# Scheduling algorithms in LHCb's upgrade software framework

technische universität dortmund

Federal Ministry of Education and Research

N. Nolte [1,2], S. Stahl [2]

J. Albrecht [1], E. Govorkova [3], C. Hasse [1,2], R. Matev [2], G. Raven [3]

[1] TU Dortmund, [2] CERN, [3] Nikhef

On behalf of the LHCb Collaboration

## LHCb in Run 3 [CERN-LHCC-2018-007. LHCB-TDR-017]

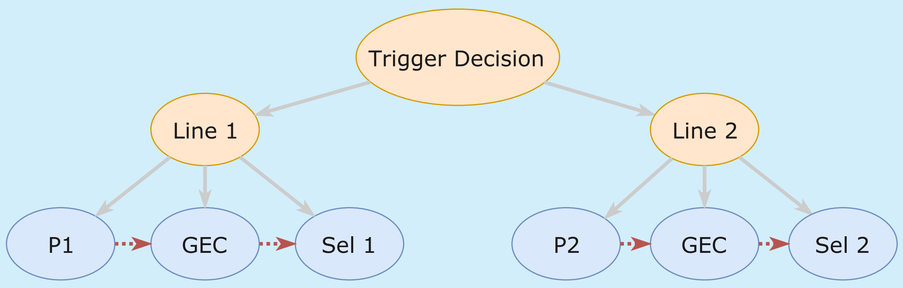
- Single arm forward spectrometer specialized on beauty and charm hadrons
- Detector upgrade during LS2 (2018-2020)
- From Run 3 onwards:
  - purely software based trigger system
  - full online reconstruction at 30 MHz

## General design choices

Running a 30 MHz software trigger requires a multithreading friendly framework with low overhead

- Functional data processing in form of reentrant and thread-safe algorithms, explicitly declaring data dependencies
- Parallelization over events
- Data and control flow is configured once before event processing to achieve minimal scheduling overhead during runtime
- Detect unmet dependencies early
- Automate dependency resolution
  - easier configuration
  - less prone to errors
  - no production of unused data by construction

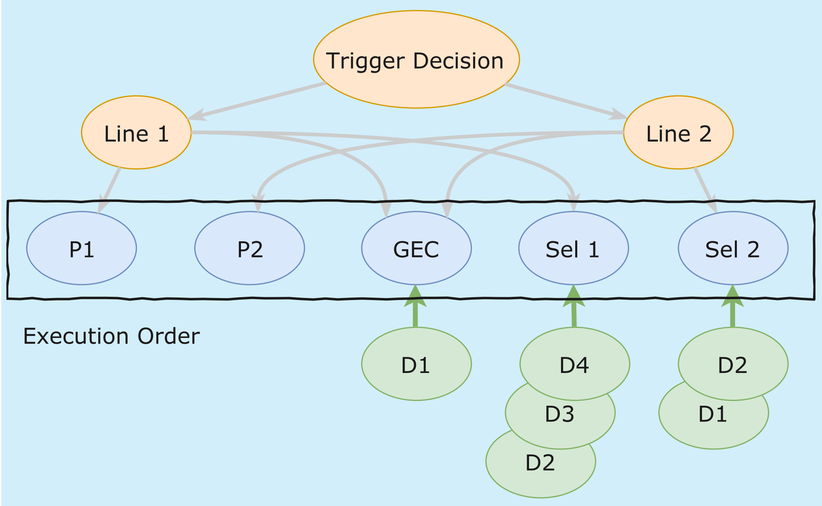
## HLT control flow




LHCb's High Level Trigger (HLT) control flow in a nutshell. Grey arrows indicate a parent-child relation, red arrows represent order constraints (control flow edges). The simplest lines comprise a prescale, a global event cut (GEC) and a selection. The top node decides whether to persist an event.

- **Composite Nodes** (yellow)
  - OR / AND / NOT of childrens' decision
  - optional short-circuiting
  - may constrain order of child evaluation
- **Basic Nodes** (blue)
  - manage single algorithm and keep track of decision
  - execute data dependencies on demand

## Preparation for processing - Configuration




Ordered list of basic nodes with their data dependencies indicated in green and their parents indicated in yellow. The GEC is a shared node.

- The user configures the desired control and data flow by defining a set of nodes and algorithm inputs and outputs
- An ordered list of data dependencies is constructed for each basic node by matching inputs and outputs of all algorithms available
- Basic nodes are ordered into a flat list, taking into account all ordering constraints imposed by composite nodes and manually by the user
- Execution and decision states are prepared for each node and algorithm

## Event processing - Runtime

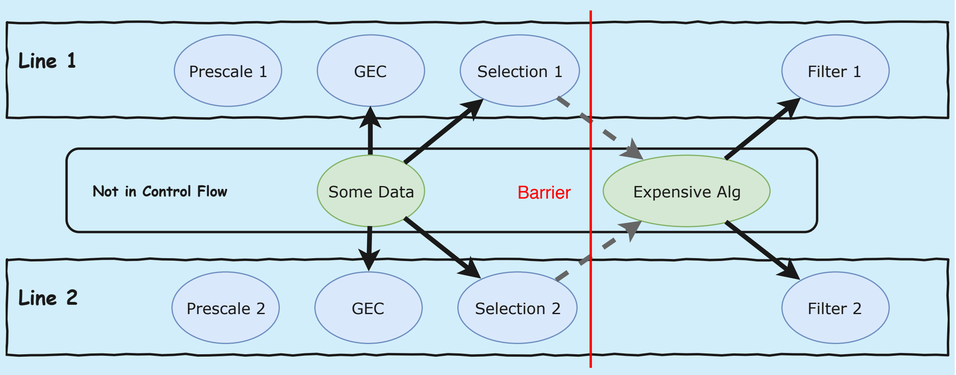
- **Iterate** over the list of basic nodes
- Is the current node **requested** by any parent node, i.e. is any parent node not yet evaluated? If so:
  1. **execute its data producers** in the right order, if they have not been executed yet
  2. **execute basic node itself**
  3. **save** its **decision** and **notify** its **parent** about the outcome
  4. the direct **parent evaluates** its decision if the execution policy permits it
     if short-circuiting is allowed: evaluate as soon as possible
     else: evaluate after the last child has been evaluated
  5. If evaluated, the parents' decision is given higher up the chain, continuing recursively from Step 4.

## Technical aspects

- The master thread prepares self-contained tasks (full events) for a threadpool
  → no thread synchronization
- Fast runtime polymorphism with C++17's std::variant and the corrsponding visitor pattern
- Timing:
  - no measurable overhead in the current upgrade HLT1
  - less than 1% overhead in a mock upgrade HLT1 with 20 trigger lines running at 30 kHz per computing node
  - less than 2% on a mock upgrade HLT2 with 1000 trigger lines

## Barrier - Sharing work

- Resource intensive work may worth be sharing
- Example: Multiple lines select tracks with some intersection
  → invoke the track fit algorithm only once on the union of all track selections
- This requires optional data dependencies, since some lines might have retired before
- Optional dependencies are not pursued in the data dependency resolution, otherwise algorithms might be scheduled unnecessarily
- Instead, the barrier introduces additional order constraints in the configuration step to ensure that all necessary algorithms ran before invoking the expensive algorithm




Required data dependencies are displayed as black arrows. The grey, dotted arrows correspond to optional data dependencies. The "expensive Alg" needs a preceeding control flow barrier to make sure all requested selections have run before it is executed

## Outlook

- Scheduler implemented as default in the LHCb upgrade framework.
- One can define control flow and write upgrade trigger lines, which are processed in multiple threads.
- Performance overhead needs to be tested in the realistic environment of HLT1 and HLT2.
- Benchmark the bookkeeping overhead of barriers
  - Gathering inputs into a union
  - Scattering results to each successing algorithm