# Allen Core : status and plans

**Gonzalo Díaz López**

**Laboratoire de Physique Nucléaire et Hautes Energies, CNRS**

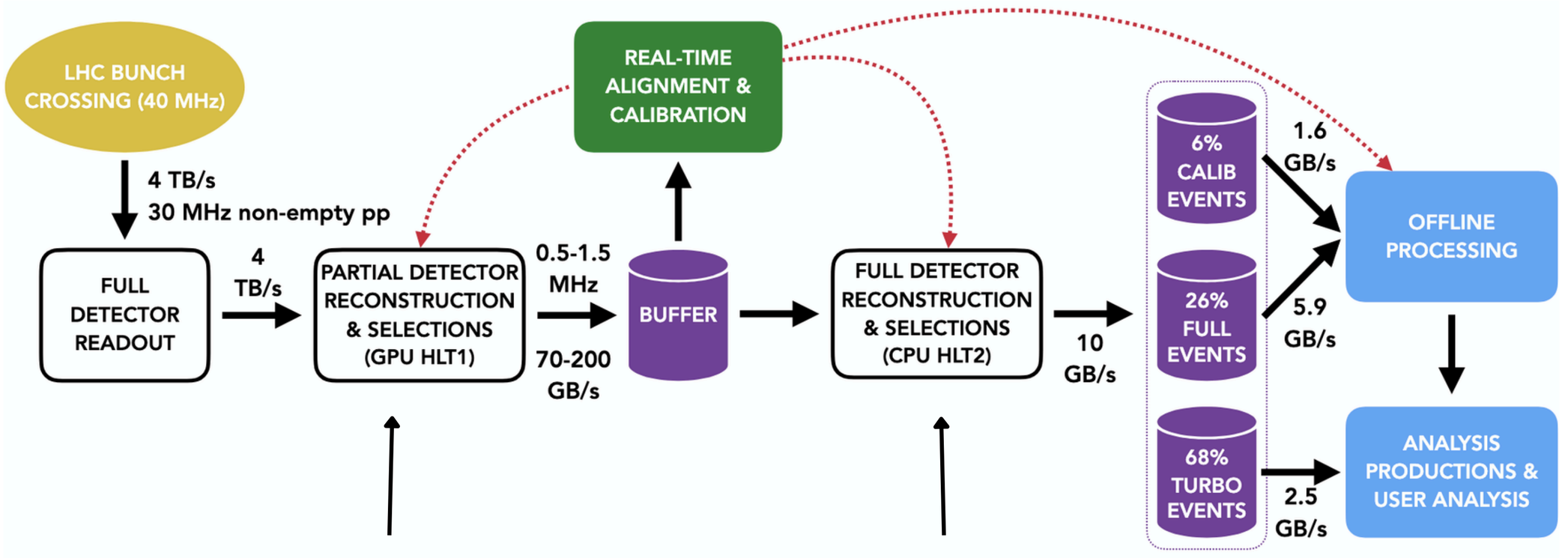



LPNHE
PARIS

**14th edition of the Streaming Readout workshop**
**September 29 - October 1 2026**



# LHCb data processing




**Allen (HLT1) — standalone application, runs on GPUs**

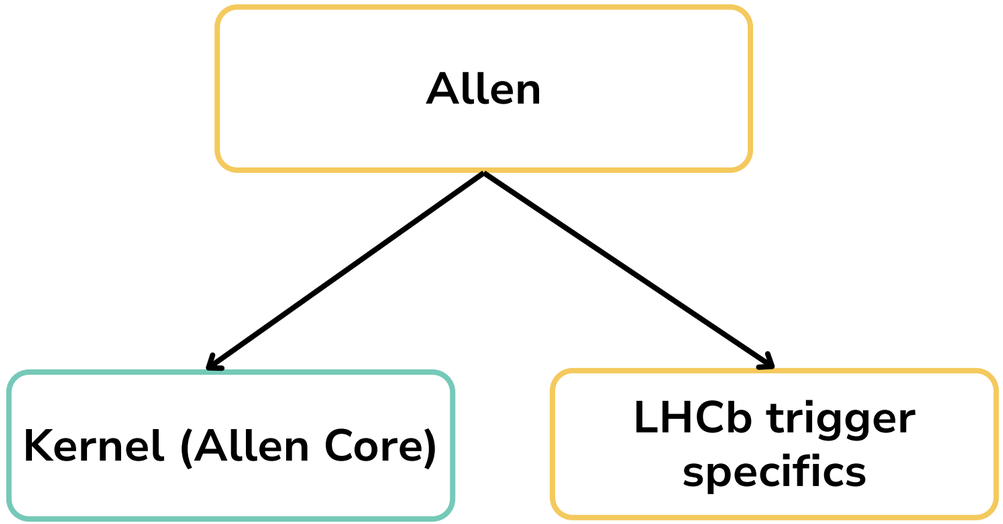
**Moore (HLT2) — Gaudi based application, runs on CPUs**

# Allen on GPUs

| Characteristics of LHCb's HLT1 | Characteristics of GPUs |
|---|---|
| **Intrinsically parallel problem**<br>• Run events in parallel<br>• Reconstruct objects in parallel | **SIMT paradigm** |
| Huge compute load | x10-100 TFLOPs compared to CPUs |
| • Throughput constrained<br>• No stringent latency requirements (full data stream readout) | Slow host-device memory transfert — handle data lifetime on the device |
| Small raw event data size (~100 kB) | Limited transfer bandwidth ~O(10GB/s) — can fit thousands of events in a single transfer |

# Allen Core framework project

- **Strip out** specific LHCb components from Allen
- **Decouple** framework from application
- **Interoperability** — provide interfaces to develop applications
- **Integrate** with Gaudi framework
- **Refactoring** and modularization
- **Simplify user experience** — documentation and maintability
- **Project funded by ODISSEE**
- **People involved**
  - PI: V. Gligorov (LPNHE) and D. vom Bruch (CPPM)
  - RE: A. Morris (CPPM) and G. Díaz López (LPNHE)




ODISSEE
Online Data Intensive Solutions for Science in the Exabytes Era

# Allen Core implementation status

**Latest release v0.2**

https://gitlab.cern.ch/allen-core

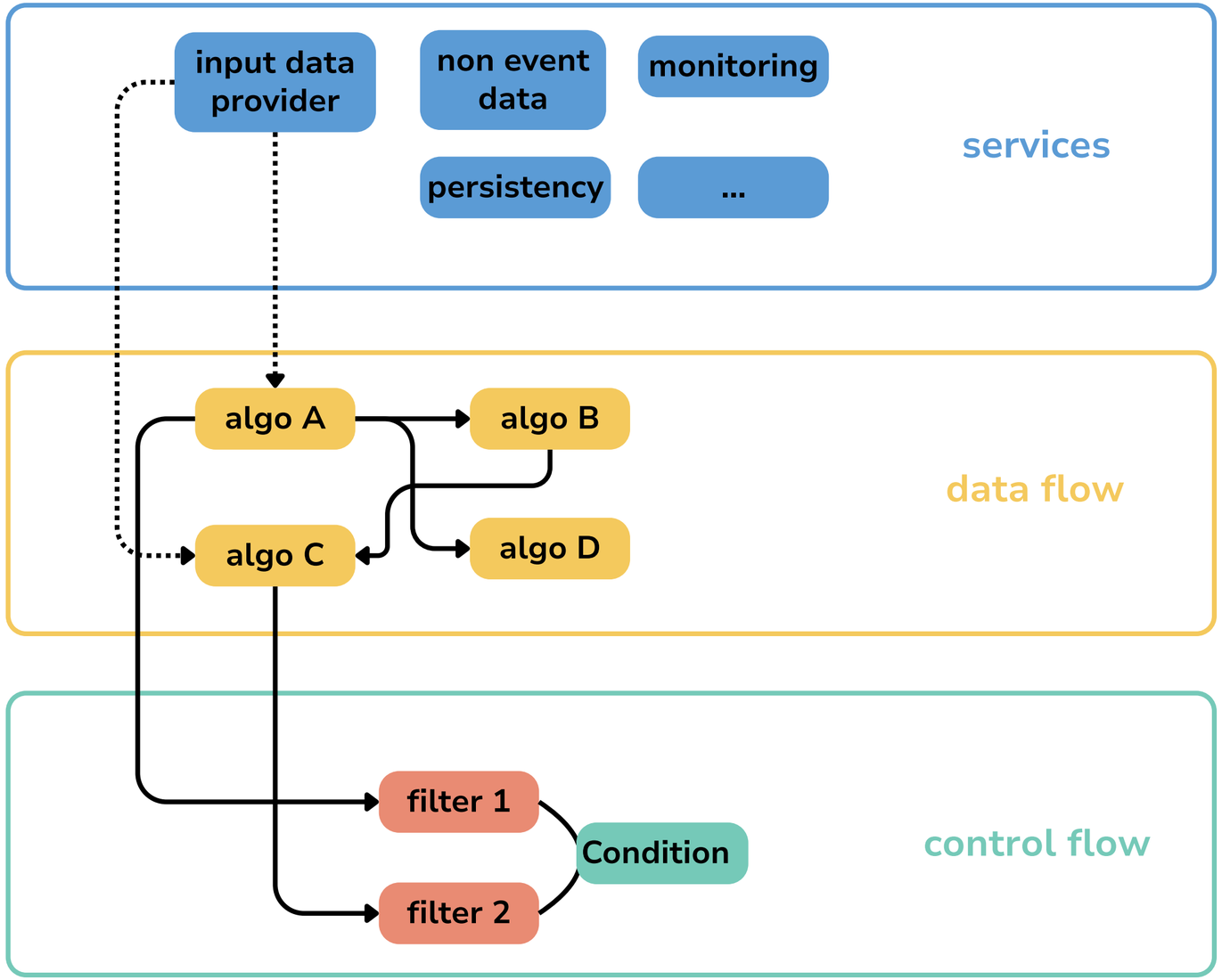


# Allen Core basics

## User side



- Implement algorithms (C++ and CUDA)
- Define sequence data and control flow (Python)
- Implement services (C++)

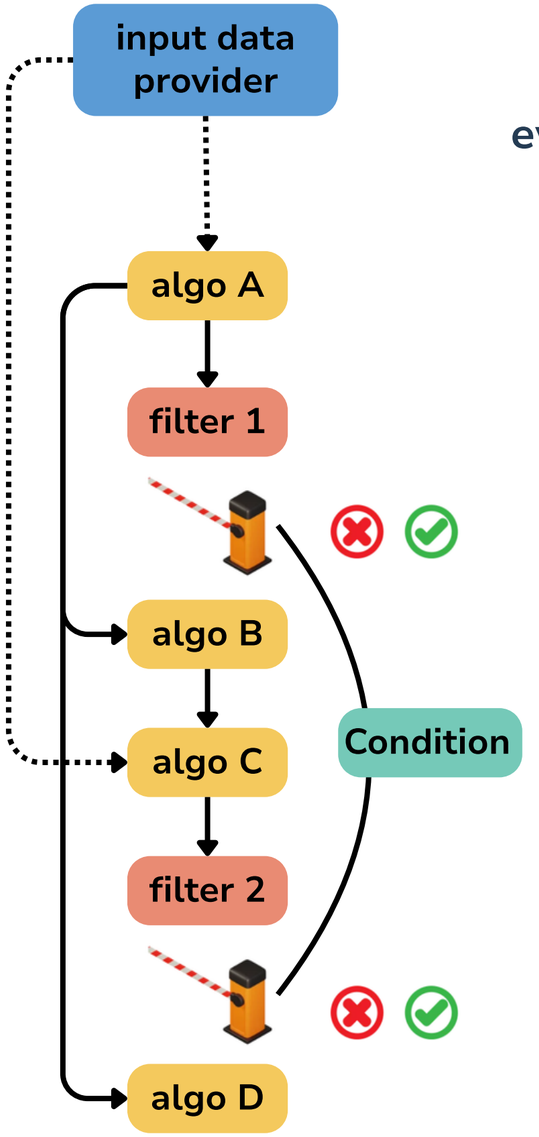


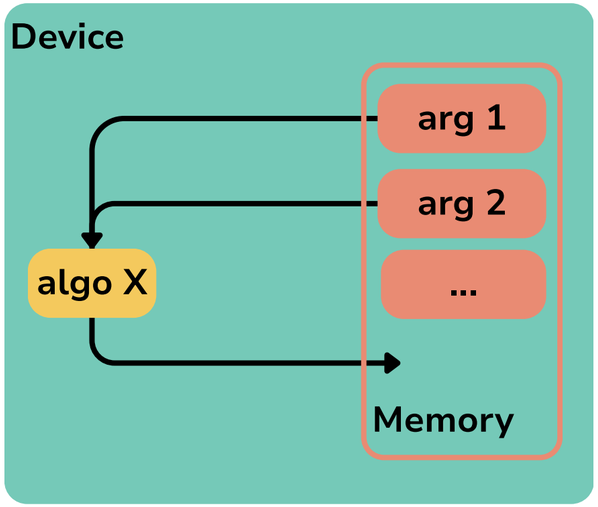
# Allen Core basics

## Framework side



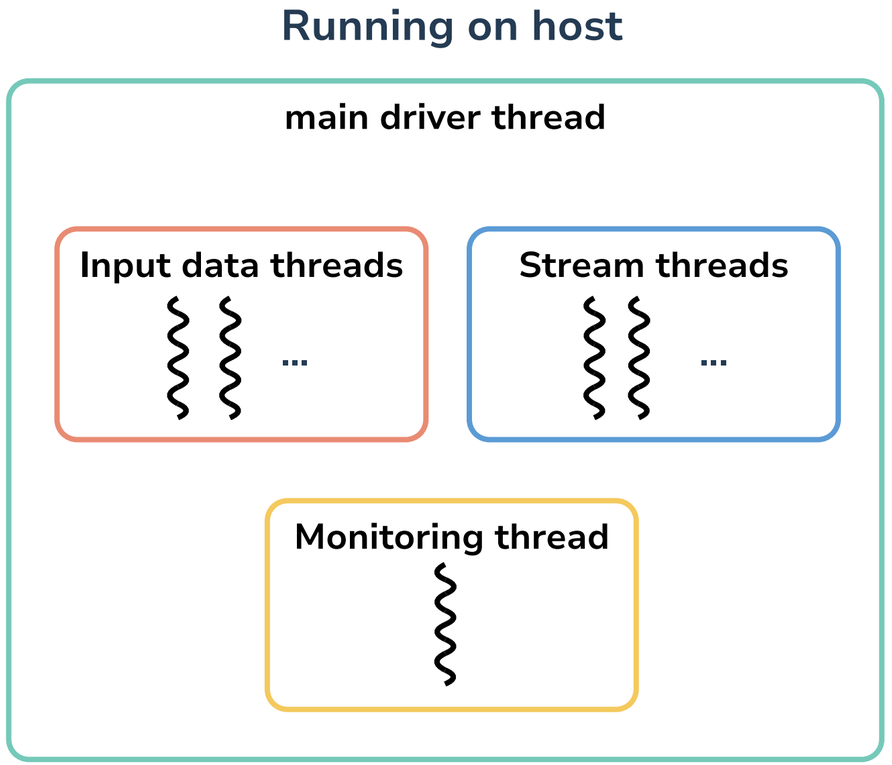
Sequencing
Portability
Event looping
Scheduling
Memory manag.



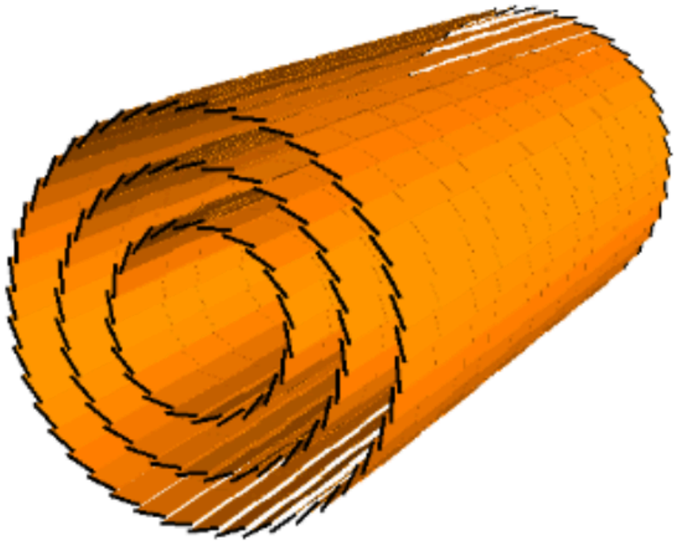
# Allen Core basics

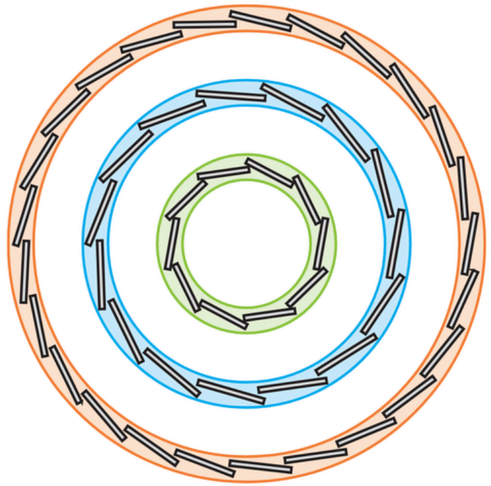
- **Sequencing** (build-time)
  - Pipeline creation fullfilling data and control flow dependencies
- **Portability** (build-time)
  - Backends allowing device-independent implementations
- **Event looping**
  - Input data fetching on parallel
  - Multi-streaming — multiple pipelines running on parallel, maintaining the device busy
  - Monitoring management — periodic aggregation
- **Scheduling**
  - Event multi-batching
  - Dispatch algorithms (allows both device and host)
- **Memory management**
  - Memory preallocation on device
  - Dynamic lifetime management

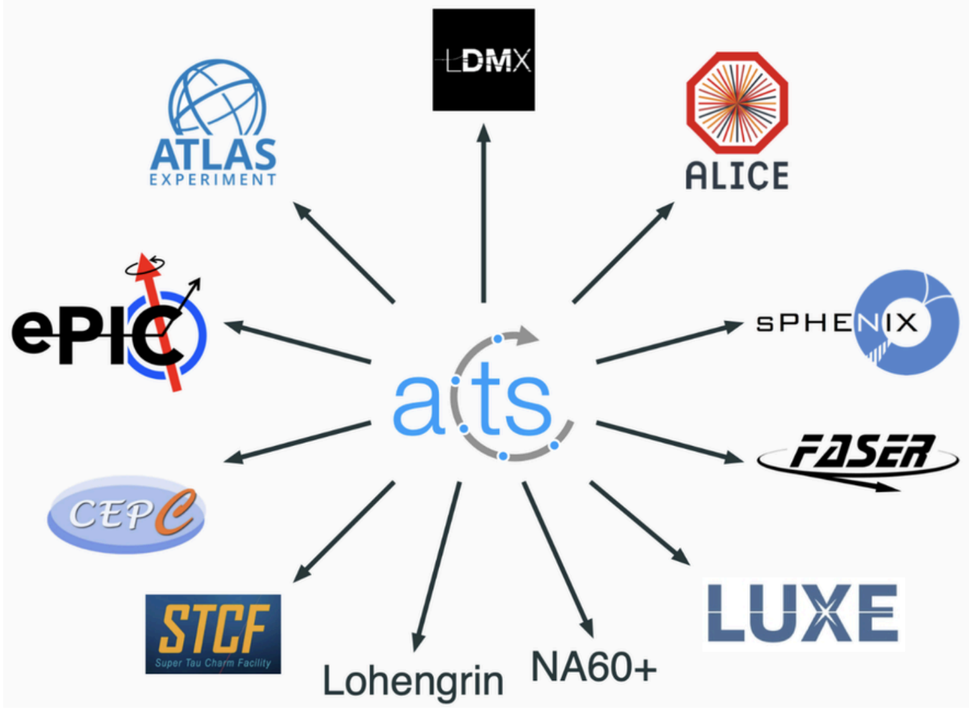


# Allen Core Demonstrator

- **LHCb independent example application**
- **Use “common” HEP tools**
  - **ODD** — virtual **DD4HEP** detector used for benchmarking (e.g https://colliderml.com)
  - **ACTS/Traccc** based reconstruction

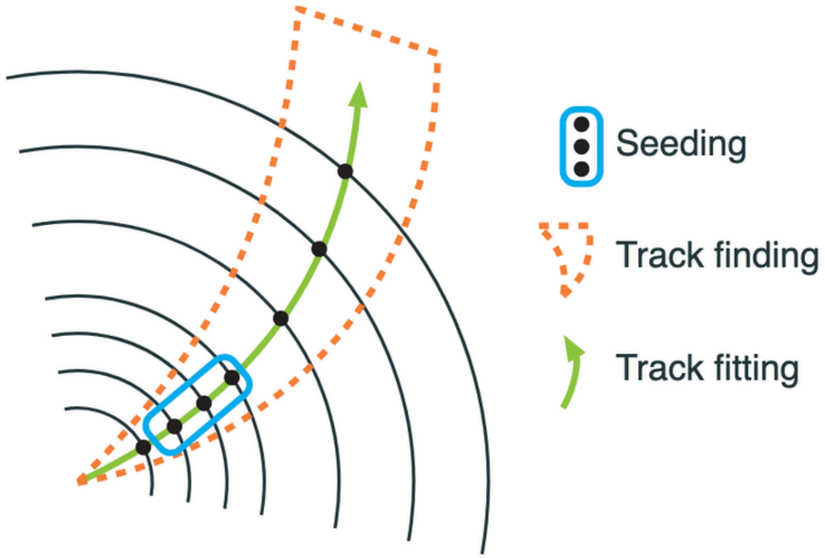


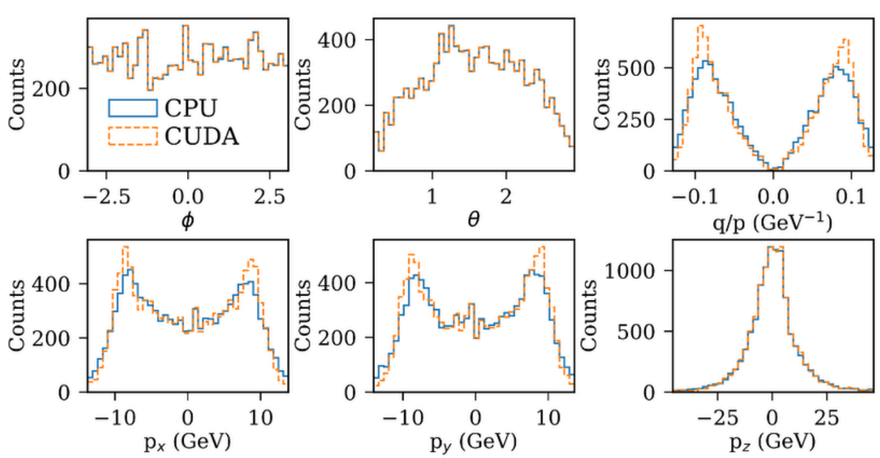




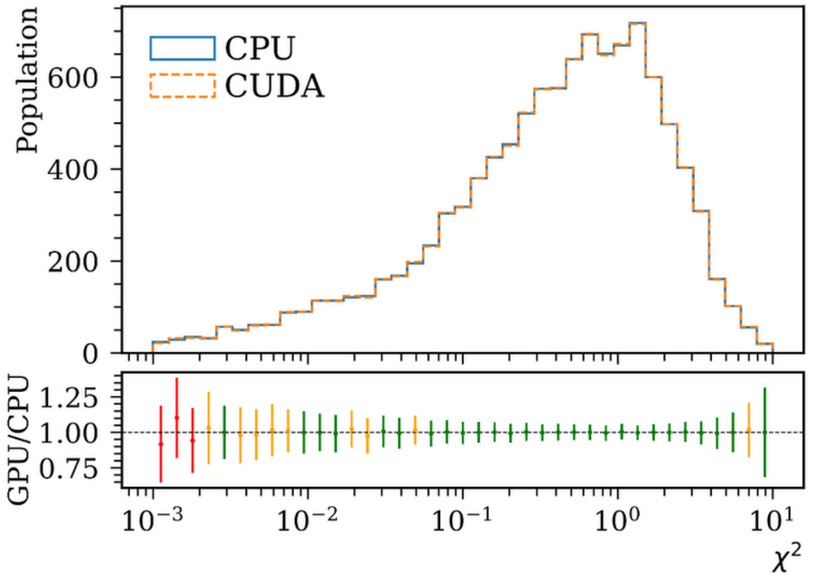
# Allen Core Demonstrator

- **Reconstruction steps**
  - Clusterization
  - Seeding
  - Track fitting and finding
- **ACTS/Traccc - Allen Core limitations**
  - Use of custom made data structures (vecmem)
  - No multi-batching
  - Per event data transfers
- **Full reco chain in a single algorithm**



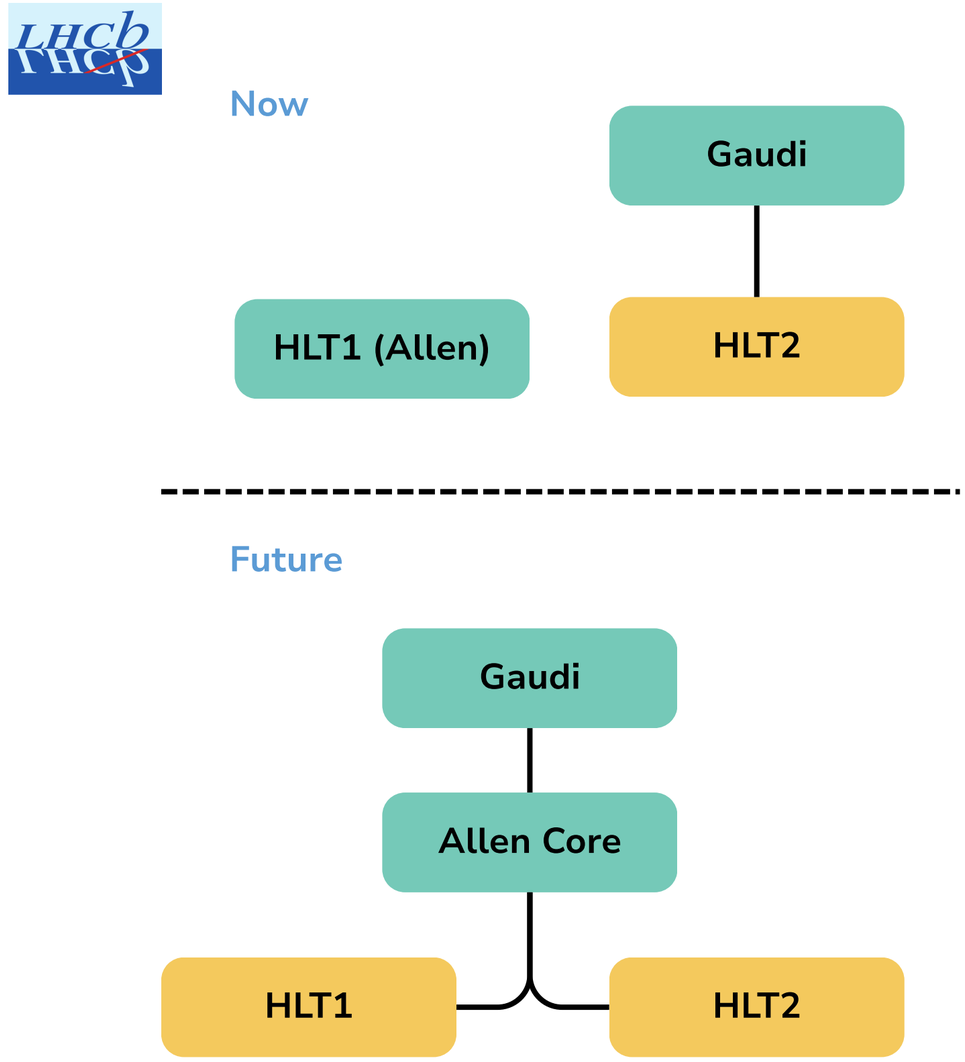




Disagreements between GPU and CPU for momentum are a known issue of traccc's CPU algorithms



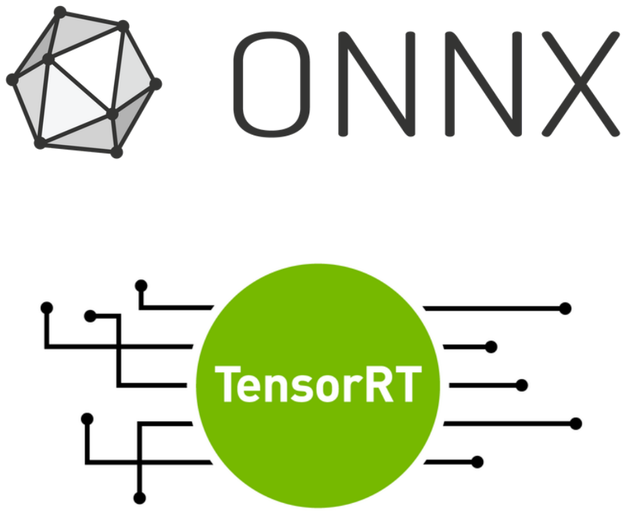
# Allen Core plans — Gaudi integration

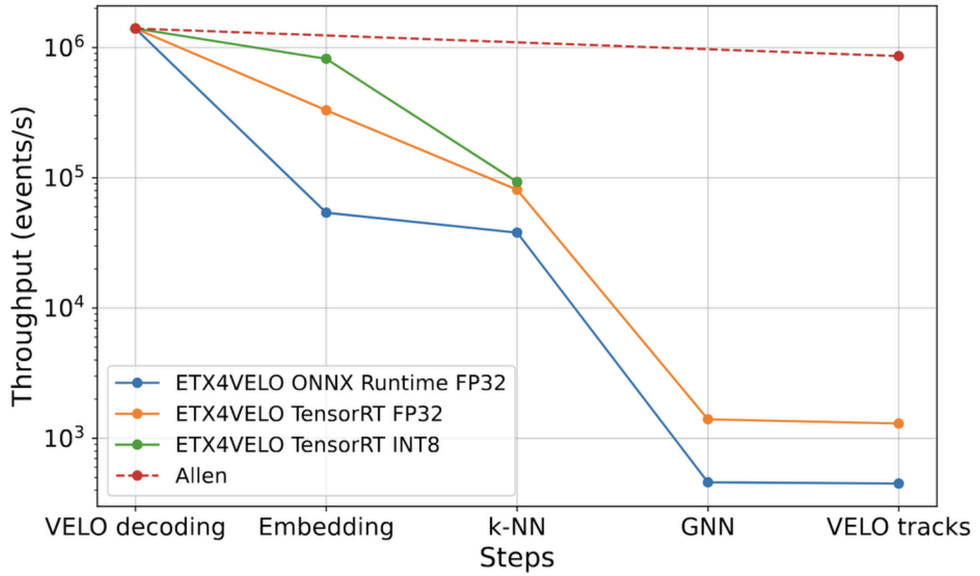
- **MultiEventScheduler** is the key missing piece for integration
- **Just merged on Allen** by LHCb
- **Features**
  - Event loop drived by Gaudi — solves inherent problems such as monitoring periodic flushing
  - Running Gaudi algorithms
  - Run-time sequencing
  - Mixing Allen and Gaudi algorithms (GaudiAllenAlgorithmWrapper)
  - Mixing of single and multi event algorithms
- Porting it to Allen Core will imply dealing with extra **LHCb dependencies**
- Can solve **LHCb's** desire of an **unified framework** for HLT1 and HLT2



# Allen Core plans —  details to be discussed

- Multi device paralelization — now one instance per device needed
- Extend portability backends (AMD, FPGA) — only CUDA is supported yet
- AI algorithms (interoperability with inference engines) — Allen works already performed
- Add abstractions and interfaces — user friendliness
- Production grade repository — well tested and documented

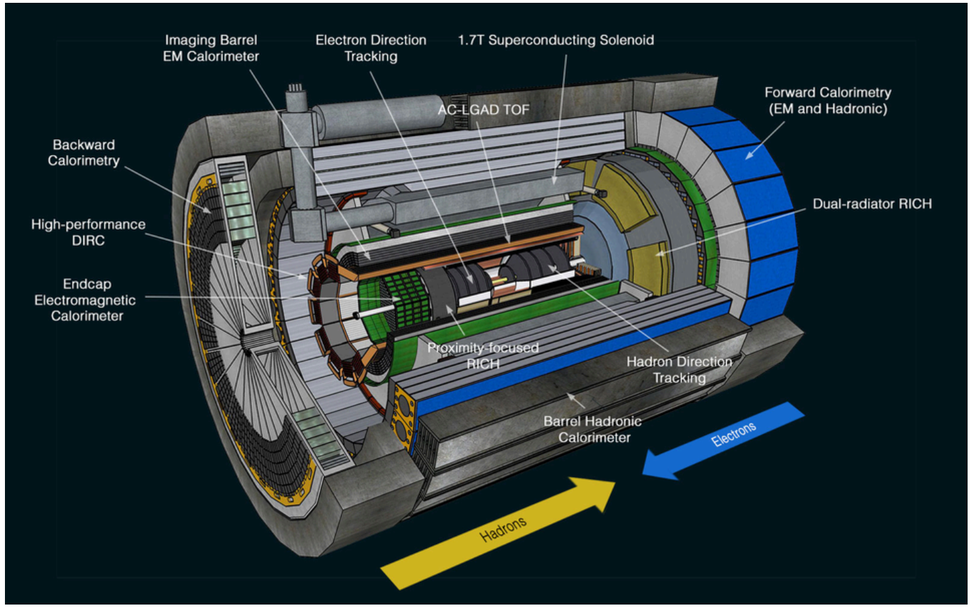


# Allen Core plans — future users

- **LHCb's** roadmap towards Run 4 would be to integrate Allen with Allen Core in the next months
  - ideally followed by HLT2
- **ePIC** interested as an online and calibration tool (echelons 1-2)
  - demonstrator application — triggerless reconstruction and calibration for the backward electromagnetic calorimeter (under design and construction at IJCLab, Paris)
  - deploy pipeline on a French cluster
  - implementing ePIC's data model, geometry and algorithms needed

ALLEN 4EIC
A HETEROGENEOUS, OPEN-ACCESS FRAMEWORK FOR REAL-TIME HIGH-THROUGHPUT DATA PROCESSING



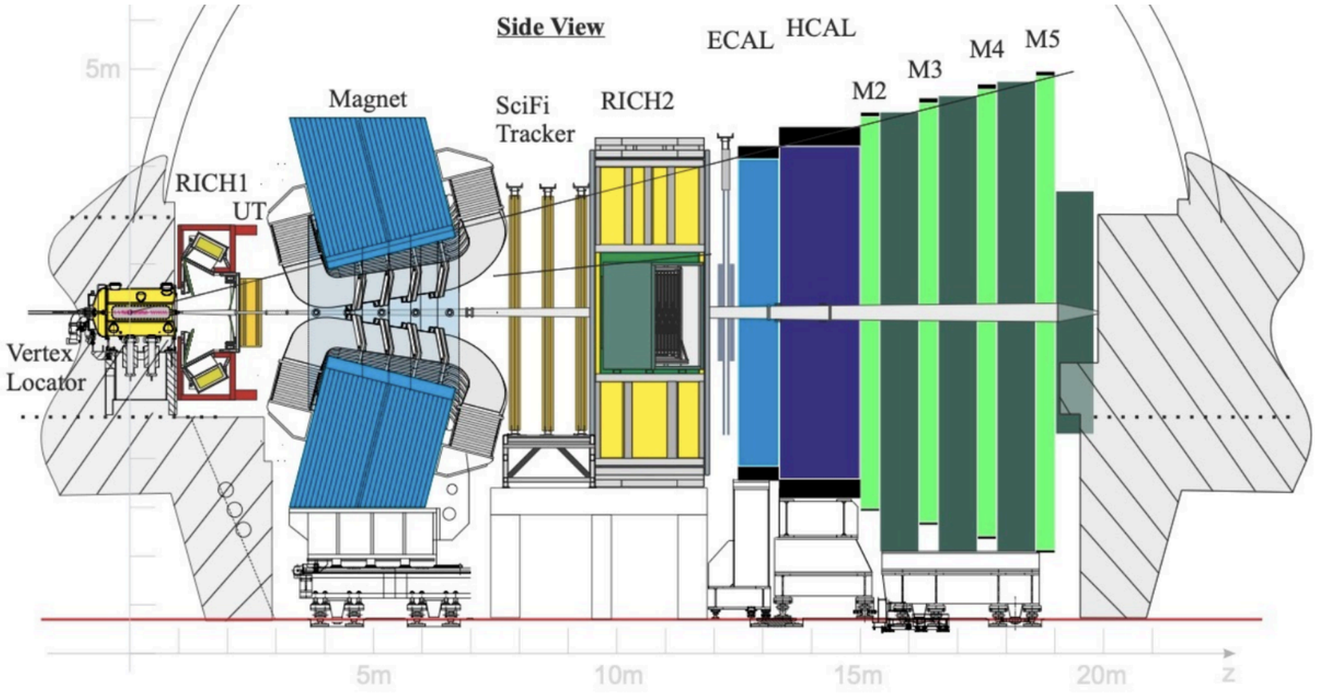
# Take home message

- **Allen Core working version** (prebuild environments with dependencies provided)
  - work still to be done in testing and documentation
- Improve Gaudi integration in the near future
- Integration of Allen Core with **LHCb**'s Allen
- **Allen Core Demonstrator working version** with a simple sequence for ODD
  - usability for LHCb independent applications
  - ACTS/Traccc algorithm implementation does not match well with Allen Core — no multibatching
  - need to implement native Allen versions for benchmarking
- An Allen Core based reconstruction planned by ePIC in the near future

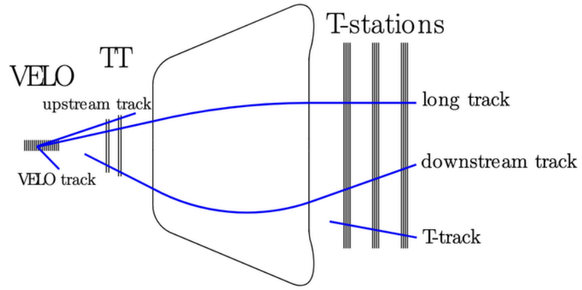
*Thanks for your attention*

# Backup

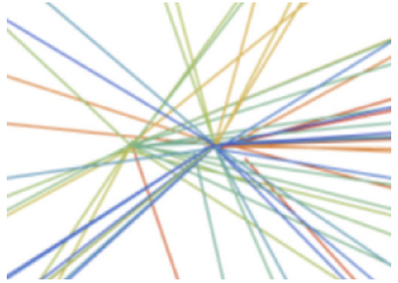
# LHCb experiment at CERN



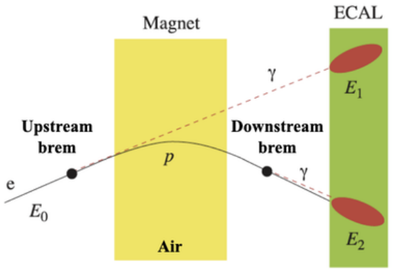

## Tracks



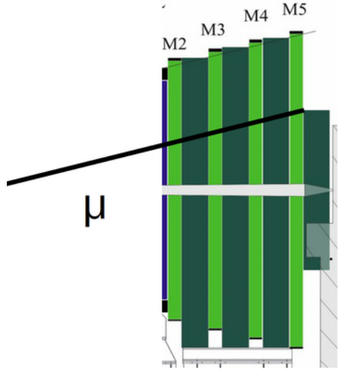

## Vertices



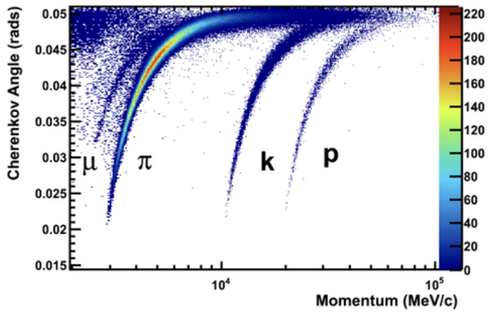
## Calorimeter hits



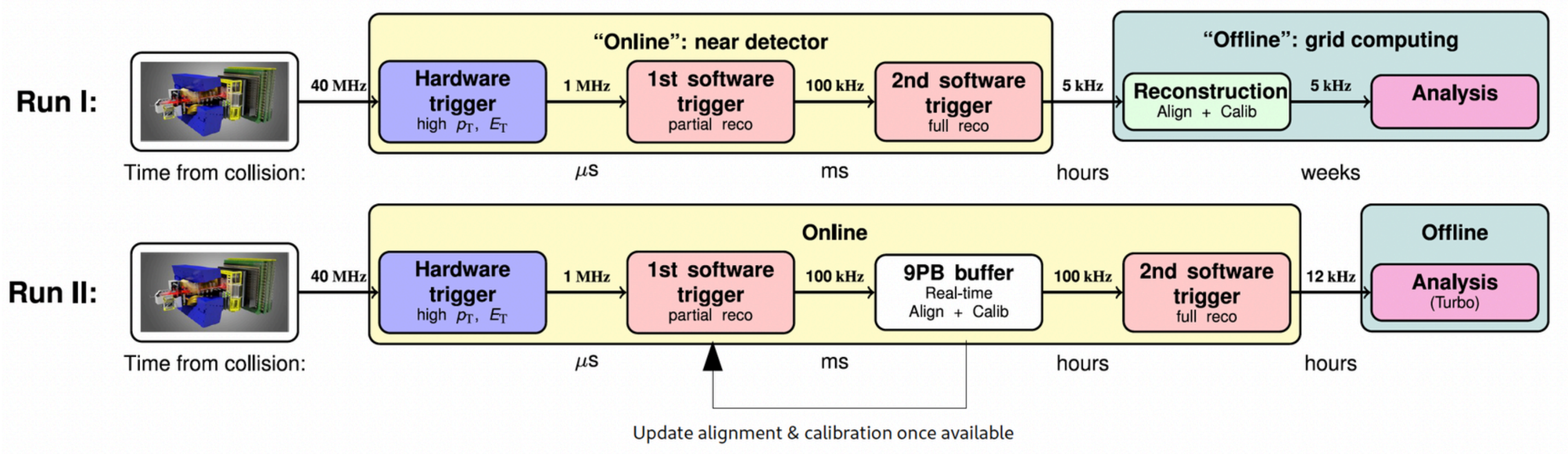

## Muon hits




## Cherenkov rings

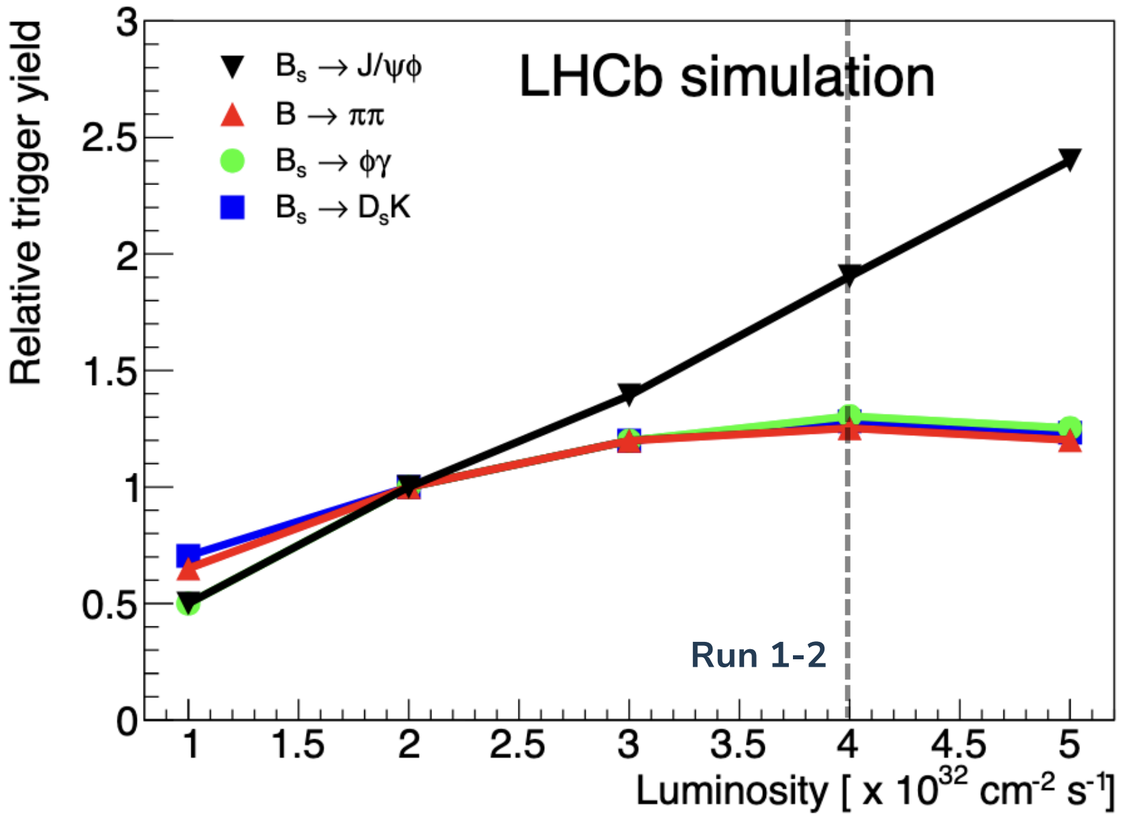


# LHCb trigger before Run 3



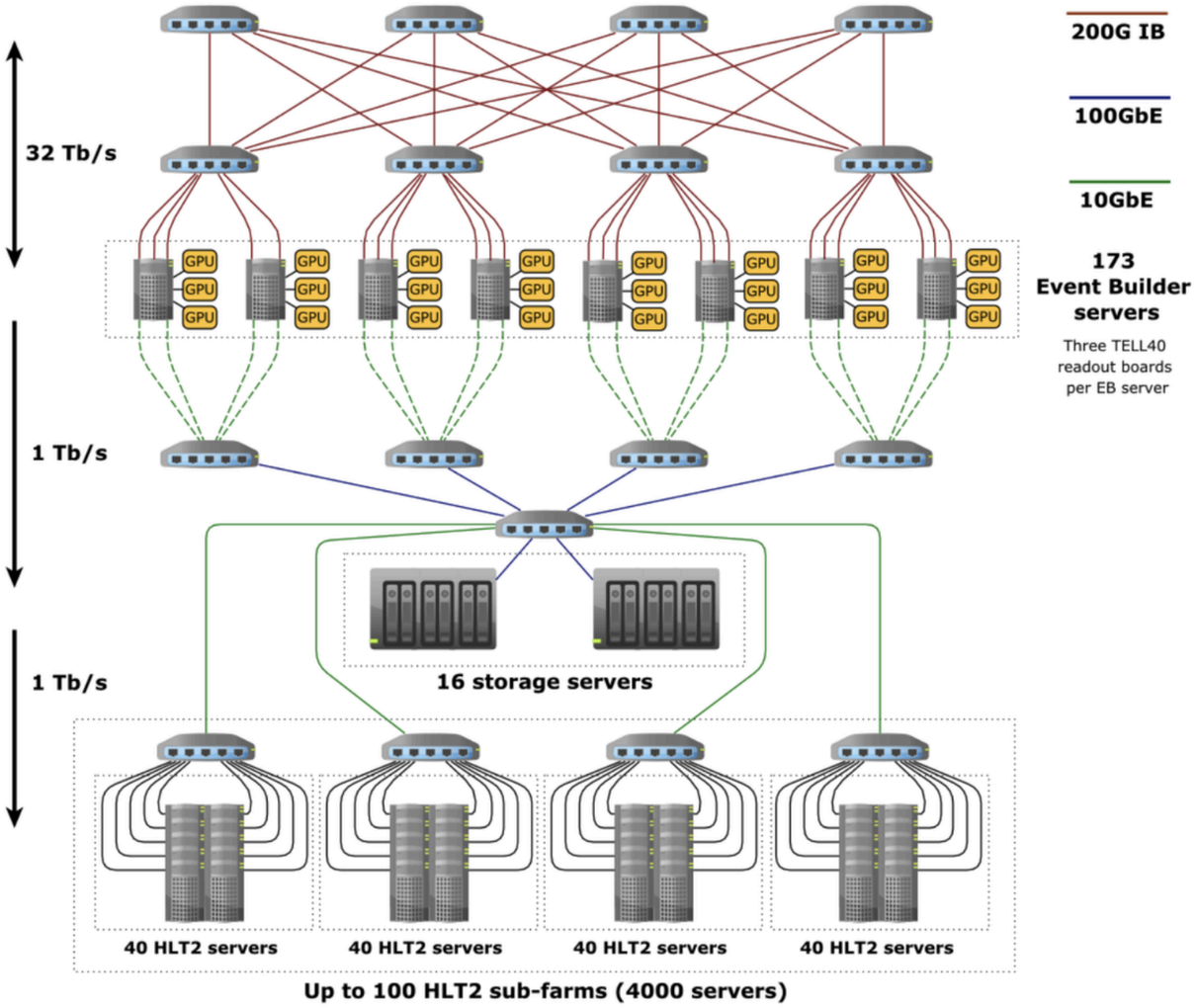
# LHCb trigger chalenge for Run 3

- x5 luminosity increase (~5-6 pileup)
- Hardware trigger inefficient for pileups > 1
- Efficiency plateau for final hadronic states

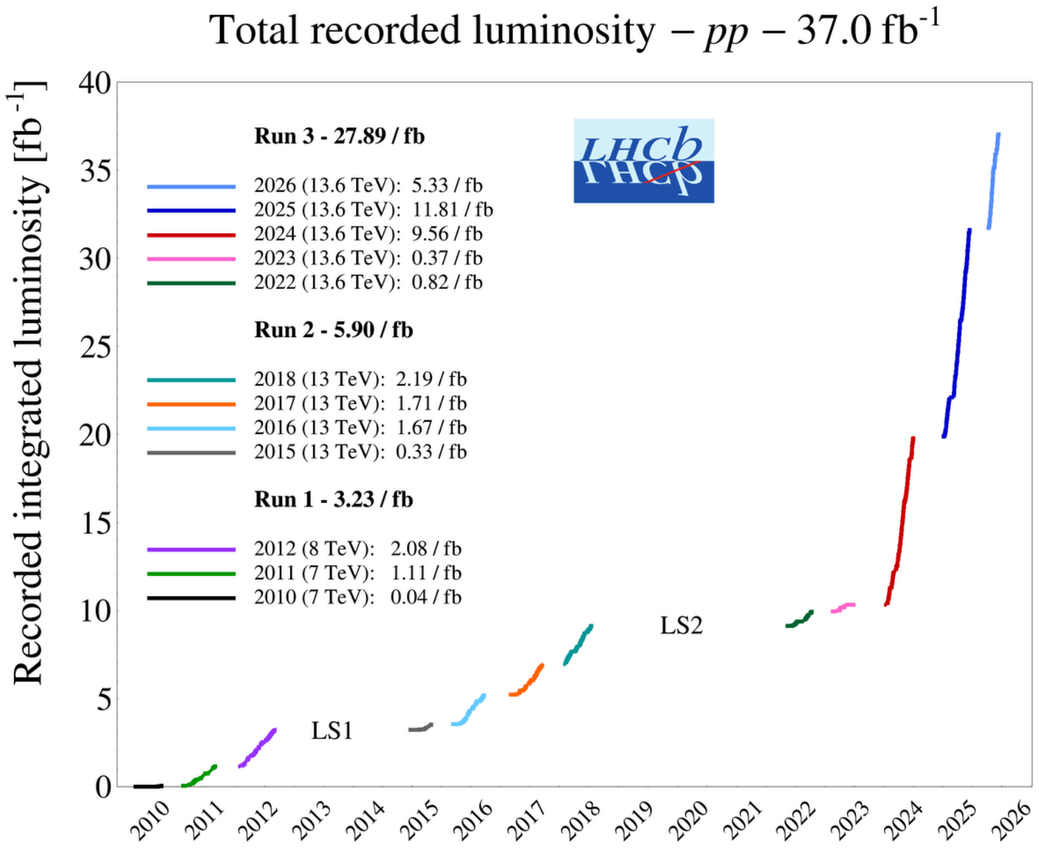


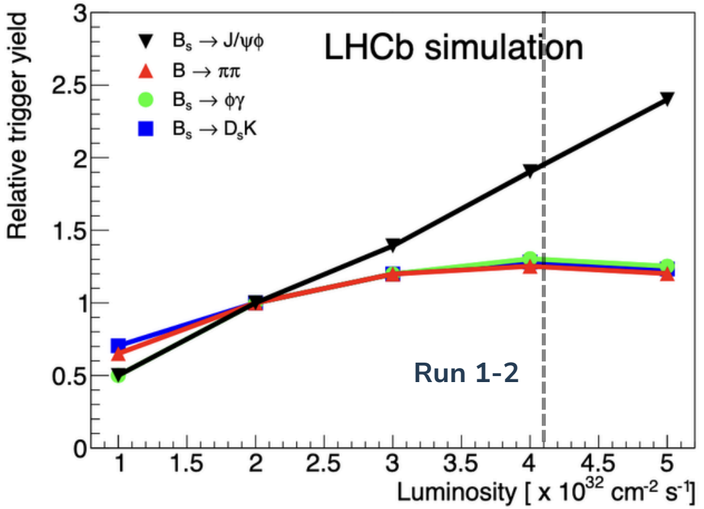
# LHCb data acquisition architecture

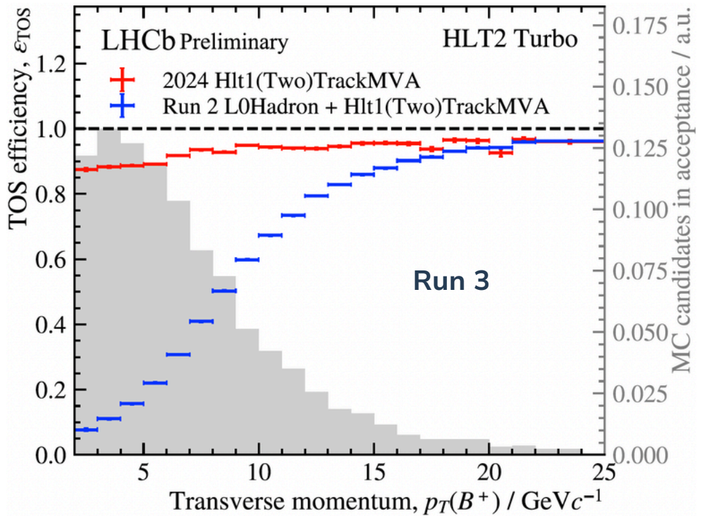
- Raw detector data received by FPGA cards
- 173 event build servers:
  - data aggregation from detectors
  - 3 GPUs/server run HLT1  (Nvidia A5000)
- HLT1 output stored on 40PB buffer while alignment and calibration is processing
- HLT2 send to 250K CPU core farm



# Allen success story — Run 3




**hardware trigger**

**software trigger**

# Allen Core basics

- Algorithms in C++ and CUDA
- Contain host and device parts
- Data and control flow in Python

```
3   #include "kernel/algorithms/AlgorithmTypes.cuh"
4
5   namespace dummy_reconstruction {
6     struct Parameters {
7       HOST_INPUT(host_event_list_t, unsigned) host_event_list;
8       HOST_INPUT(host_raw_bank_version_t, int) host_raw_bank_version;
9       MASK_INPUT(dev_event_list_t) dev_event_list;
10      DEVICE_INPUT(dev_raw_input_t, char) dev_raw_input;
11      DEVICE_INPUT(dev_raw_input_offsets_t, unsigned) dev_raw_input_offsets;
12      DEVICE_INPUT(dev_raw_input_sizes_t, unsigned) dev_raw_input_sizes;
13      DEVICE_INPUT(dev_raw_input_types_t, unsigned) dev_raw_input_types;
14      DEVICE_OUTPUT(dev_reconstruction_t, float) dev_reconstruction;
15    };
16
17    struct dummy_reconstruction_t : public DeviceAlgorithm, Parameters {
18      void set_arguments_size(ArgumentReferences<Parameters>, const RuntimeOptions&
19          , const Constants&) const;
20
21      void operator()(
22        const ArgumentReferences<Parameters>&,
23        const RuntimeOptions&,
24        const Constants&,
25        const AllenCore::Context& context) const;
26
27      private:
28          AllenCore::Property<bool> m_property {this, "dummy_property", true, "dummy property"};
29          AllenCore::Property<dim3> m_block_dim {this, "block_dim", {32, 1, 1}, "block dimensions"};
30    };
31
32    __global__ void dummy_reconstruction(Parameters, const bool);
33  } // namespace dummy_reconstruction
```

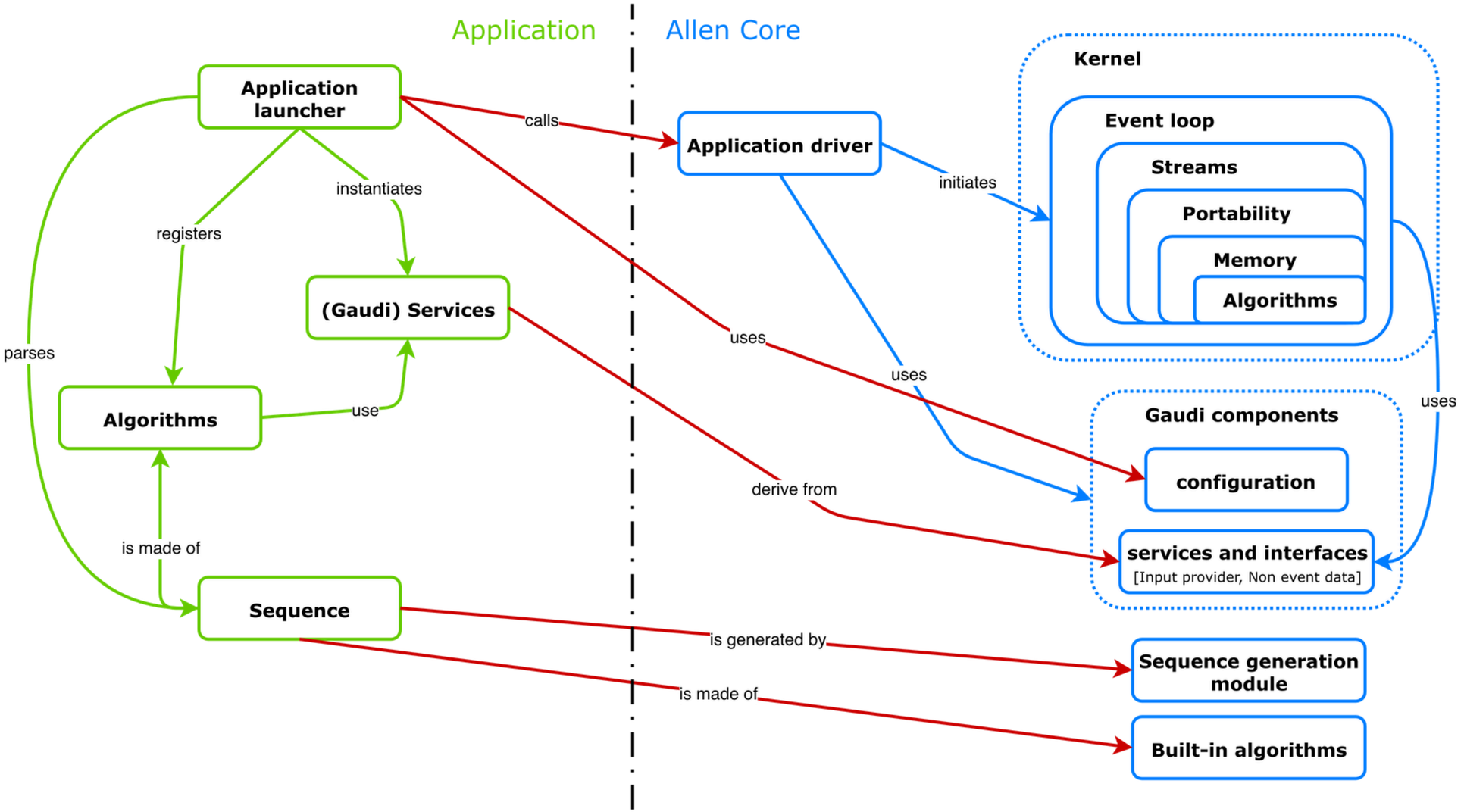
Data dependencies

Running on host

Running on device

*dummy algorithm example to show interface

# Allen Core components



# Allen Core plans — Gaudi integration

