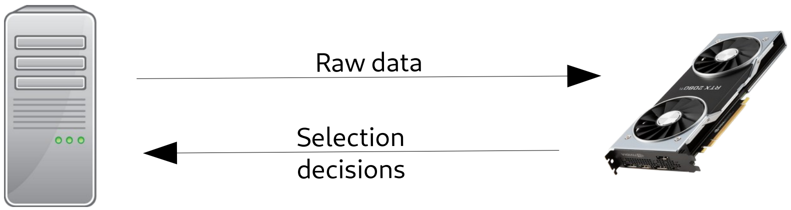


How does HLT1 map to GPUs?

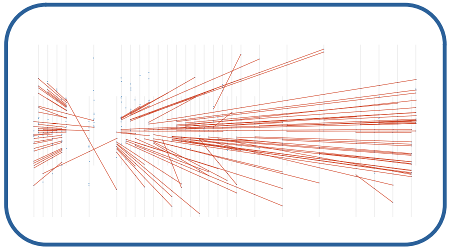
Characteristics of LHCb HLT1	**Characteristics of GPUs**
Intrinsically parallel problem: - Run events in parallel - Reconstruct tracks in parallel	Good for - Data-intensive parallelizable applications - High throughput applications
Huge compute load	Many TFLOPS
Full data stream from all detectors is read out → no stringent latency requirements	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

Allen design principles

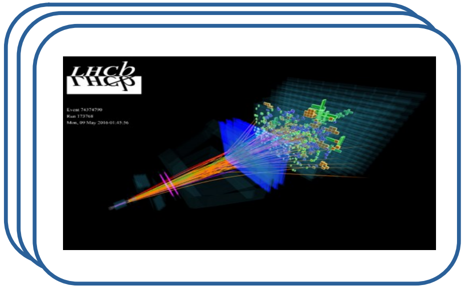
- Do all work on the GPU
  - Minimize copies to/from GPU
- Parallelize on multiple levels
- Maximize (GPU) algorithm performance
- Implement performant reconstruction algorithms
  - significantly faster/$ than on CPU
- Execution on multiple compute architectures possible
- Simple event model
  - Avoid dynamic allocations
  - Mostly SoA containers



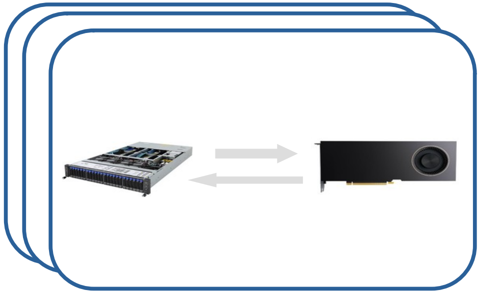
Intra collision: tracks, vertices,...



Proton collisions



Collision batches

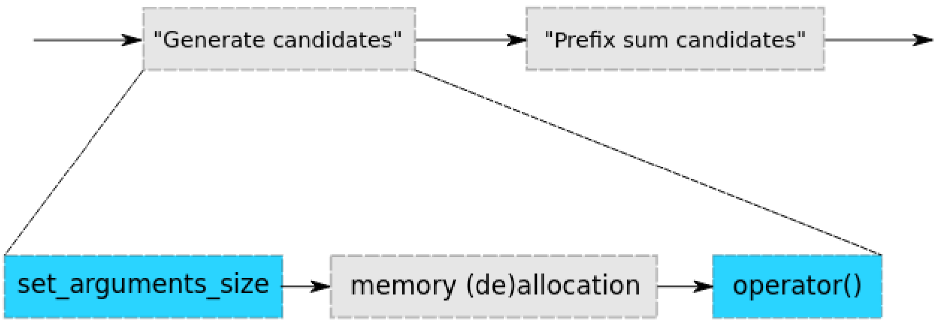


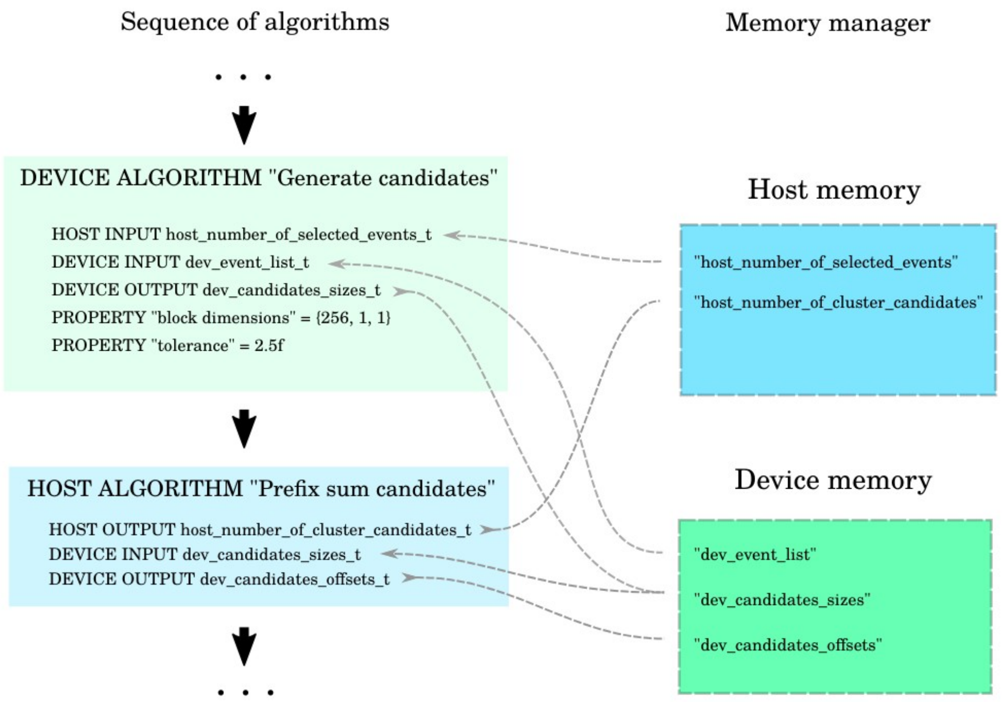
Allen software framework

- Named after Frances E. Allen
- Hosted on gitlab: gitlab.cern.ch/lhcb/Allen
- C++17 (soon 20), CUDA (12.X), HIP (5.X)
    - Algorithms implemented in CUDA
- Built with CMake and runs on CPU and GPU (NVIDIA and AMD)
- Standalone build and integrated with LHCb software stack
- Single precision everywhere (have not yet identified cases where double precision is needed, significant performance impact)
- Portability between architectures provided by macros and few simple guide lines
    - Allow dispatching to architecture-specific function implementations where needed for extra performance
- Custom memory manager
- Multi-event manager
- Algorithms configurable from python
- Geometry loaded from DD4Hep, converted to simple structs easily usable on GPU

Memory manager

- Memory allocations on the GPU are very slow
- Allocate chunk of memory at start of application
- Strong preference for “Count First, Write Later”
- Sequence uses data dependencies to track lifetime
  - Device Memory is released as soon as possible
- Host memory done analogously, but not released until after data is output from the application





Multi-event scheduler

- For efficient GPU execution, every algorithm processes many events
- Multi-Event Scheduler generates sequence of algorithms to be executed, considering all possible branching paths
- Running many events in parallel requires extending the “success” or “failure” of an algorithm execution to a vector
    - Implemented as vector of active elements, referred to as “mask”
    - May eg. look like: [0, 1, 4, 3] (event 2 is inactive)
- Masks are picked up by the scheduler and are required for the control flow
- Masks live on the device, or alternatively both host and device

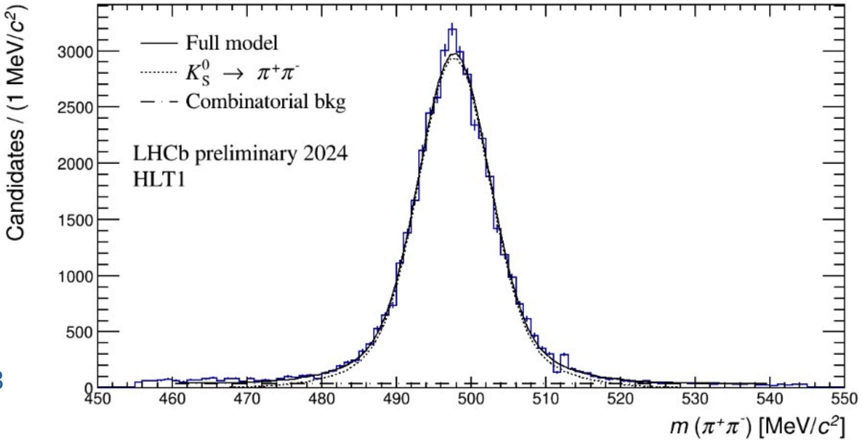
Python configuration

- Database of algorithms, inputs, outputs and properties built using code parsing with libclang
- Allow configuration of the sequence of algorithms/kernels
- Allow properties of algorithms to be set
- Multiple instances of an algorithm with separate inputs and outputs
- Configuration in Python using LHCb's PyConf package

```
seed_tracks = make_algorithm(
    seed_confirmTracks_t,
    name='seed_confirmTracks_{hash}',
    host_number_of_events_t=number_of_events["host_number_of_events"],
    dev_number_of_events_t=number_of_events["dev_number_of_events"],
    dev_scifi_hits_t=decoded_scifi["dev_scifi_hits"],
    dev_scifi_hit_count_t=decoded_scifi["dev_scifi_hit_offsets"],
    dev_seeding_tracksXZ_t=xz_tracks["seed_xz_tracks"],
    dev_seeding_number_of_tracksXZ_part0_t=xz_tracks[
        "seed_xz_tracks_part0"],
    dev_seeding_number_of_tracksXZ_part1_t=xz_tracks[
        "seed_xz_tracks_part1"],
    tuning_nhits=10,
    tuning_tol_chi2=100,
    tuning_tol=0.8,
)
```

Monitoring

- Ntuple writing for algorithm development
  - Supported only in CPU compilation
  - Any variable can be written to an Ntuple for further offline studies
  - Used for example to tune search windows for pattern recognition
- Histogram filling for monitoring during data-taking
  - Class provided that makes filling of histograms on the GPU easy
  - Mutliple instances of each algorithm are running → monitor aggregator merges all counters & histograms
  - Interfaced with LHCb’s monitoring infrastructure “Monet”



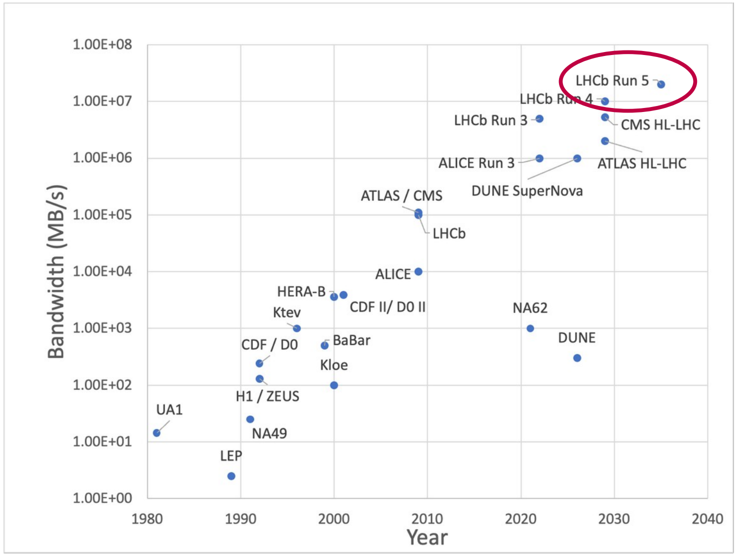
LHCb-Figure-2024-013

Future developments

- Future upgrade: process 25 TB/s
- Computing challenge will move from HLT1 to HLT2
- HLT2 algorithms are executed on CPU architecture as of now
- Only viable slution is to re-design HLT2 reconstruction algorithms for parallel architectures

- Evolve Gaudi and Allen software frameworks to combine best features of each
    - Gaudi: Framework used by LHCb & ATLAS
    - Medium throughput, CPUs only
- Separate LHCb-specific code from core framework code
- Explore processing on remote data centers

CERN Courier 03/2023



A. Cerri - University of Sussex

Possible use-case in ePIC

- Similarities between LHCb and ePIC DAQ (described in T. Wenaus' presentation at the ePIC Software & Computing meeting):
  - "All particles count" → reconstruct all particles w/o pre-selection in hardware
  - Full detector data available in counting room / at computing infrastructure
  - Data emerges from DAQ in "time frames" containing all subdetector data for hundreds of events

    → ideal for parallel processing
- Data-rate produced by ePIC comparable to LHCb in 2024
  - Allen could already cope with the processing with O(hundreds) of GPUs
- Allen provides
  - Core infrastructure for highly performant reconstruction code
  - Infrastructure for monitoring and development tools
  - Easy user interface via python

Development experience with Allen

- Allen was built with a core team of 1 postdoc, 1 SW engineer, 1 PhD student (computer science), one senior and several early career researchers contributing from time to time
- Took 4 years from nothing to a working system including the majority of LHCb's track reconstruction, vertex finding, HLT1 selections and some PID
- Systematic experience from LHCb:
  - 1-2 motivated PhD students can write a performant algorithm in ~6 months (with some support from core developers)
  - Takes another ~6 months to commission the algorithm for data-taking
- Seems plausible that Allen can be adapted for EIC reconstruction in a few years
- Happy to provide support on core Allen functionality if a demonstrator for the ePIC DAQ was to be tested

Resources

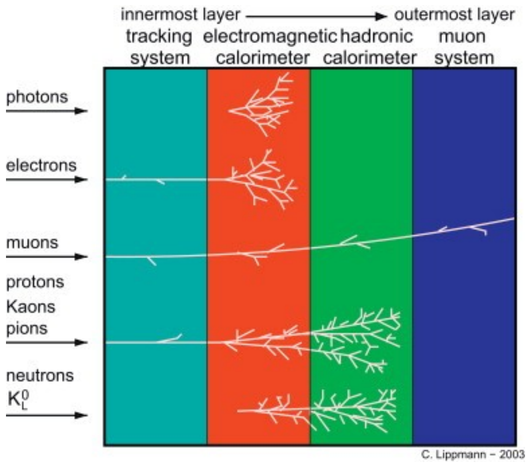
- Allen documentation: https://allen-doc.docs.cern.ch/index.html
- Allen publication: https://doi.org/10.1007/s41781-020-00039-7
- GPU High Level Trigger TDR: https://cds.cern.ch/record/2717938/files/LHCB-TDR-021.pdf
- Comparison of CPU and GPU implementations of LHCb Run 3 trigger: http://arxiv.org/pdf/2105.04031
- Evolution of the energy efficiency of LHCb's real-time processing: https://cds.cern.ch/record/2773126?ln=en
- Workshop organized in 11/2023 for future software framework developments: https://indico.cern.ch/event/1327907/overview

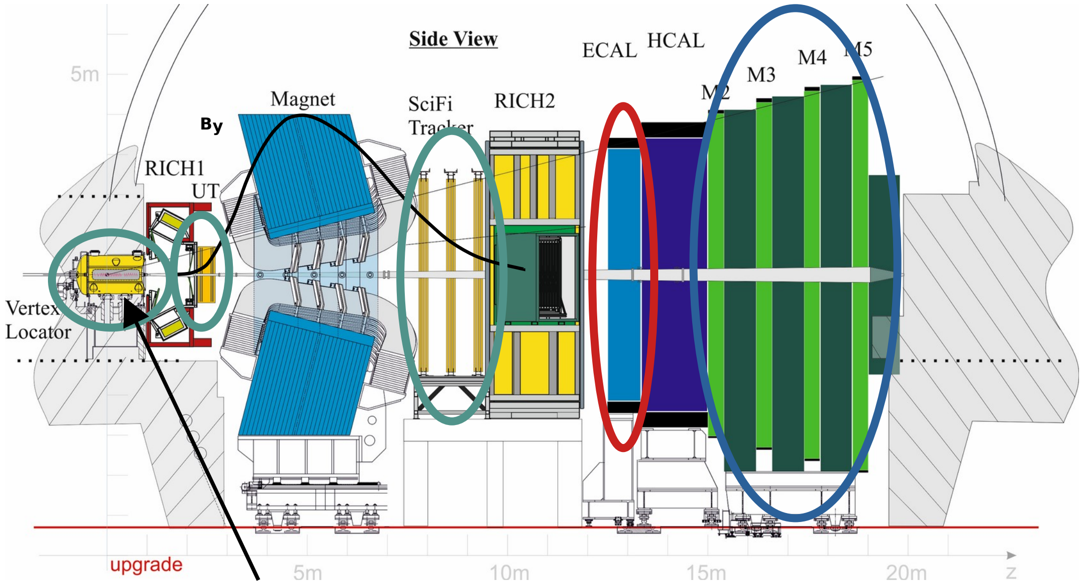
Backup

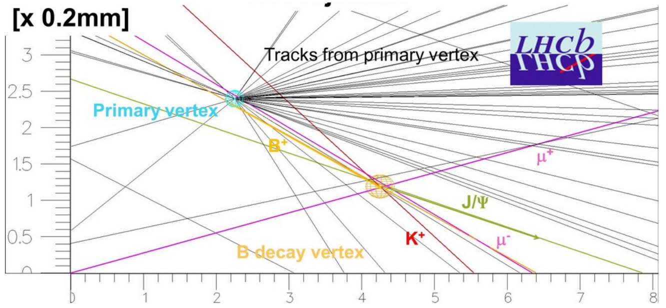
LHCb's first level real-time analysis: HLT1

High Level Trigger 1 (HLT1) tasks

- Decode binary payload of sub-detectors
- Reconstruct charged particle trajectories
- Identify electron and muon particles
- Reconstruct particle decay vertices
- Select proton-proton bunch collisions to store



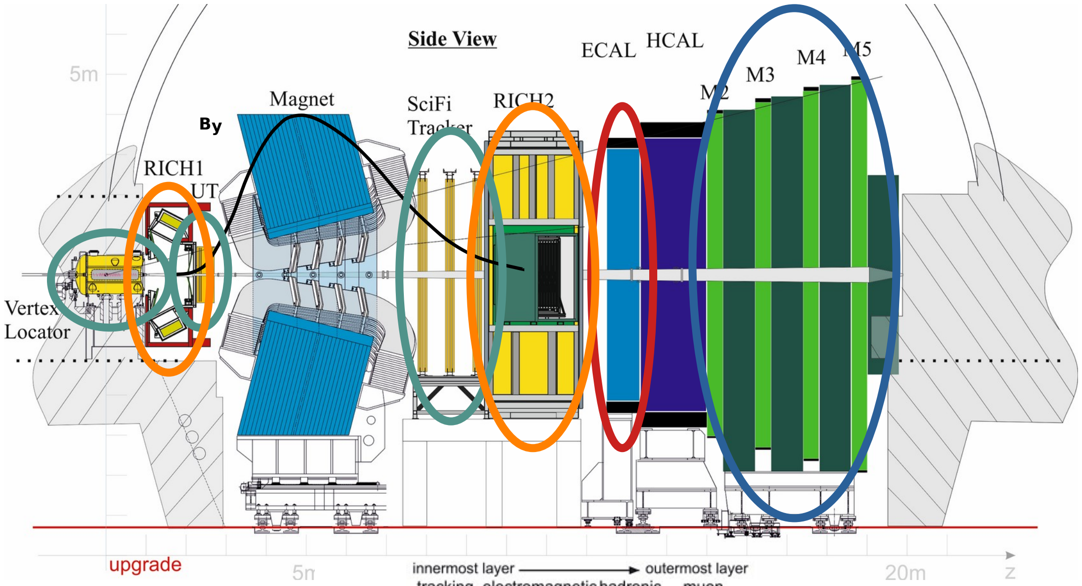


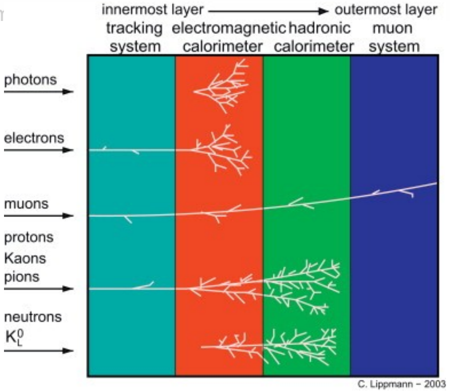


LHCb's second level real-time analysis: HLT2

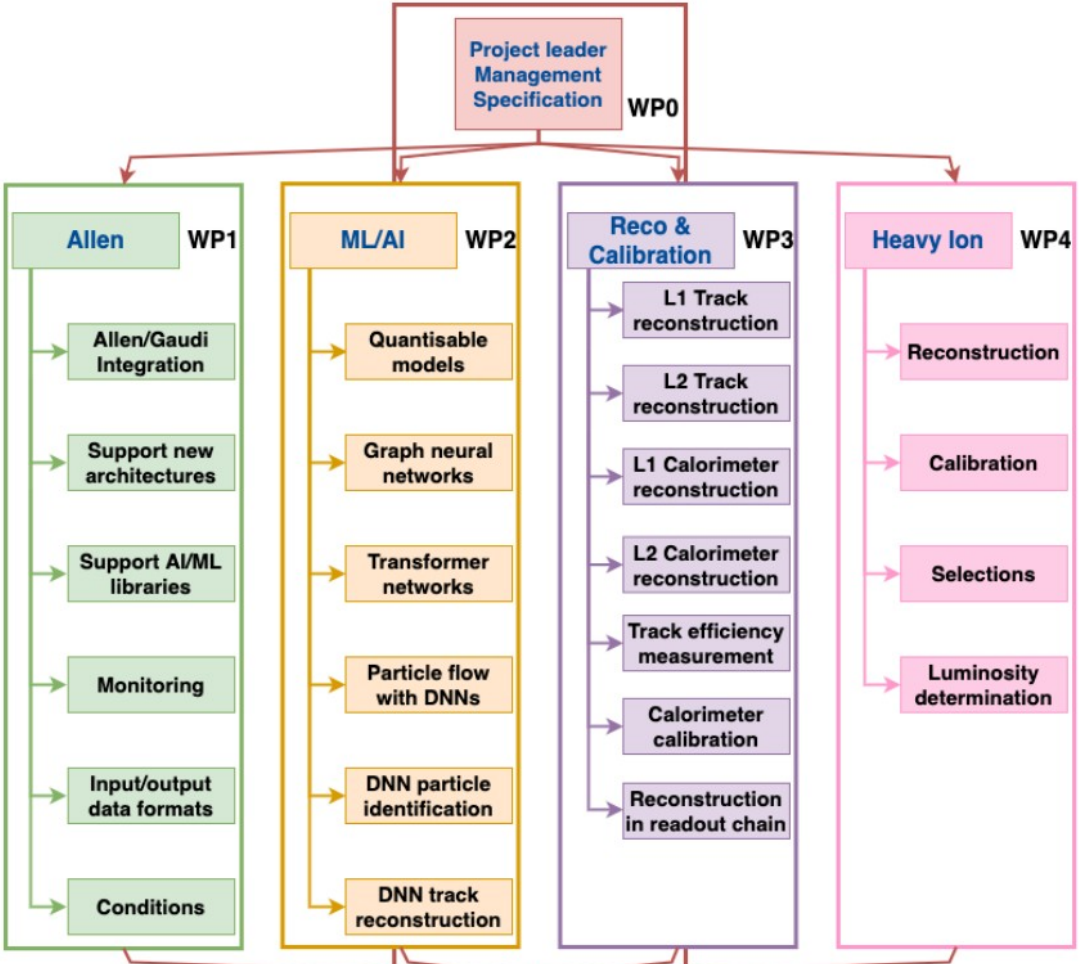
High Level Trigger 2 (HLT2) tasks

- Reconstruct charged particle trajectories with highest possible efficiency
- Fit particle trajectories with highest possible precision
- Identify electron and muon particles
- Identify hadron particles: pions, kaons, protons
- Reconstruct particle decay vertices
- Exclusively select particle decays of interest for offline analysis (around 1000 selections)
- Save only high-level objects for offline analysis

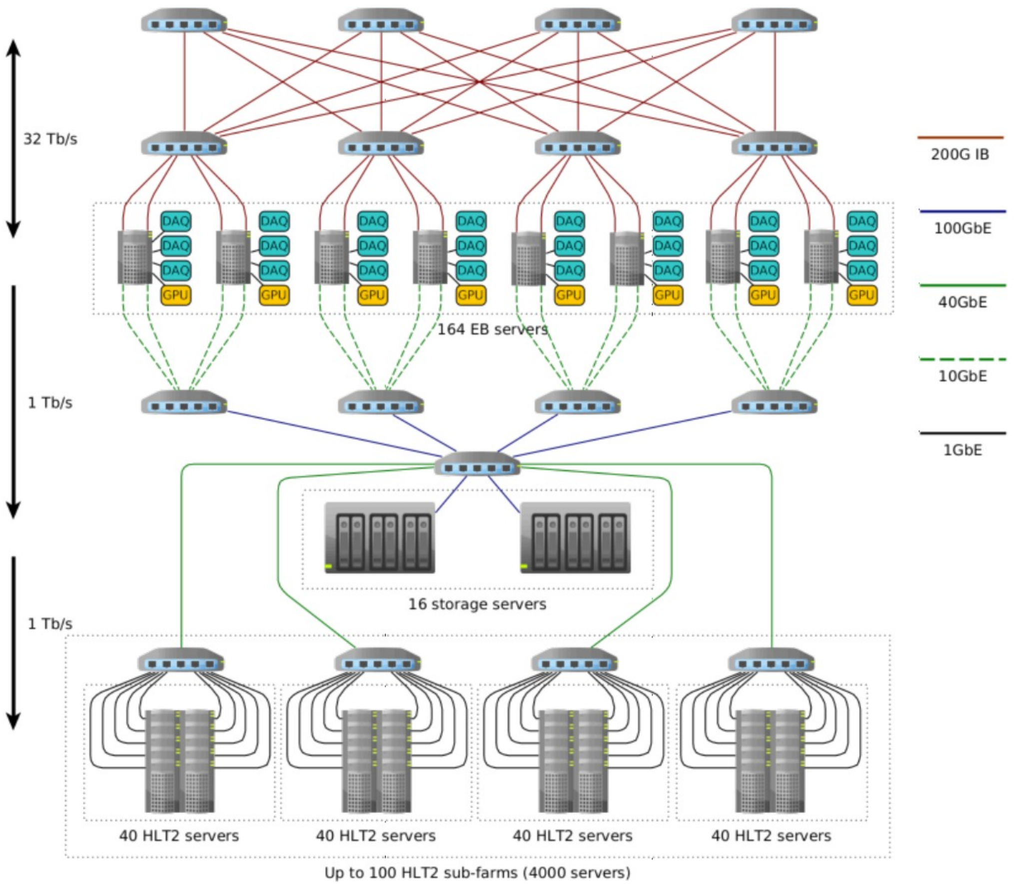




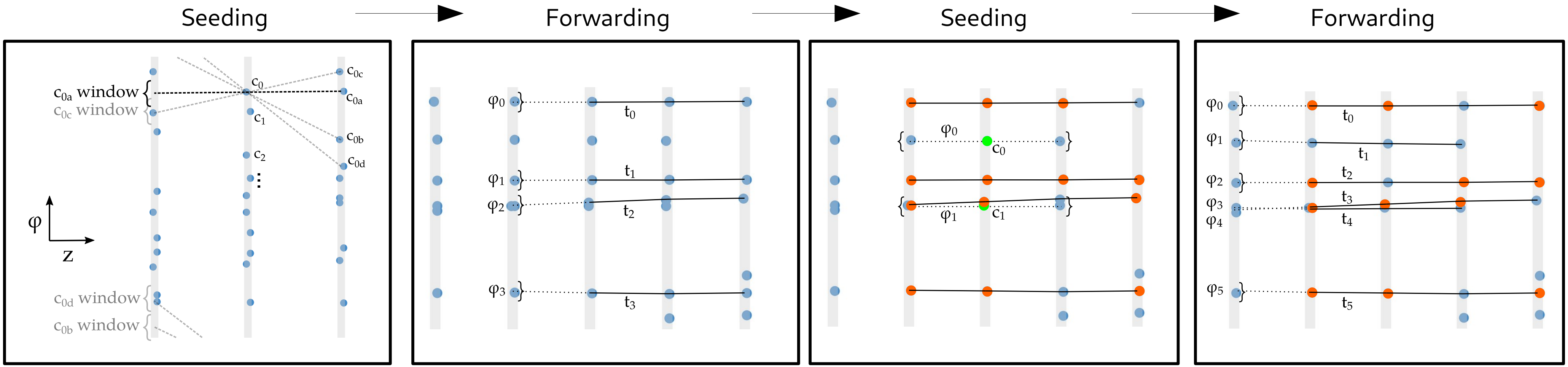
Projet IN2P3 proposé



GPU HLT1 within data acquisition system

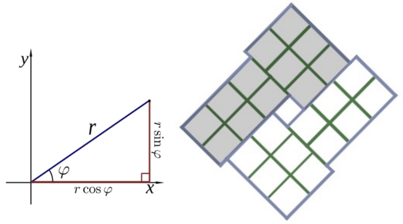


Example algorithm: “Triplet” finder

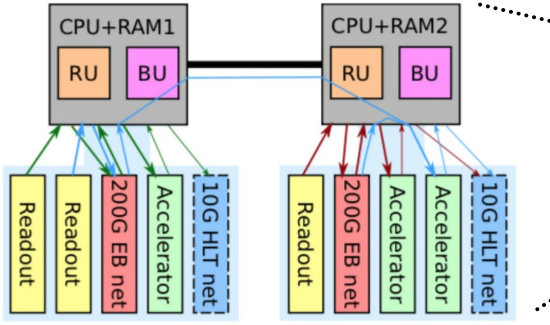


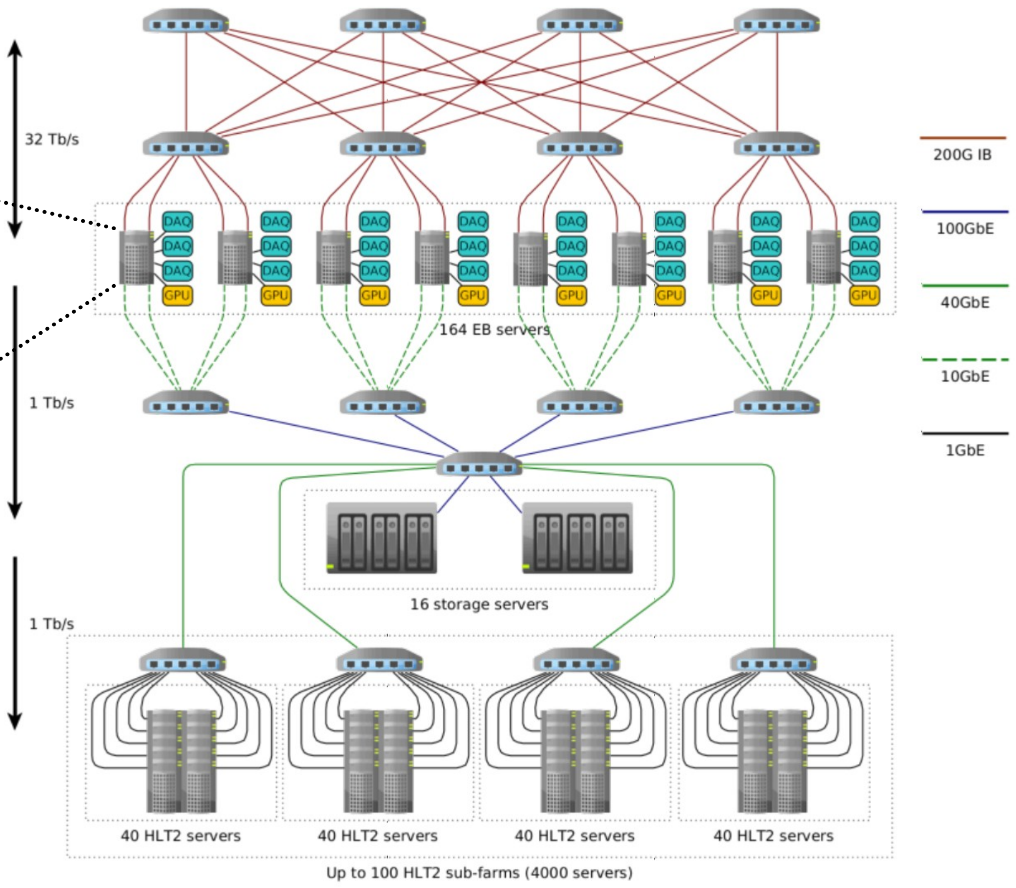
D. Campora et al, “Search by triplet: An efficient local track reconstruction algorithm on parallel architectures”, Journal of Computational Science 54, 101422 (2021)

- Build “triplets” of three measurements on consecutive layers → parallelization
- Choose them based on alignment in phi
- Hits sorted by phi → memory accesses as contiguous as possible: data locality
- Extend triplets to next layer → parallelization



GPU HLT1 within data acquisition system



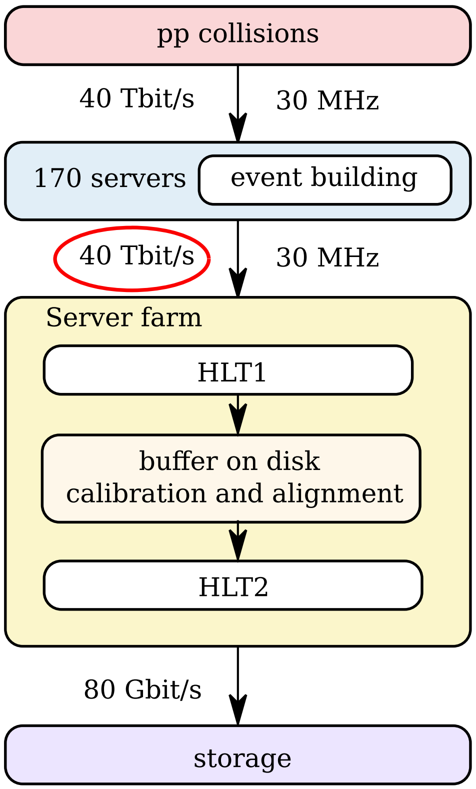


The converged architecture significantly reduces the cost of the full system

History: HLT1 architecture choice

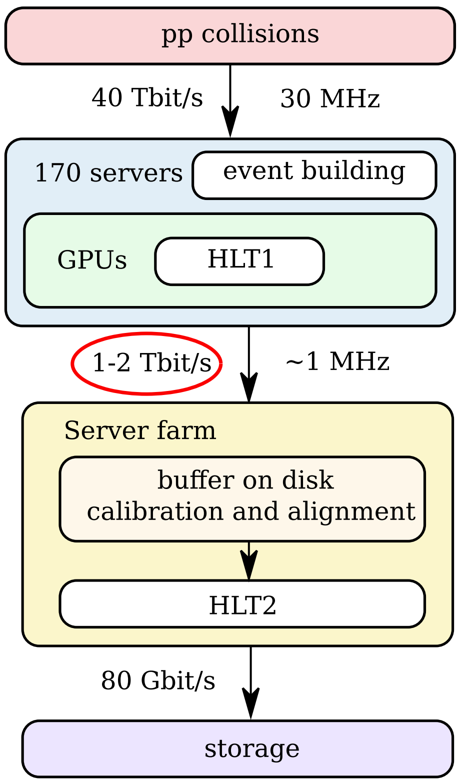
Proposal in TDR (2014)

CERN-LHCC-2014-016

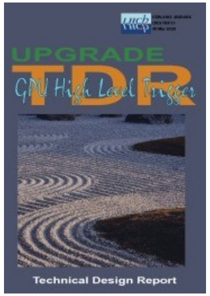


Updated strategy (as of 5/2020)

CERN-LHCC-2020-006



- Developed two solutions simultaneously
- Both the multi-threaded CPU & the GPU HLT1 fulfilled the requirements from the 2014 TDR
- Detailed cost benefit analysis (arXiv:2105.04031)
- GPU solution leads to cost savings on processors and the network
- Throughput headroom for additional features
- Decision: A GPU-based software trigger will allow LHCb to expand its physics reach in Run 3 and beyond.



See also arXiv:2106.07701 on LHCb's energy efficiency with a CPU and GPU HLT1

Overview of GPU usage in various HEP experiments

Experiment	Main tasks processed on GPU	Event / data rate	Number of GPUs	Deployment date
Mu3e	Track- & vertex reconstruction	20 MHz / 32 Gbit/s	O(10)	2023
CMS	Decoding, clustering, pattern recognition in pixel detector	100 kHz		2022 (tbc)
ALICE	Track reconstruction in three sub-detectors	50 kHz Pb-Pb or < 5 MHz p-p / 30 Tbit/s	O(2000)	2022
LHCb	Decoding, clustering, track reconstruction in three sub-detectors, vertex reconstruction, muon ID, selections	30 MHz/ 40 Tbit/s	O(250)	2022

https://arxiv.org/pdf/2003.11491.pdf

Allen software analysis

- Code analysis with SCC tool, using the cocomo model
- Source code written in CUDA, counted as "C" in table below

Allen scc analysis

Language	Files	Lines	Blanks	Comments	Code	Complexity	Complexity/Lines
C	269	27019	3621	2857	20541	870	646.61
Python	208	23099	1900	1582	19617	1226	1214.52
C++	142	19747	2478	2344	14925	2417	1758.51
C Header	133	12963	1741	2486	8736	977	403.35
Plain Text	101	20611	1480	0	19131	0	0.00
CMake	63	3847	545	685	2617	232	299.65
Shell	26	1677	299	160	1218	150	249.21
Markdown	25	1334	367	0	967	0	0.00
ReStructuredText	17	2454	666	0	1788	0	0.00
YAML	9	897	157	143	597	0	0.00
C++ Header	8	2837	396	312	2129	269	30.36
Autoconf	1	5	0	4	1	0	0.00
License	1	176	26	0	150	0	0.00
Docker ignore	1	5	0	0	5	0	0.00
Dockerfile	1	23	2	0	21	6	28.57
gitignore	1	36	0	1	35	0	0.00
Batch	1	45	8	1	36	6	16.67
XML	1	74	2	4	68	0	0.00
Total	1008	116849	13688	10579	92582	6153	4647.46

Estimated Cost to Develop $3,136,521
Estimated Schedule Effort 21.235592 months
Estimated People Required 13.121991