Run 3 Selections in HLT2

Alex Pearce

23rd September 2020
LHCb FCC discussion

Introduction

- LHCb has a very broad physics programme
  - Flavour, electoweak, QCD, exotics, heavy ion and SMOG running, …
- “Flavour physics” is very broad in and of itself
  - Hundreds of interesting decay channels, with varying topologies, across many hadron species
  - Spans wide energy range and large set of discriminatory features
- Cannot write a trigger menu for all of this upfront
- LHCb trigger must cater for analysts wanting to capture things not yet considered

- HLT2 performs full, offline-fidelity event reconstruction on ~1 MHz of HLT1 output
  - Must be maximally **efficient** *and* **fast**
- Selects 10 GB/s of data based on decisions on O(1000) trigger lines, mostly exclusive
  - Must be **flexible** and easy to use, inspect, and monitor

Flexible selections

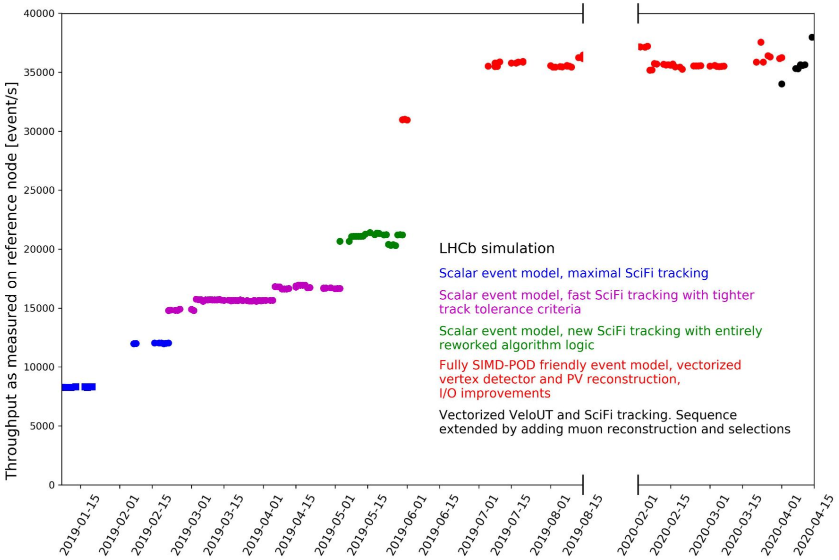
- All LHCb data-processing applications configured by a Python initialisation layer
- Gaudi algorithms written in a way which allows them to be chained together in Python
  - "Filter *these* photons and *those* pions, then vertex them together to form a $D^0 \rightarrow \pi^+\gamma$ candidate"
- Very expressive filtering based on "functors"

```
(PT > 2 * GeV) & (IP < 2 * mm) & (NINTREE(PIDK > 5) >= 1)
```

- Creates a function predicate object: accepts a candidate and returns pass or fail
  - Used to filter sets of candidates and therefore make trigger decisions
- Entire selection-writing lifecycle is in Python!
  - Rapid onboarding of new analysts, simpler debugging, no need for compilation step
- (Sidenote: we use the same functor framework for making ntuples, which means less to learn and reduces inconsistencies between online and offline definitions)

Lessons learnt in HLT1

- Spent around 2 years speeding up HLT1
- Incredible return on investment →
- What did we learn?
  - Maximise CPU cache usage
  - Exploit parallel architectures/instruction sets
  - Minimise pointer and function call indirection
- Integrating these lessons into our selection framework today: ThOr, throughput-oriented functors!
- Similar selection interface, so analysts get speed for free 😎



LHCb-FIGURE-2020-007

No “experts” please

- Majority of HLT2 lines are defined, written, & maintained by the physics working groups
    - Really important for maintainability; do not want a tiny group responsible for all the physics
    - Focuses ‘core’ developers on building a system that is accessible to all, e.g. should not require expert knowledge to maintain throughput

Flexible applications

- Configuration O(1000) selections is non-trivial
- They are *logically* independent, but in practice we want them to share expensive pieces such as the reconstruction
- Also want to process multiple events at once in multi-threaded applications
- Developed a new scheduler to cleanly separate two concepts:
  - Control flow, which ultimately determines the trigger decision
  - Data flow, which algorithms must be run to satisfy data dependencies
- Individual algorithms idempotent, acting on immutable inputs
- New configuration logic deployed in parallel, PyConf, encourages small, encapsulated configuration fragments to increase debuggability: super helpful in an online setting

Considerations for future experiments

- We're making these changes after very successful operation since 2010
    - Lots of lessons learnt from a specific set of experiences
- Optimising for runtime speed and configuration safety and clarity
- Experiments in an exploratory phase have very different priorities and so may benefit from different or hybrid approaches
- Still, we think we're doing a lot of cool things and would love to see others digging it too!

End