Core software challenges of the GPU High Level Trigger 1 of LHCb

Daniel Hugo Cámpora Pérez, on behalf of the LHCb Collaboration
dcampora@cern.ch

ACAT, March 14th, 2019

Universidad de Sevilla
CERN

The LHCb Upgrade

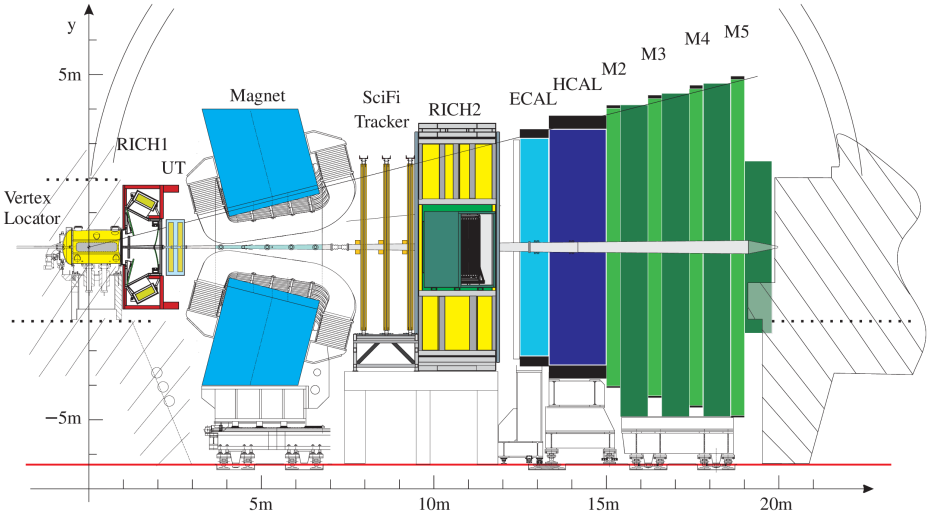
The LHCb detector and Data Acquisition system will be upgraded to prepare for a new data taking period in 2021. The design of the upcoming trigger[1] will be challenging due to two factors:

- LHCb will remove its hardware level trigger, turning to a full-software trigger
- Luminosity will increase to $2 \times 10^{33} \mathrm{cm}^{-2} \mathrm{s}^{-1}$

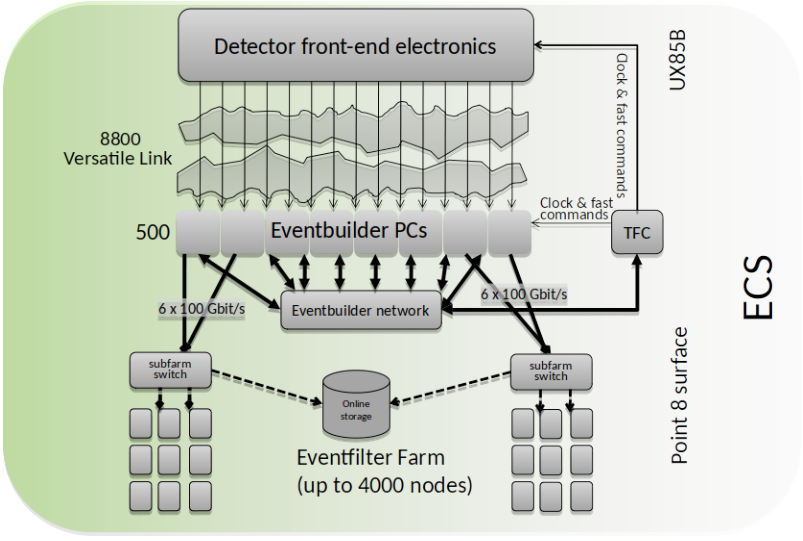
LHCb will filter data at a rate of 40 Tbit/s in a software trigger

[1] *LHCb Trigger and Online Upgrade Technical Design Report*, https://cds.cern.ch/record/1701361

A refresher of the LHCb detector layout



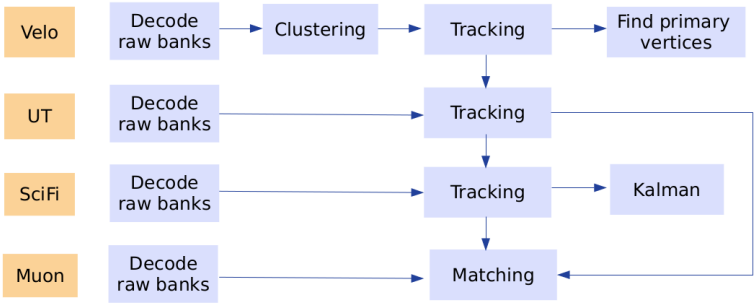
LHCb Online System Run 3



High Level Trigger 1 at LHCb

The first stage of software trigger, also known as *High Level Trigger 1*, is a critical stage of the software reconstruction. It must make a decision in real-time over all input data, at the collision rate.

The entire HLT1 involves the decoding, clustering and track reconstruction of all tracking detectors at LHCb, as well as the Kalman filter, PV finder and trigger decision algorithms.



A GPU HLT1 for the LHCb Upgrade

We are exploring various avenues for dealing with the increase in data rate. GPUs have shown to deliver good performance for scientific workloads in recent years.

GPUs have been used to accelerate parts of other experiment's reconstruction. Our objective however is to propose the possibility of running the *entire HLT1* on GPUs. This design decision would have several benefits:

- *Reduced memory footprint*: Data is kept in GPU memory between kernel executions. Pipeline hides data transmission times.
- *Adequate workload*: LHCb event size is very small, $O(50)$ KiB. Therefore, thousands of events are required to be run in parallel to fully utilize accelerators.
- *Compact solution*: Accelerators can be placed at various places in the DAQ system. Strategically placed, this can allow for cost savings in the full trigger system.

Allen

The Allen framework

The *Allen* framework is a compact, scalable and modular framework, built for running the LHCb HLT1 on GPUs.

Requirements

- A C++14 compliant compiler
- CUDA v10.0

Features

- Configurable static sequences
- Pipelined stream sequence
- Custom memory manager, no dynamic allocations
- Support of MDF and custom binary input format
- Built-in validation with Monte Carlo
- Compilation with nvcc / llvm backend
- Optional compilation with ROOT for generation of graphs

Algorithm design

All algorithms in Allen are by design **SIMD**. Conventional GPU algorithms require parallelism at two levels:

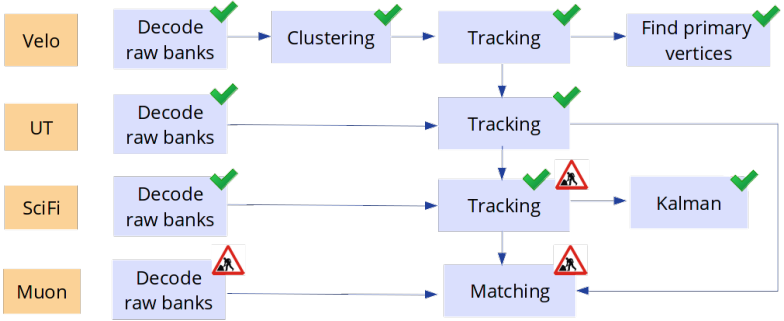
- Blocks - Independent groups of work
- Threads - Workers sharing a common cache and program counter

Every event in LHCb is an independent physics event. Within each event, some algorithms exhibit higher parallelizability than others. Most times, algorithms benefit from the following convention:

- Blocks - *Events under execution*
- Threads - *Intra-event parallelism*

Additionally, all data types are consolidated into Structure of Arrays (SOA). Any SIMD architecture would benefit from this design. Targetting multiple architectures will require additional effort at the algorithmic infrastructure level, but *not at the algorithmic level*.

A codebase under development



Computing challenge #1: Scalability and modularity

As the codebase of Allen grows, the underlying framework should be scalable and accessible, requiring as little framework-specific knowledge as possible, and using common practices where possible.

Sequence configuration is done at compile time in Allen. Adding / removing an algorithm is as easy as modifying one line in a sequence configuration file.

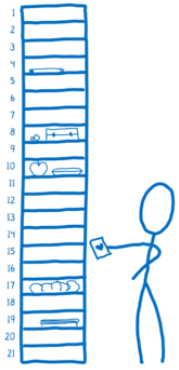
```
SEQUENCE_T(
  velo_estimate_input_size_t,
  prefix_sum_velo_clusters_t,
  velo_masked_clustering_t,
  velo_calculate_phi_and_sort_t,
  velo_fill_candidates_t,
  velo_search_by_triplet_t,
  velo_weak_tracks_adder_t)
```

Computing challenge #2: Memory management and many-event execution

- Memory available per core is very scarce on GPUs
- Due to tiny LHCb event size, must execute thousands of events in parallel
- Memory allocation / deallocation is a blocking operation on GPUs

In Allen, we allocate memory at the **startup** of the application. A custom memory manager allocates and frees memory on demand.

Developers don't have to invoke the memory allocation routines. Compilation will resolve all argument dependencies, and determine when must arguments be allocated / deallocated. This static analysis is not that expensive: It currently takes less than 1 minute to compile Allen.

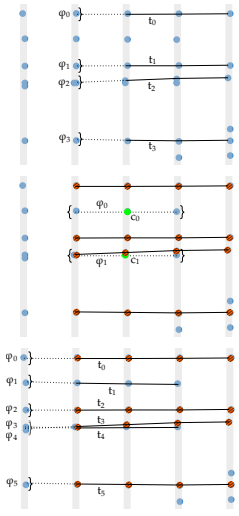


Computing challenge #3: Spatial and temporal locality, data access patterns

- SIMD architectures benefit from coalesced and contiguous data access patterns
- Cache memory is limited in size
- Locality: Access patterns should restrict to a portion of memory

Search by triplet employs an SOA data structure for the VELO reconstruction, so that every access to memory has an increased probability of returning several required data in a cache line.

Additionally, modules in the VELO subdetector are visited only once, interleaving *seeding* and *forwarding* for all building tracks, maximizing spatial and temporal locality.

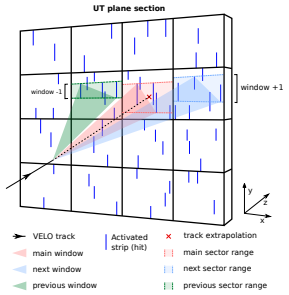


Computing challenge #4: Spatial reductions

- Track reconstruction typically presents a high multiplicity of hit candidates
- Spatial reductions like KD-tree structures or search windows help reduce the dimensionality of hits under consideration

The UT subdetector is partially decoded into *sector groups*, aka groups of sectors sharing the same starting *x* coordinate. Within each sector group, hits are ordered by their *y* coordinate.

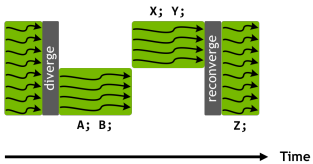
CompassUT determines search windows for each incoming Velo track. A configurable number of windows is determined, and binary searches are performed over *x* and *y*.



Computing challenge #5: Even workload across cores, reduction of branches

General Purpose GPU execution is masked: If the operation should be performed on that particular thread, it is commited - otherwise, it isn't. Attaining a low IPC in the application requires an even workload across groups of cores (*warps*).

```
if (threadIdx.x < 4) {
    A;
    B;
} else {
    X;
    Y;
}
Z;
```



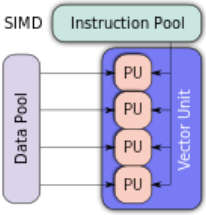
Beneficial to SIMD processors in general

The above fundamentals apply not only to GPU processors.

We have translated part of our reconstruction to demonstrate this principle. The ISPC compiler employs the SPMD programming model in a similar manner to the SIMT model present in GPU computing.

Our translation can target any vectorization-capable CPU, with a configurable vector-width at compile time. Our algorithm design is not specific for GPUs, but **benefit any SIMD processor**. Compatibility with x86-64 processors can be achieved with a low-effort translation.

`https://ispc.github.io`



Results

Methodology

For all results shown, Allen was configured with the following options:

- Sequence with Velo, PV, UT and decoding of SciFi
- Number of events: 4000 *minbias*
- Number of streams: 8
- Number of repetitions: 200

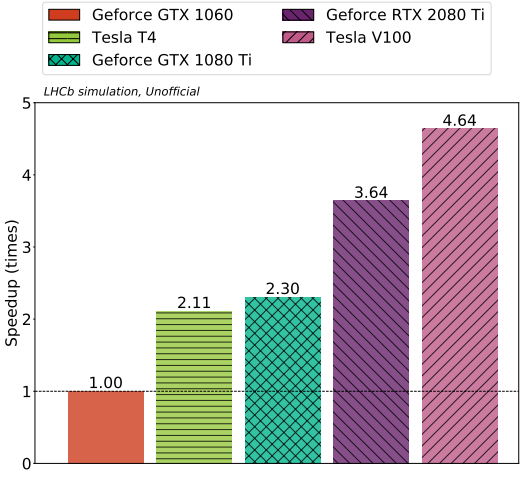
We are looking at various GPUs. Here is a table with their main features:

Feature	Geforce GTX 1060	Geforce GTX 1080 Ti	Geforce RTX 2080 Ti	Tesla T4	Tesla V100
# cores	1280 (CUDA)	3584 (CUDA)	4352 (CUDA)	2560 (CUDA)	5120 (CUDA)
Max freq.	1.81 GHz	1.67 GHz	1.545 GHz	1.59 GHz	1.37 GHz
Cache (L2)	1.5 MiB	2.75 MiB	6 MiB	6 MiB	6 MiB
DRAM	5.94 GiB GDDR5	10.92 GiB GDDR5	10.92 GiB GDDR5	16 GiB GDDR6	32 GiB HBM2
CUDA capability	6.1	6.1	7.5	7.5	7.0
TDP	120W	250W	250W	70W	250W

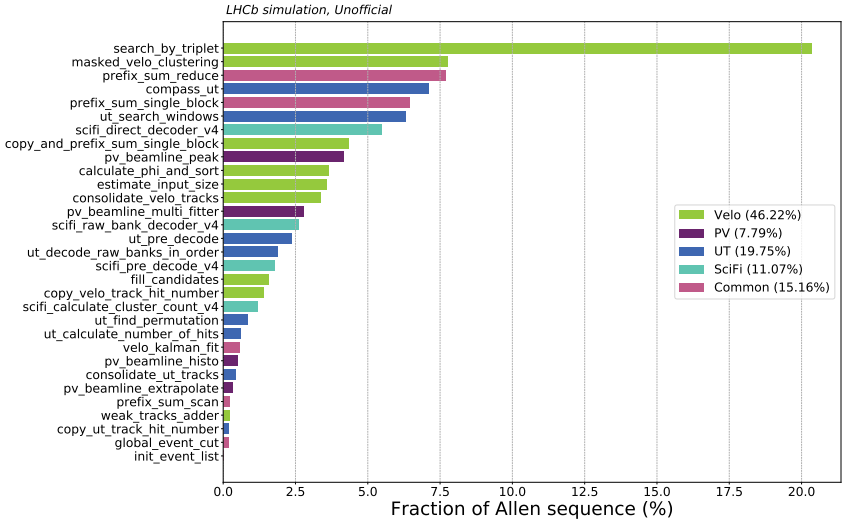
Computing performance on all architectures

Peak performance on all platforms

- Geforce GTX 1060: 24 kHz
- Geforce GTX 1080 Ti: 56 kHz
- Geforce RTX 2080 Ti: 88 kHz
- Tesla T4: 51 kHz
- Tesla V100: 112 kHz

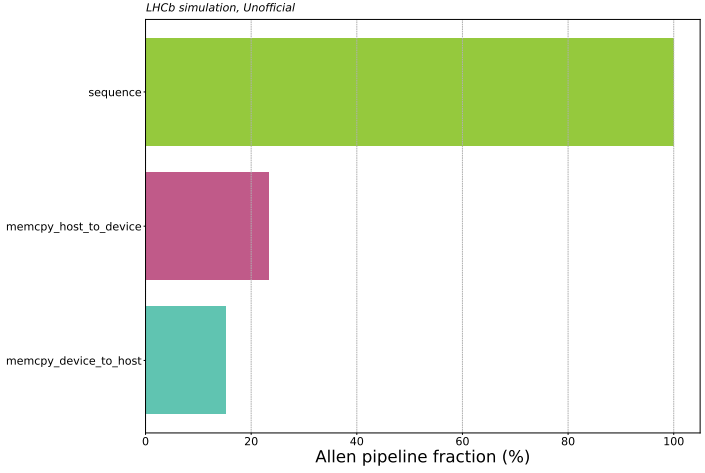


Breakout of performance

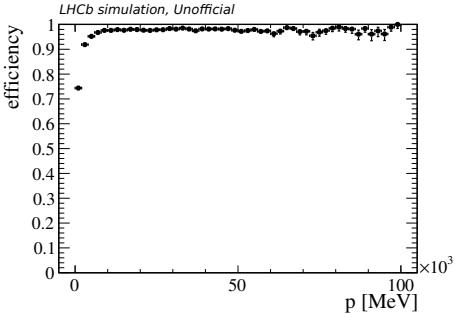


Pipeline timing

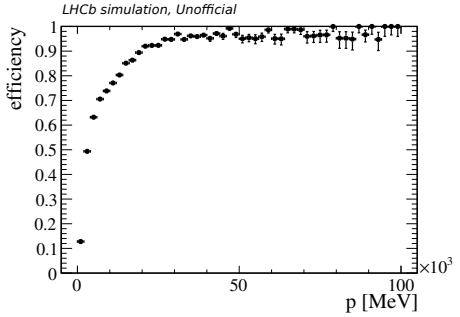
All data transmissions are hidden behind the sequence execution



Physics performance: Velo and Velo+UT



Velo



Velo+UT

- Velo window: 0.062 radians
- UT threshold: $p > 1.5\text{GeV}$, $p_t > 300\text{MeV}$
- Efficiencies are similar to baseline reconstruction

Conclusions

A GPU HLT1 framework: Allen

- Allen, a GPU HLT1 framework, has been presented
- We are close to completing the full LHCb HLT1 sequence on GPU
- Our software scales well to new iterations of hardware
- Our framework is modular, allowing parallel ongoing developments without conflicts

- Physics performance is acceptable and matches our requirements for the Upgrade
- We are evaluating a variety of graphics cards
- Computing performance and development cycle has been extremely successful so far

Future work

- Muon ID and SciFi tracking are under development
- Stress tests should be done on a slice of the system
- The full system should be integrated with the Online system

Thanks!

Thanks a lot to all people involved in the development of Allen!

`https://gitlab.cern.ch/lhcb-parallelization/allen`