# Introduction to Allen

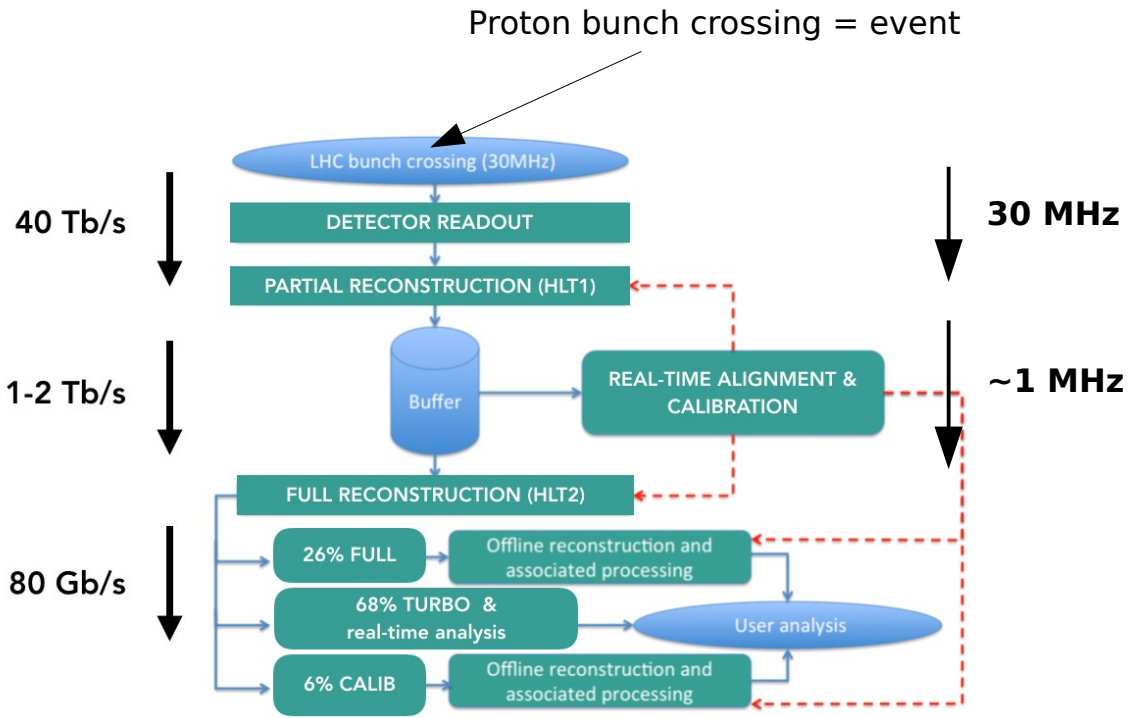
**Dorothea vom Bruch**

July 2019

1st Real Time Analysis Workshop

Institut Pascal, Université Paris-Saclay, France

# LHCb: Real time event selection




**High Level Trigger 1 (HLT1)**

- Perform full charged particle track reconstruction
- Small number of inclusive single or two-track selections

  → Efficiently select events that contain particles of interest for LHCb
- Reduce event rate by roughly factor 30

  → very compute intensive!
- Baseline solution:

  use filter farm of 1000 PCs for HLT1 & HLT2

# The Allen project

- Fully standalone software project: https://gitlab.cern.ch/lhcb-parallelization/Allen
- Compact, scalable and modular framework built for running HLT1 on GPUs
- Only requirements: a C++17 compliant compiler & CUDA v10.0

- Configurable static sequences of algorithms
- Pipelined stream sequence → hide memory copies to and from the GPU
- Custom memory manager → no dynamic allocations
- Built-in physics validation
- Optional compilation with ROOT[1] for plot generation

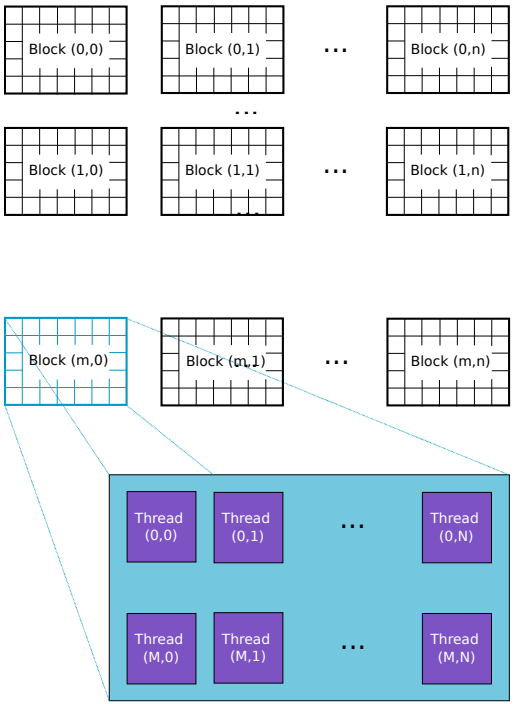
- Continuous integration: throughput and algorithm breakdown checked for every merge request

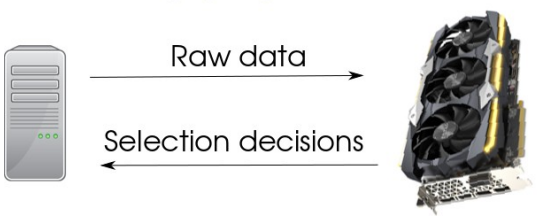
- Project started in February 2018
- Roughly 14 part-time developers, 2 almost full-time
- R&D intended for Run 3 (2021)

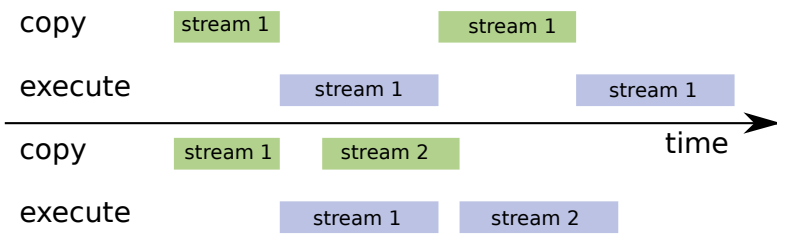
[1] ROOT data analysis framework: https://root.cern.ch/
More details on core software: Talk by D. Campora at ACAT

# HLT1 on GPUs

- Run thousands of events in parallel, using:
  - Blocks: Events under execution
  - Threads: Intra-event parallelism
- All data is stored in Structure of Arrays (SoA) data layout
- Memory accesses are contiguous
- All algorithms have been (re-)designed for the GPU architecture
- Data locality is preserved within algorithms
- Using single precision only
- Low memory I/O requirements, use PCIe connection
- Memory transfers are hidden by calculations: one CUDA stream launched with one CPU thread



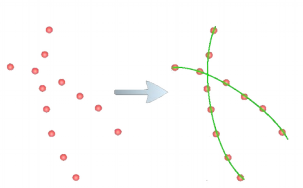



# Recurrent tasks of HLT1

**Raw data decoding**

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



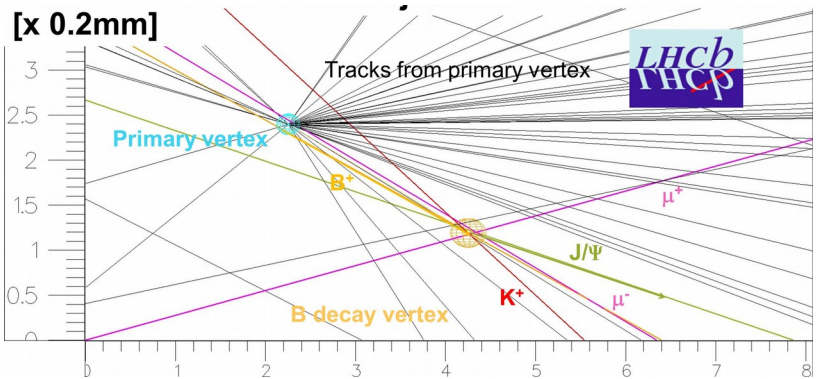
f(x) = … +/- …

**Track reconstruction**

- Consists of two steps:
  - Pattern recognition: Which hits belong to which track? → Huge combinatorics
  - Track fitting: Done for every track
- Parallelizable over events, combinations of hits, and tracks

**Vertex finding**

- Where did proton-proton collisions take place?
- Where did particles decay within the detector volume?
- Parallelizable over events, combinations of tracks



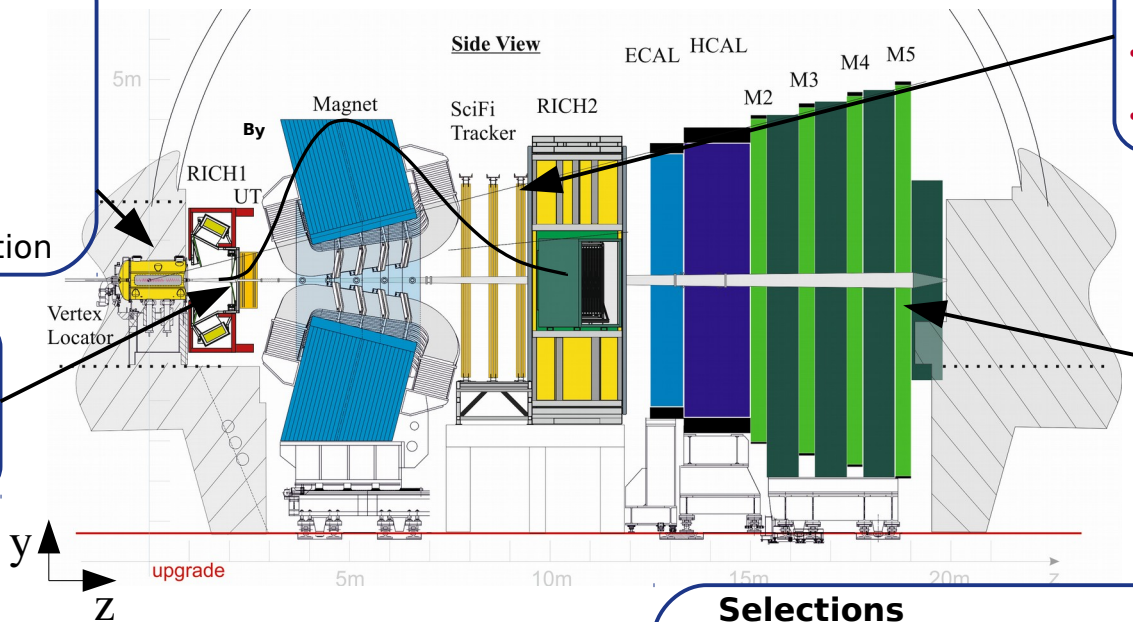
# LHCb HLT1 elements

**Velo**

- Decode raw banks
- Clustering of pixel hits
- Track reconstruction
- Primary vertex reconstruction

**UT**

- Decode raw banks
- Track reconstruction




**Kalman filter**

- Track fit

**SciFi**

- Decode raw banks
- Track reconstruction

**Muons**

- Decode raw banks
- Match hits to tracks

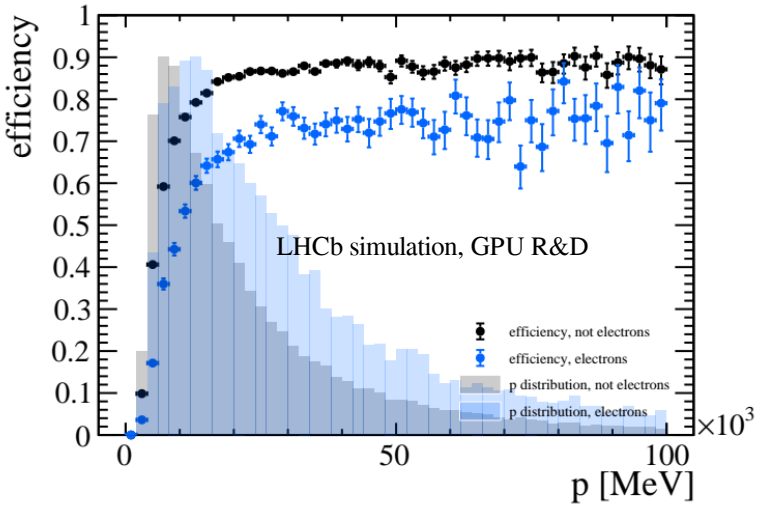
**Selections**

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

Framework TDR for the LHCb Upgrade: CERN-LHCC-2012-007

# Physics performance checked within Allen

Track reconstruction efficiency for tracks passing through the Velo, UT and SciFi detectors, Bs→PhiPhi events




Tuned for 1 MHz output rate (factor 30 reduction in event rate)

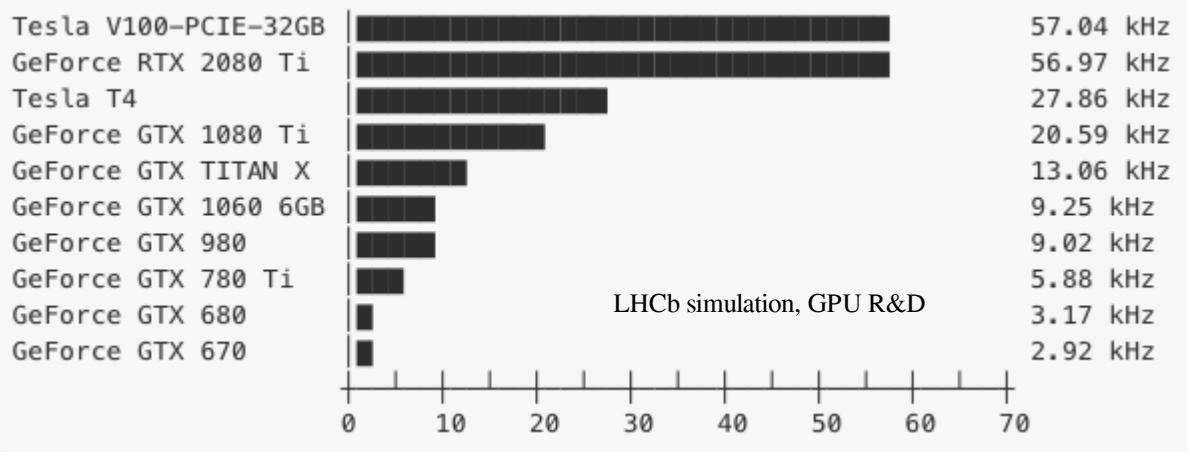
| Signal | GEC | TIS -OR- TOS | TOS | GEC × TOS |
|---|---|---|---|---|
| $B^0 \to K^{*0}\mu^+\mu^-$ | 88 | 81 | 77 | 67 |
| $B^0 \to K^{*0}e^+e^-$ | 84 | 69 | 60 | 50 |
| $B_s^0 \to \phi\phi$ | 85 | 82 | 78 | 66 |
| $D_s^+ \to K^+K^-\pi^+$ | 84 | 48 | 35 | 29 |
| $Z \to \mu^+\mu^-$ | 75 | 90 | 89 | 67 |

Values given in %

More details on algorithms: Talk by D. vom Bruch at PASC

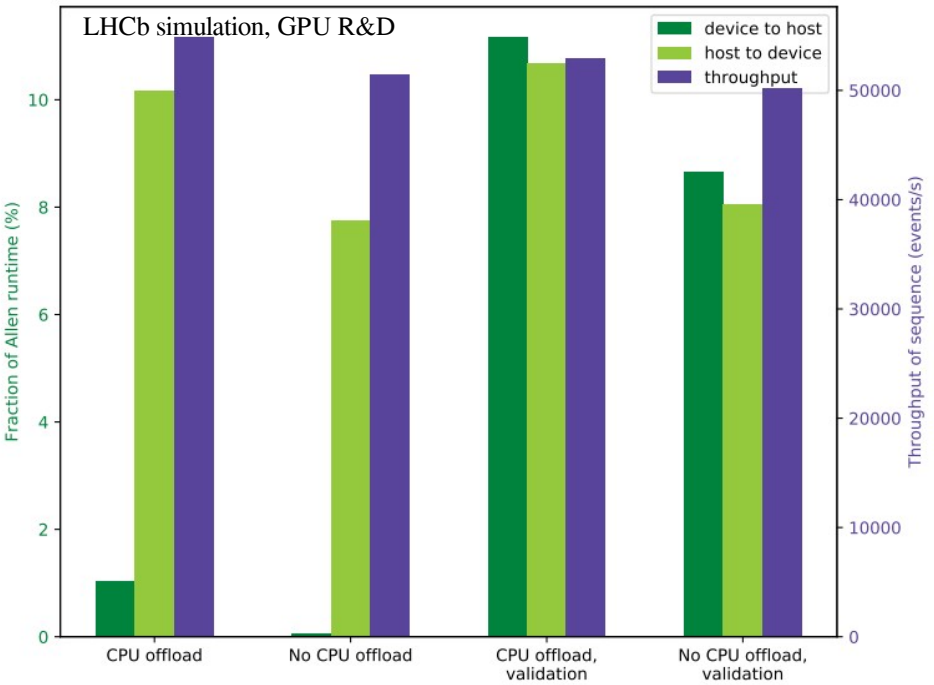
# Throughput on various GPUs

Throughput of the full HLT1 sequence, taken from our continuous integration output




**The system can run on 500 consumer / scientific GPU cards**

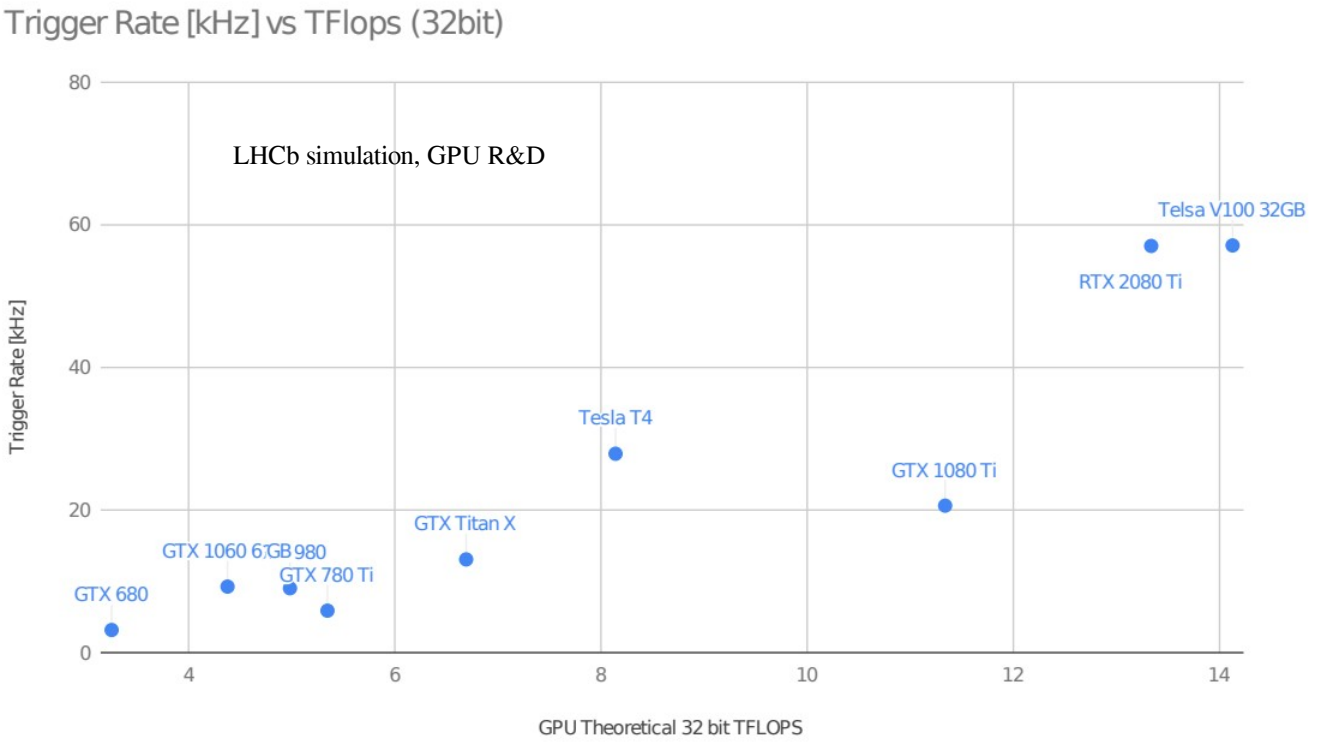
# Hide host - device data transmission




- Host - device data transmission via PCIe takes at most 12% of the computation time

  → Data copies can be hidden using different pipelines (streams)

# Allen on various GPUs




- Software scales to newer generations of cards
- Can expect increasing performance with the next generations

# Scheduler

- HLT1 consists of currently 70 algorithms
- Allen is scalable for when new features are added
- Average developer needs little framework-specific knowledge
- Sequence of algorithms is configured at compile time in Allen
- Simply adding / removing a line in a configuration file

```
SEQUENCE_T(
  velo_estimate_input_size_t,
  prefix_sum_velo_clusters_t,
  velo_masked_clustering_t,
  velo_calculate_phi_and_sort_t,
  velo_fill_candidates_t,
  velo_search_by_triplet_t,
  velo_weak_tracks_adder_t)
```

# Memory manager

- Recent GPUs have O(10) GB memory available → scarce resource
- Memory allocation is a blocking operation → cannot be done on different streams in parallel

- In Allen, allocate large memory buffer for every stream before event processing
- A custom memory manager assigns memory segments on demand
- Runtime dependencies determined at compilation time
- For 70 algorithms in the sequence, compilation takes less than five minutes

- All algorithms are designed to use as little memory as possible
- For 1000 events, need 340 MB at maximum

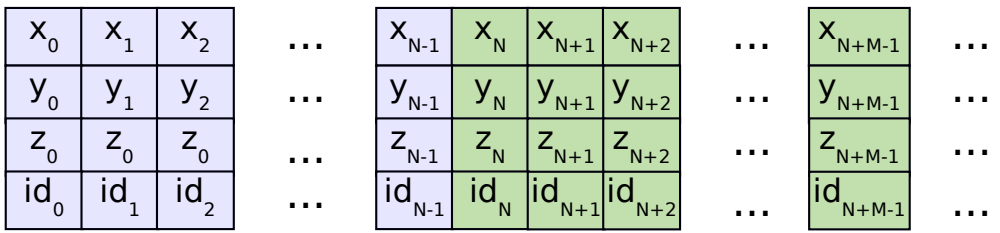
# Event Model

- **Hits**: SoA; allocated size corresponds exactly to the number of hits / clusters in the sub-detector; used for output of clustering / decoding and input for pattern recognition; specific for every sub-detector; lives until consolidation step
- **TrackHits**: used for track candidates; contains indices of hits in the SoA, additional information (i.e. qop) if necessary; specific for every sub-detector; lives only during the pattern recognition and consolidation steps of a sub-detector
- **Consolidated tracks**: after every pattern recognition step: only hits belonging to a track remain in memory together with arrays of track-specific variables, i.e. qop, state; lives after the pattern-recognition step for as long as the track information is needed
- Prefix sum used to calculate total # of hits before full decoding / clustering, # of tracks, # of hits on all tracks for n events

# Hit container: Velo

- On GPU: 32 threads read same data member (possibly of different index) at the same time
- Tracking algorithm: don’t need all hit variables at the same time

  → Use structure of arrays for hit variables
- Velo: pixel detector




Example: Velo hits
module 0 has N hits
module 1 has M hits

- Pre-calculate number of hits in event during decoding / clustering step
- No gaps between sectors / zones

# Hit container: UT & SciFi

- Same layout as for the Velo hits
- Variables used for UT hits:

  silicon strip detector

```
float* yBegin;
float* yEnd;
float* zAtYEq0;
float* xAtYEq0;
float* weight;
uint8_t* planeCode;
uint32_t* LHCbID;
```

- Variables used for SciFi hits:

  scintillating fibre detector

```
float* x0;
float* z0;
float* endPointY;
uint32_t* channel;
uint32_t* assembled_datatype;
```

- Obtain w, dxdy, dzdy, endPointY, yMin, yMax, LHCbID, planeCode from channel, assembled_datatype and the geometry information

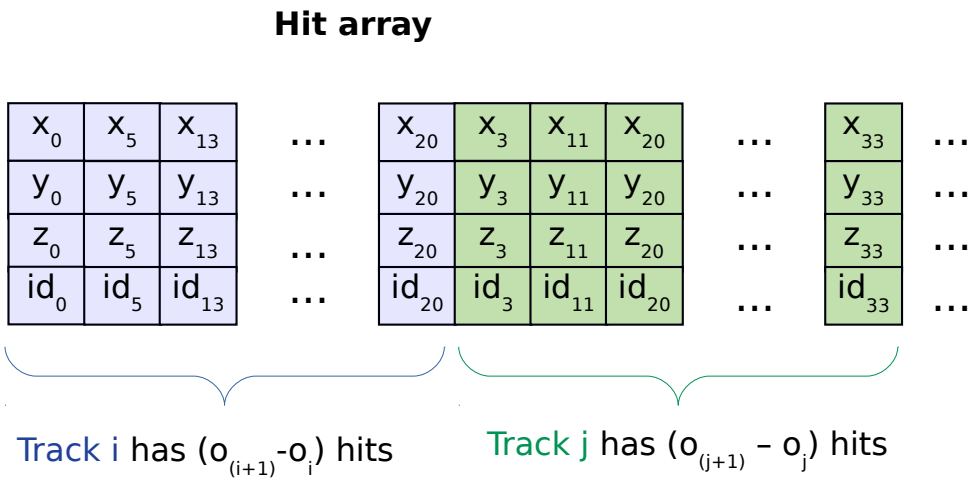
# TrackHits for pattern recognition

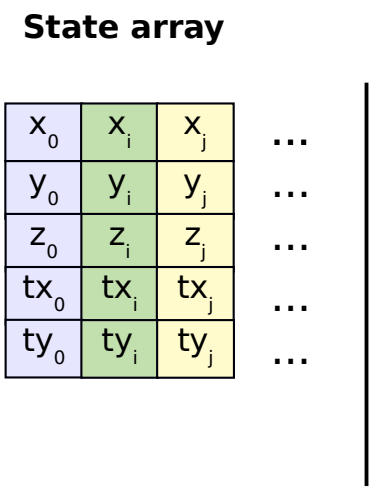
- For candidate tracks in pattern recognition algorithms
- Hit indices: local index of hit within one event stored as short
- Size: 26 hits for Velo, 4 for UT, 12 for SciFi
- Number of tracks not know before pattern recognition

  → safe upper limit is used for allocating the memory
- This could be changed to estimating the number of tracks based on the number of hits in an event

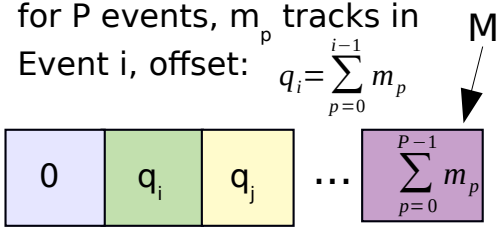
# Tracks after pattern recognition

- After every pattern recognition step: consolidate hits SoA of respective sub-detector to keep only hits that are part of a track
- Tracks object: pointers to hit array, state array, track offset array, track hit offset array, qop...

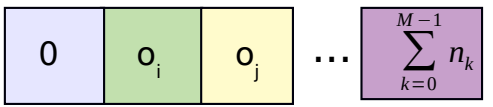



**Track offset array**
for P events, $m_p$ tracks in Event i, offset: $q_i = \sum_{p=0}^{i-1} m_p$




**Track hit offset array**
for M tracks in all events, $n_k$ hits on track k




$$o_i = \sum_{k=0}^{i-1} n_k$$

# Track from several sub-detectors

- Same style track container for every sub-detector
- Example: UT track includes index to Velo track

  → can use Velo track container to look up hits / states
- Use `number_of_events` and `current_event_number` to access correct tracks

```
Struct UTTracks {
    uint* track_offsets;
    uint* track_hit_offsets;
    uint* velo_track_indices;
    float* qop;
};

UT::Consolidated::Hits ut_hits_on_tracks;
UT::Consolidated::States ut_states;
```

# How to add an algorithm in Allen

- Follow instructions in readme:
  https://gitlab.cern.ch/lhcb-parallelization/Allen/blob/master/contributing.md

# Running on openlab server (CERN account)

- `ssh username@olquanta1.cern.ch`
- Setup as recommended in Allen readme:

```
source /cvmfs/sft.cern.ch/lcg/views/setupViews.sh LCG_95 x86_64-centos7-gcc7-opt
export PATH=/cvmfs/sft.cern.ch/lcg/contrib/CMake/3.14.2/Linux-x86_64/bin:$PATH
export PATH=/usr/local/cuda/bin:$PATH
```

- Compile with Cmake:

```
mkdir build
cd build
cmake -DCUDA_ARCH=COMP ..
make
```

- `CUDA_ARCH=COMP` will compile for the highest available compute architecture
- Input data location is specified with the -f option
  - Bs→ Phi Phi (for efficiency studies): `/data/gligorov/WorkshopDatasets/bs2phiphi/allen/bs2phiphi/`
  - Minimum bias data (for throughput studies): `/data/gligorov/WorkshopDatasets/minbias/allen/minbias`