

Implementation of the ATLAS trigger within the multi-threaded AthenaMT framework

Stewart Martin-Haugh
on behalf of the ATLAS collaboration

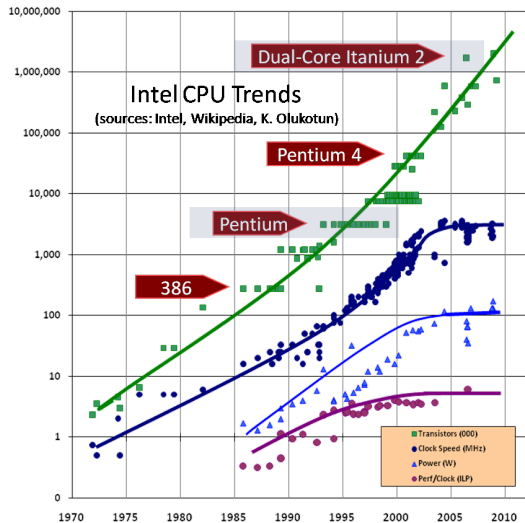
CHEP 2018, Sofia
11 July 2018



Science & Technology
Facilities Council

Hardware trends

- ▶ Moore's law stopped helping us some time ago
 - ▶ 2003 ATLAS TDAQ TDR estimated 8 GHz dual-core machines
 - ▶ In practice, ended up with multi-core 2.3 GHz machines
 - ▶ Memory per core has decreased
- ▶ Need multi-threading to make memory-efficient use of many cores

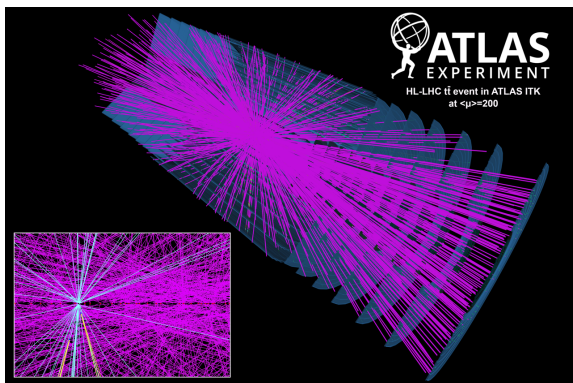


Source: [Herb Sutter](#)

LHC upgrade plans

- ▶ 2018 is last year of LHC Run 2
- ▶ Long shutdown 2 (LS2) 2019-2021
 - ▶ At least $1.25 \times$ design luminosity
 - ▶ Move from 13 - 14 TeV
 - ▶ Increased pileup
- ▶ Multi-threading would already be useful now
- ▶ Vital for Run 3

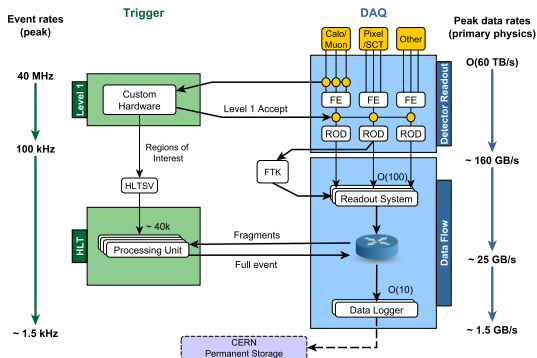
LHC upgrade plans



- ▶ For Run 4, expect factor of 4-10 increase in HLT **input rate**
- ▶ $2.5\text{-}3.5 \times$ Run 2 instantaneous luminosity: ≈ 200 simultaneous collisions
- ▶ Retain Run 3 software infrastructure, optimise
 - ▶ May also need to support FPGA, GPU etc
- ▶ See talk by [I Riu](#)

ATLAS Trigger/DAQ in Run 2

- ▶ $\approx 50,000$ commodity CPUs in High Level Trigger (HLT) reducing event rate from 100 kHz to 1.5 kHz
- ▶ Single-threaded AthenaHLT framework running over multiple events per node (multi-process)
- ▶ Regional reconstruction in geometric regions of interest (RoIs) to reduce processing time, some full-event processing at reduced rate
- ▶ See talk by [A Martyniuk](#)



ATLAS Trigger/DAQ in Run 2

- ▶ ATLAS offline software (Athena) uses Gaudi (common with LHCb) scheduler to control algorithms
- ▶ HLT-specific component (“steering”) controls execution of algorithms
- ▶ Extra steering component functionality:
 - ▶ Regional reconstruction
 - ▶ Early rejection
 - ▶ Decision recording
 - ▶ Online monitoring
 - ▶ Running all triggers on accepted events: “rerun”

ATLAS Run 3 Trigger/DAQ upgrade

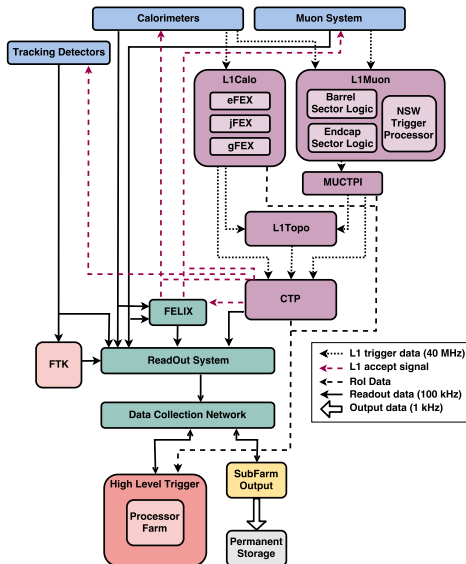
New/upgraded components

- ▶ New Small Wheel (improved muon chambers for trigger)
- ▶ Readout: Front End Link Exchange (FELIX)
- ▶ Significant Level 1 Calo hardware trigger upgrade

HLT implications

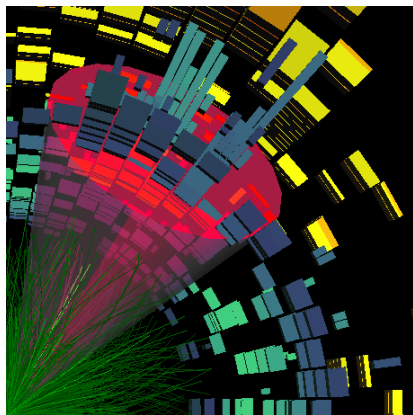
- ▶ Support output from new subsystems
- ▶ Use Fast Tracker (FTK) preprocessor to reduce CPU cost of tracking, allow full event tracking (see T. Seiss [talk](#))

Ready for 2021



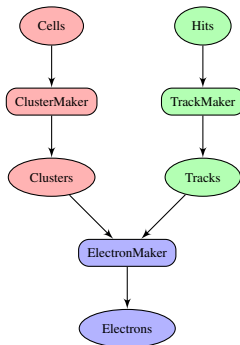
Software in Run 3

- ▶ **AthenaMT (Multi-Threaded)** framework
- ▶ Design from start with HLT and offline in mind
 - ▶ Share offline algorithms more easily, reduce custom components (e.g. steering layer)
- ▶ HLT requirements
 - ▶ Data flow: scheduling of algorithms
 - ▶ Control flow: early rejection
 - ▶ EventViews: regional reconstruction
- ▶ Significant code rewrite to remove reliance on steering layer



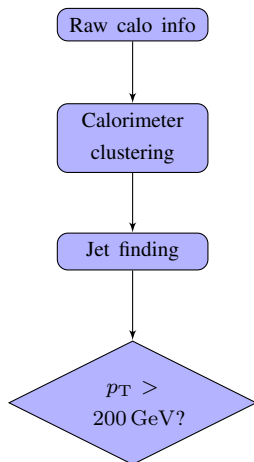
Gaudi Parallel Scheduler

- ▶ Gaudi Parallel Scheduler: uses Intel Threading Building Blocks to create tasks etc
- ▶ TBB handles low-level implementation details
- ▶ Data dependencies expressed with Read and WriteHandles
- ▶ Algorithms run in EventViews with transparently restricted access to event data
- ▶ Algorithms can run in parallel if no common dependencies (e.g. track and calo)



- ▶ Chain: Reconstruction, then decision
 - ▶ Each step depends on previous: **data flow**
 - ▶ Decision accept/reject point: **control flow**
- ▶ Generate directed acyclic graph of data dependencies
- ▶ Schedule algorithms in parallel
- ▶ Some algorithms will run in **EventViews**, others (e.g. E_T^{miss}) run globally
- ▶ Run ≈ 1500 chains in trigger menu
- ▶ Use early rejection (stopping chains early) to minimise processing time

Example jet chain



Menu and configuration

- ▶ Run 2 Athena configuration is now \approx 1M fragile lines of Python (cf 4M lines of C++ in Athena)
- ▶ Work underway on a new modular configuration system
 - ▶ Composable fragments, standalone verification
 - ▶ Generates serialisable configuration (probably Python pickle file)



Configuration and database

- ▶ Must be able to configure trigger easily, particularly online (add/remove chains, prescale)
- ▶ Reproducible, permanent storage
- ▶ Run 1/2: relational database of algorithm properties
- ▶ Run 3: Store “blob” in database: python pickle file, JSON
 - ▶ Common configuration with offline: reduce maintenance overhead

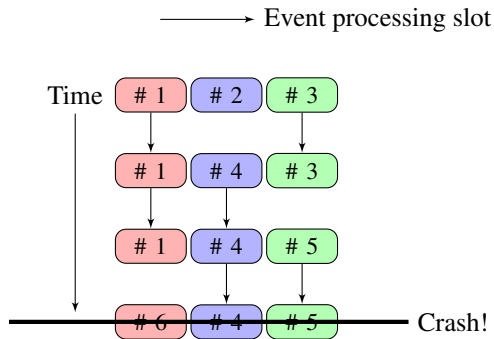
Online integration

Adapt to multi-threading and Phase-I TDAQ changes

- ▶ Run 2: Base process forks multiple processes to run on multiple events
- ▶ Run 3: Same as Run 2 but each forked process is multi-threaded and runs on multiple events

Optimise number of forked processes, threads

- ▶ Events may cross conditions boundaries: handle correctly
- ▶ Must handle crashes, timeouts correctly: store concurrently processed events for debugging (e.g. #4, #5, #6)



Conclusions

- ▶ Work in progress to prepare ATLAS HLT code for multi-threading in AthenaMT framework, ready for Run 3
- ▶ Simplify code, share more with offline, and improve performance
- ▶ Thank you for your attention!

Backup

- ▶ Title slide: [Dalmatian pelican](#)
- ▶ Menu and configuration: [Portlandia](#)