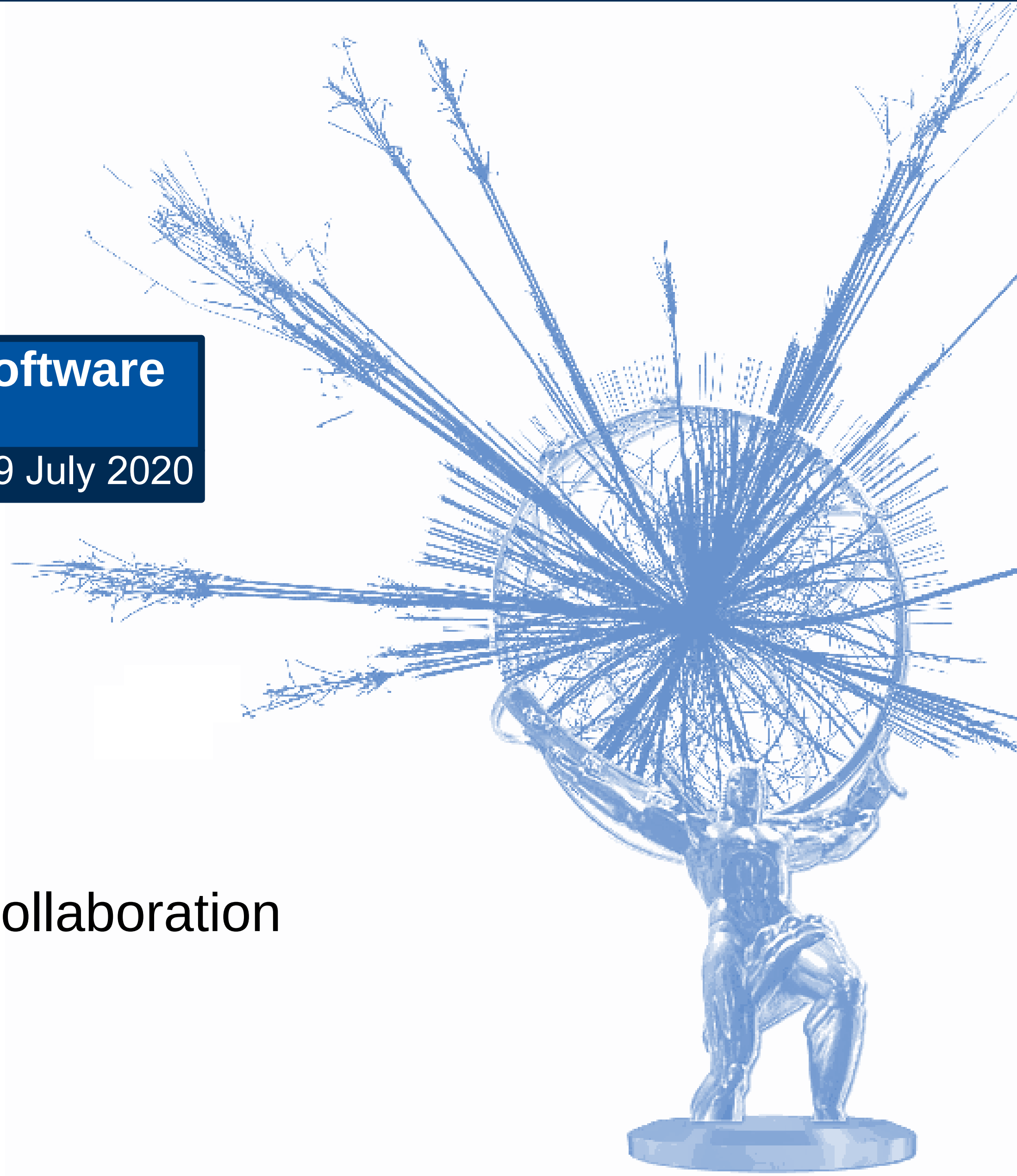
Preparing the ATLAS Trigger Software for Multi-threaded Operation

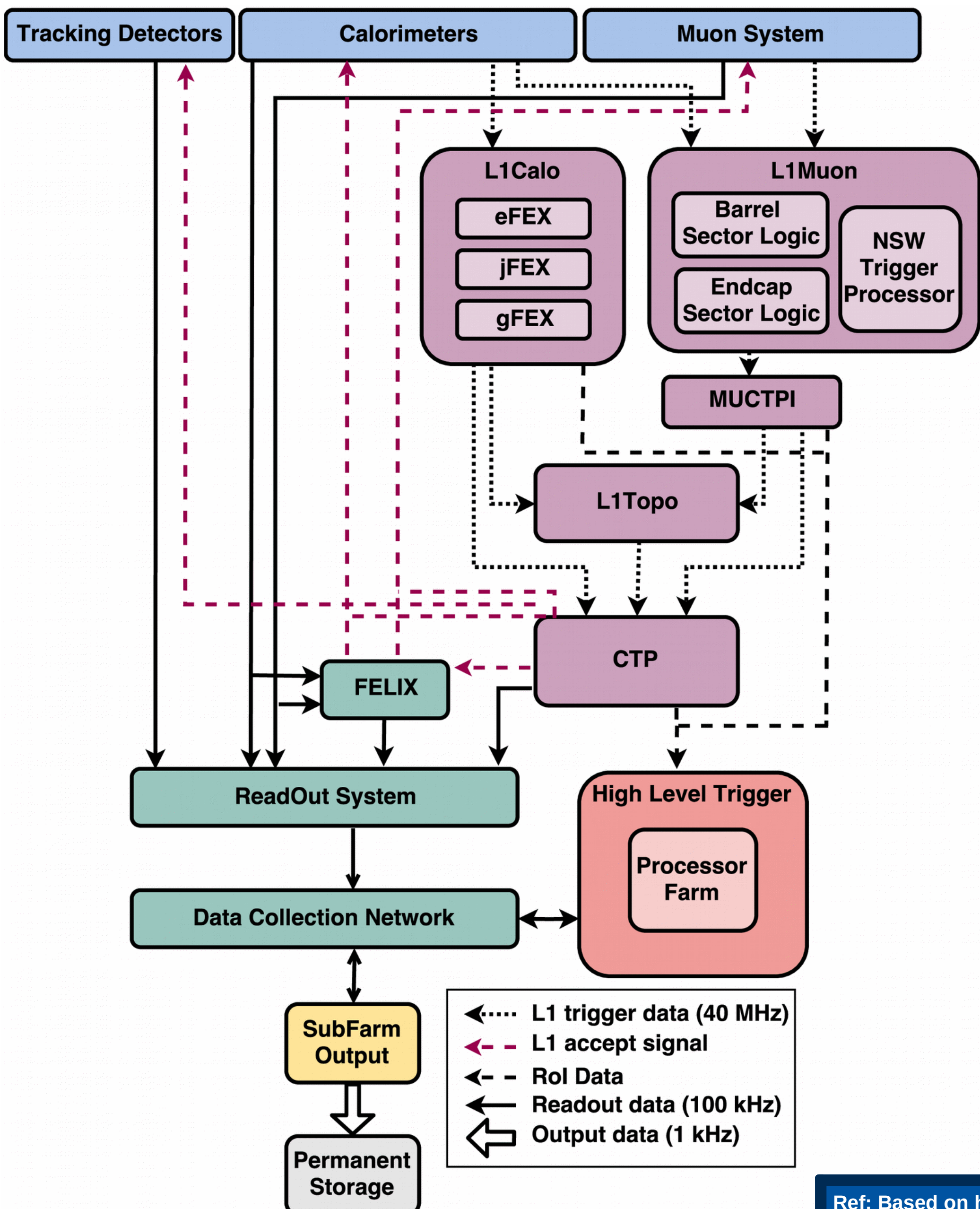
29 July 2020



- ICHEP 2020

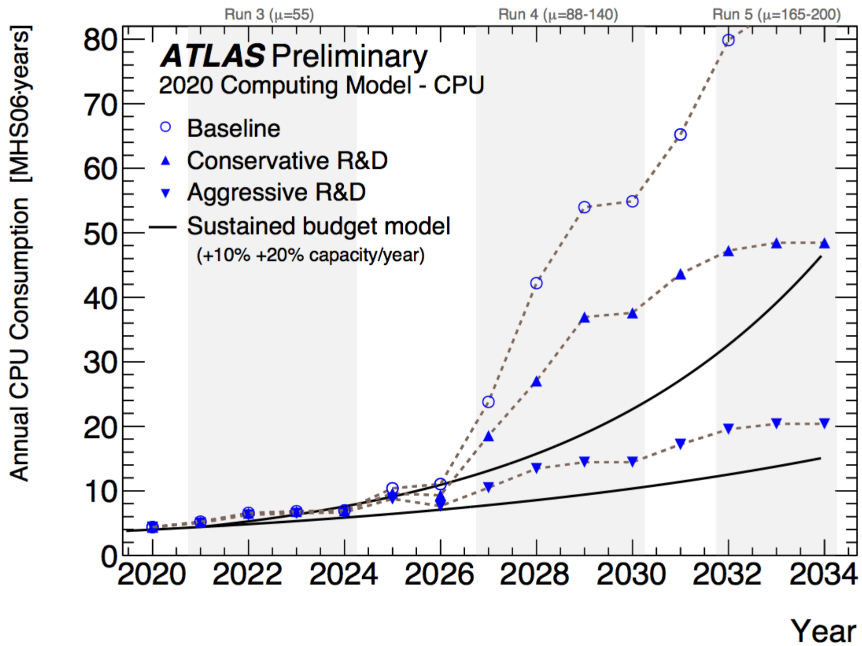
- Mark Stockton
  - CERN
- On behalf of the ATLAS Collaboration

ATLAS data flow overview



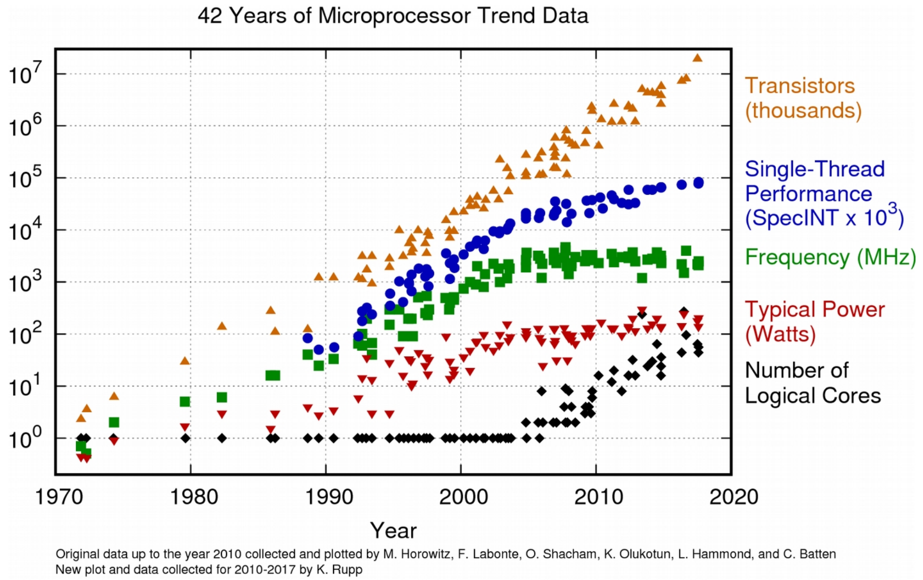
- Overview of the ATLAS data flow for Run 3:
  - Sub-Detectors
  - Level-1 Trigger
  - High Level Trigger (HLT)
  - Data Acquisition (DAQ)
- The HLT farm during Run 2:
  - Consisted of ~40k Processor Units
  - Had a peak input rate of 100 kHz
  - Produced an output rate of 1 kHz on average per LHC fill
- This talk will cover the software running on the HLT
  - Uses Athena Software framework, which is also used for reconstruction, simulation and physics analysis in ATLAS

Why Multi-threaded



- The required CPU to run ATLAS reconstruction will increase dramatically for future LHC data taking (Run 4 and beyond)
- Given modelling of the expected CPU budget clearly minimal (baseline) R&D is not enough to be able to match these requirements
  → start improvements now

- Single thread performance has plateaued in the computing market
- Number of cores is growing, yet memory is not getting cheaper
  → Maximal throughput is limited by the memory per process
- Additionally, with multi-threading the SW could make use of accelerators
  - Using a GPU to process a thread

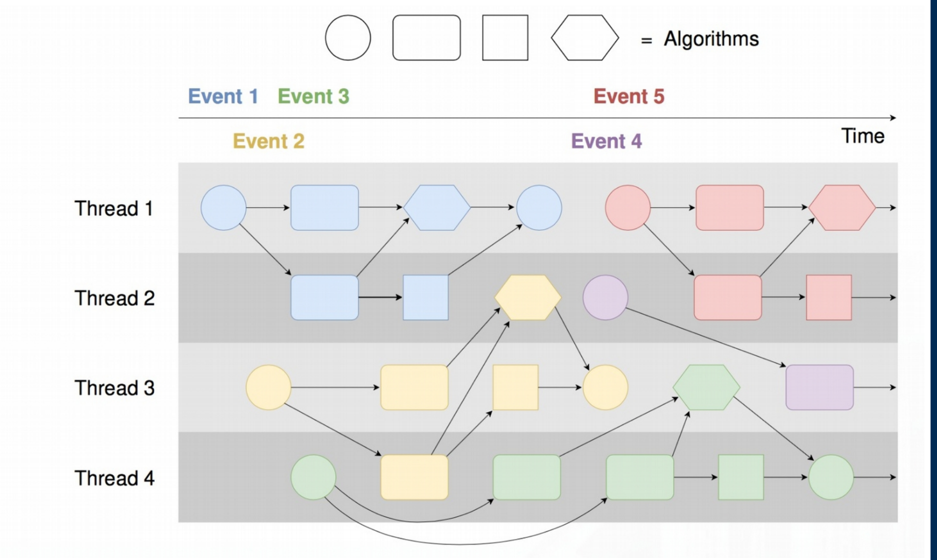


Ref: https://twiki.cern.ch/twiki/bin/view/AtlasPublic/ComputingandSoftwarePublicResults

Ref: https://www.karlrupp.net/2018/02/42-years-of-microprocessor-trend-data/

AthenaMT

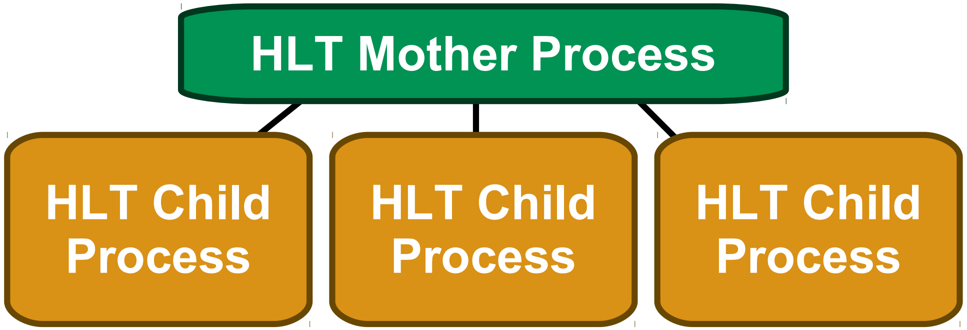
- Athena used Gaudi, which was not designed for multi-threading
  → Design and implement AthenaMT
- Include HLT requirements from the start
  - e.g. partial event data processing
- Includes three types of MT processing
  - Inter-event:
    - Multiple events are processed in parallel
  - Intra-event:
    - Multiple algorithms can run in parallel for an event
  - In-algorithm:
    - Algorithms can utilize multi-threading and vectorisation



- Event processing is managed by:
  - Each algorithm has input and output data dependancies
  - Once inputs are available for an algorithm, GaudiHive Scheduler pushes it into the Intel Threading Building Blocks queue
  - Execution also depends on the configured number of threads and event slots

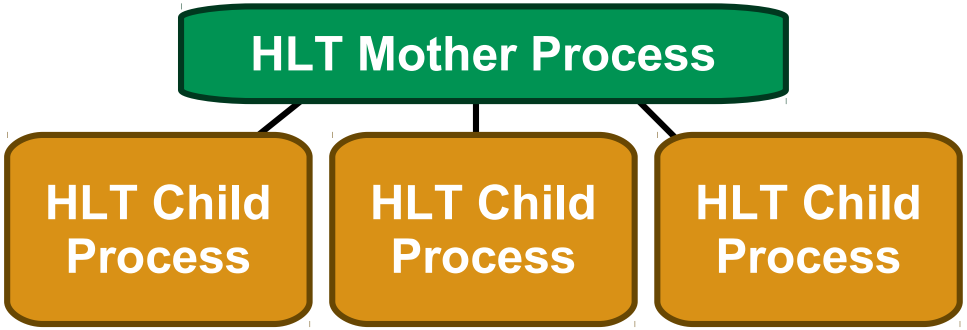
References:
AthenaMT – ATLAS Collaboration, ATL-SOFT-PROC-2017-019
GaudiHive - http://concurrency.web.cern.ch/GaudiHive
Threading building blocks - https://github.com/oneapi-src/oneTBB
Diagram - R. Bielski, ATLAS Collaboration, ATL-DAQ-PROC-2019-004

HLT in AthenaMT



- One Mother process per Processor Unit
- Mother process loads the configuration using Athena/AthenaMT
- From this fork Child processes
  - Mother process handles just the child processes, no events
  - Retain multi-process approach as used in Run 2

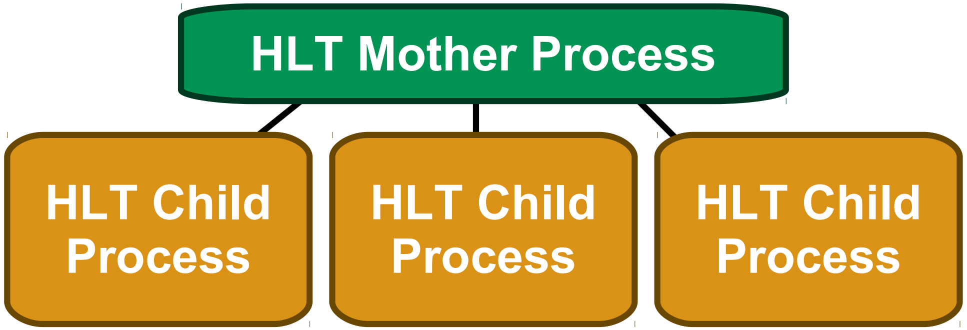
HLT in AthenaMT



- Run 2:
  - Memory saved by using copy-on-write
  - Each child runs single instance of Athena to process events sequentially
  - HLT Child Process drives event loop requesting event to process

- One Mother process per Processor Unit
- Mother process loads the configuration using Athena/AthenaMT
- From this fork Child processes
  - Mother process handles just the child processes, no events
  - Retain multi-process approach as used in Run 2

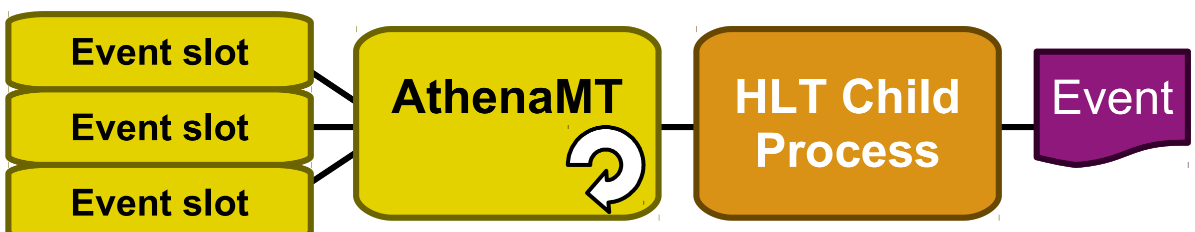
HLT in AthenaMT



- Run 2:
  - Memory saved by using copy-on-write
  - Each child runs single instance of Athena to process events sequentially
  - HLT Child Process drives event loop requesting event to process

- One Mother process per Processor Unit
- Mother process loads the configuration using Athena/AthenaMT
- From this fork Child processes
  - Mother process handles just the child processes, no events
  - Retain multi-process approach as used in Run 2

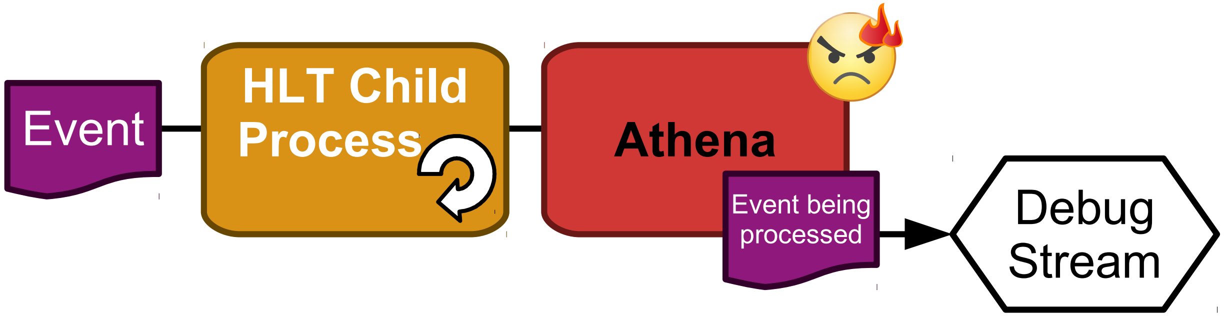
- Run 3:
  - Can now share both read and write memory
  - AthenaMT on HLT Child process now can contain multiple threads and multiple event slots
  - AthenaMT now requests events (via HLT Child process) when it has free processing slots
    - Interfaces to Processor Unit are also changed
  - The offline emulation of this configuration is improved for better development/testing
  - Performance will be optimised by adjusting number of forks, threads and slots



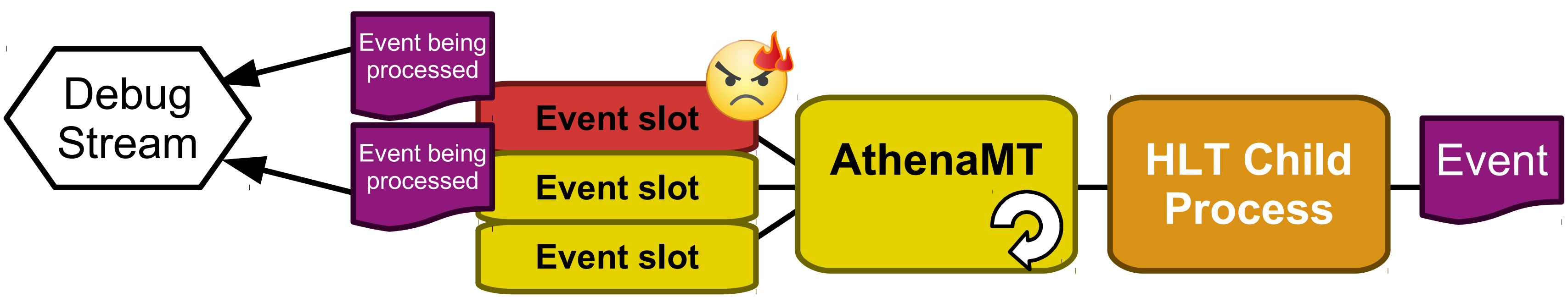
Error handling

- Optimised configuration has to take into account stability not just performance
- If there is a crash in Athena or if the process reaches a processing timeout threshold the event data is “force accepted” to a Debug stream for offline debugging/recovery

- In Run 2 the single event being processed would be written to this Debug stream



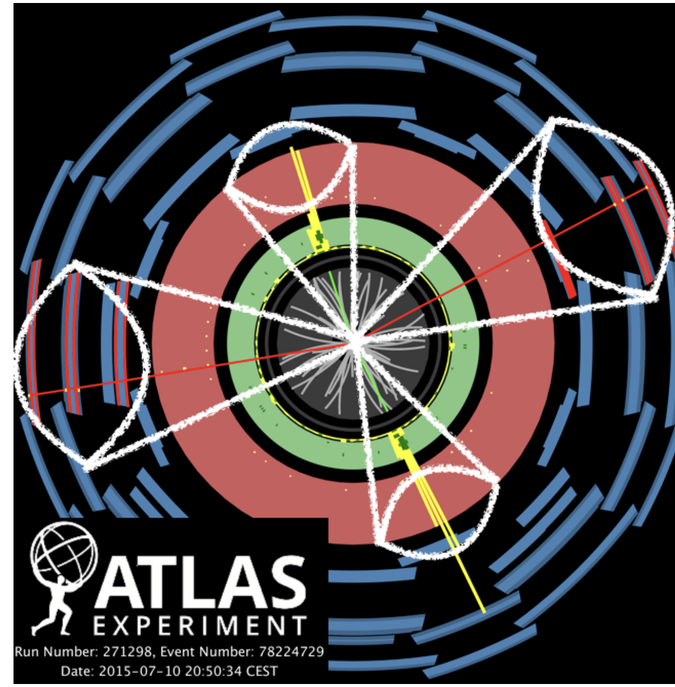
- In Run 3 this applies to all events being processed by the same fork at that time
  - Too many event slots per fork will increase the number of unrelated (potentially good for physics) events in the Debug stream
    - However, the number of threads does not affect the number of events lost



HLT software

- HLT event selection is based on using Chains
  - These are built up of HLT algorithms, which share as much code as possible with offline versions, and hypothesis to test conditions for acceptance
  - Chains are seeded by L1 items
  - In Run 2 ~1500 Chains were active

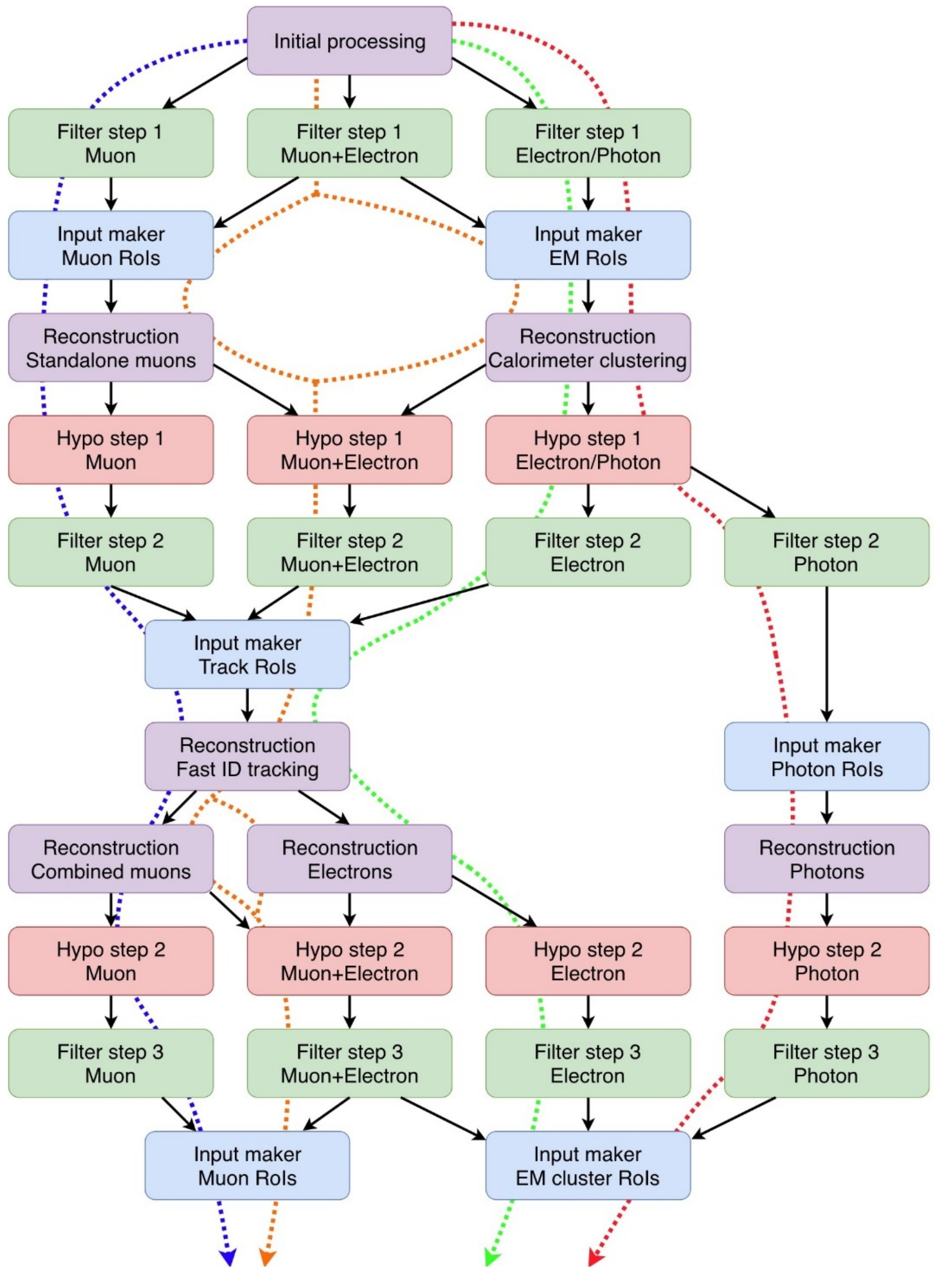
- The HLT takes ~0.5s to process an event, compared with ~30s for offline reconstruction
- Achieve this by:
  - Only reconstruct part of the event (regions of interest)
    - Defined as a cone around the collision point and HLT seeds are provided by the Level 1 trigger
  - Early rejection
    - Early steps in a chain are fast, later steps take longer but provide more detailed analysis

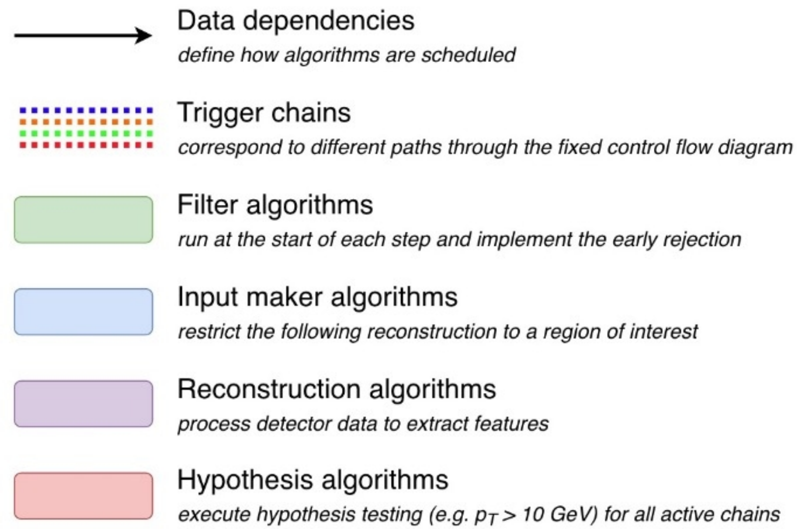


- In Run 2 these features were achieved by custom HLT scheduling and data caching
- For Run 3 the HLT software is rewritten and integrated into AthenaMT
  - Not just aspects related to moving to multi-threading
- Allows better unification with the offline software and the GaudiHive framework
- Add HLT specific extensions:
  - Event views → to provide partial event reconstruction
  - Control Flow → for early rejection

Ref: Based on https://twiki.cern.ch/twiki/pub/AtlasPublic/EventDisplayRun2Collisions/

Processing an event

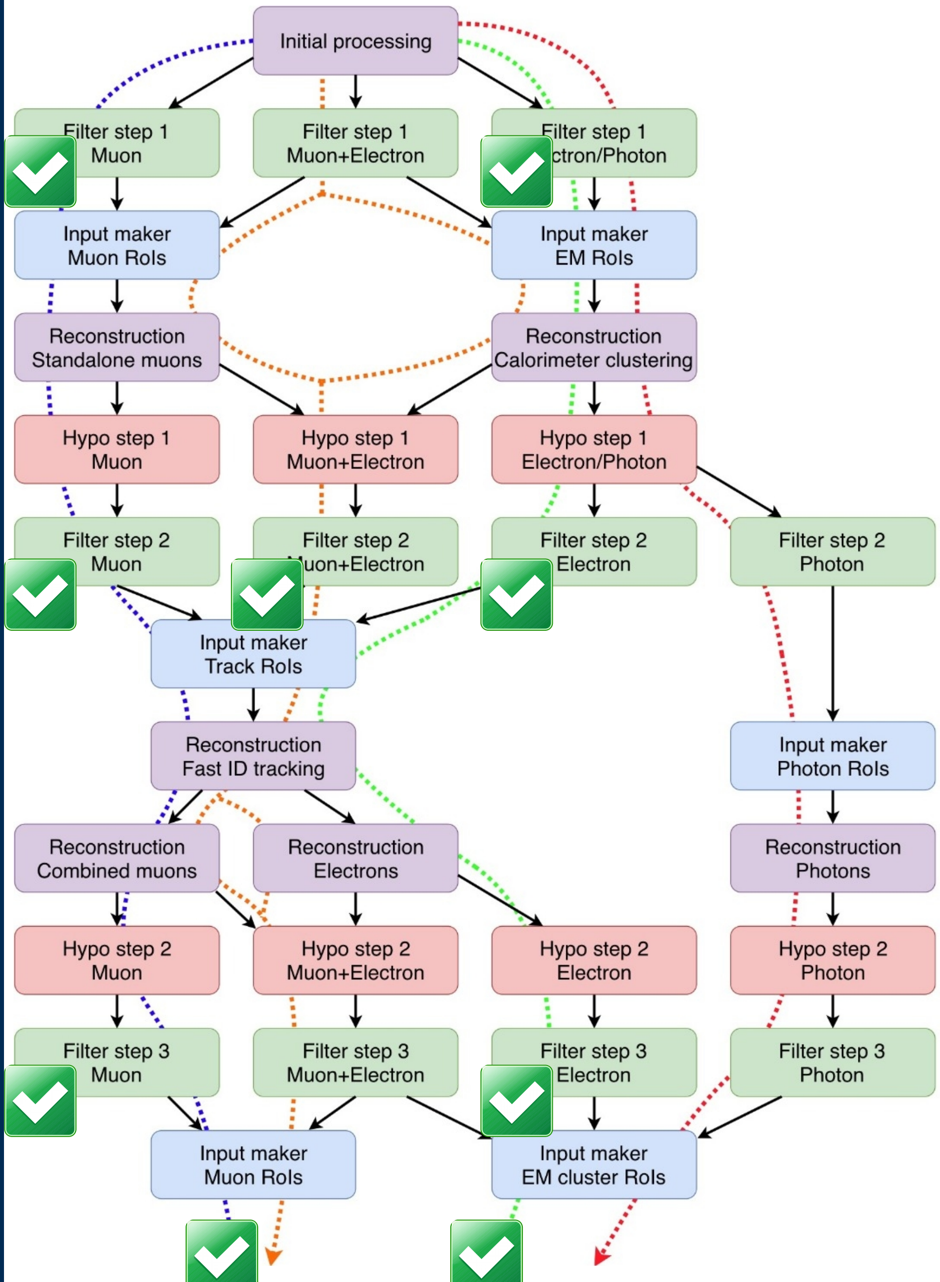




- Control flow graph is created in initialisation
- The steps are then executed based on the data available in an event

Ref: R. Bielski, ATLAS Collaboration, ATL-DAQ-PROC-2019-004

Processing an event



Data dependencies
define how algorithms are scheduled

Trigger chains
correspond to different paths through the fixed control flow diagram

Filter algorithms
run at the start of each step and implement the early rejection

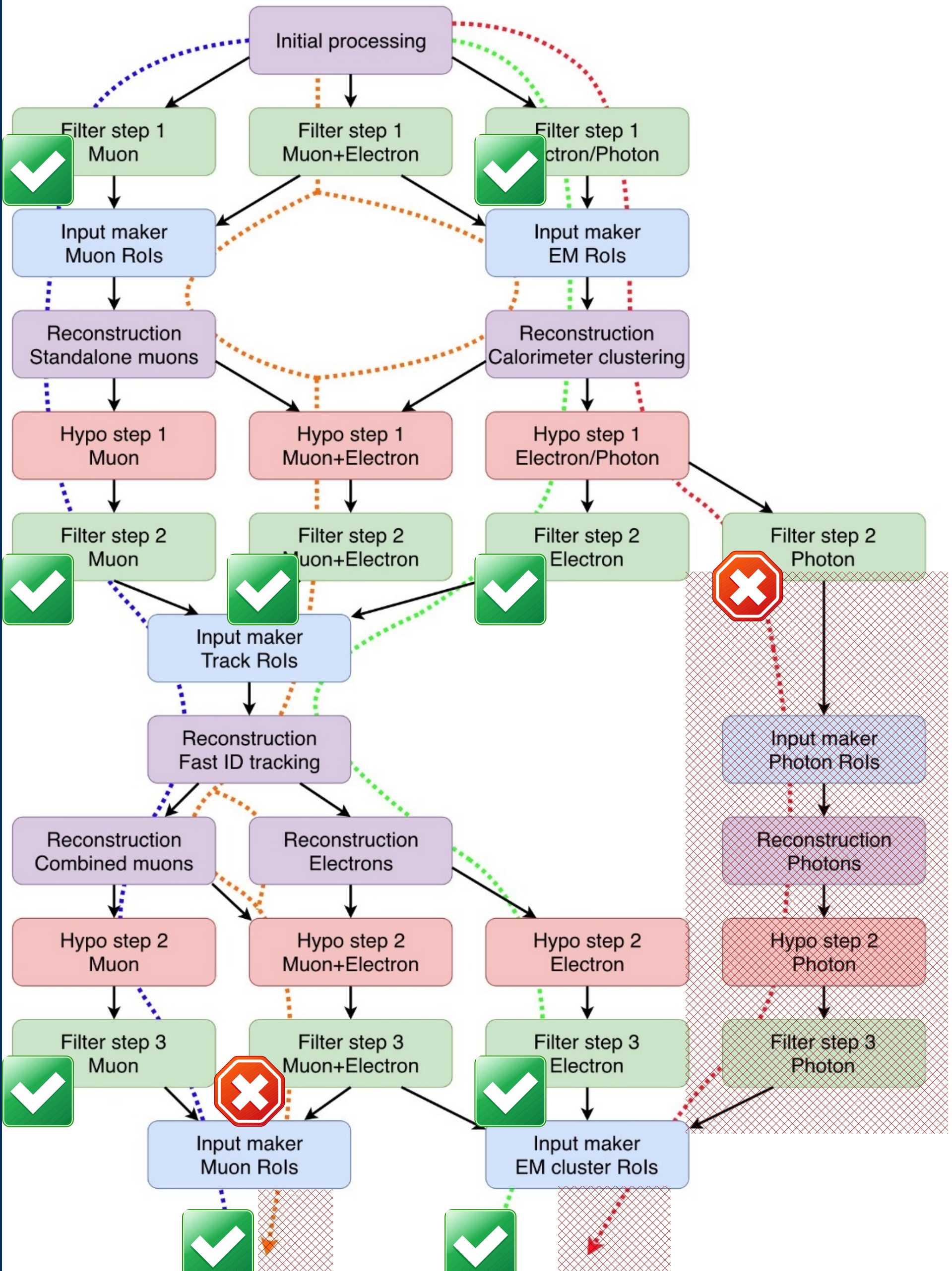
Input maker algorithms
restrict the following reconstruction to a region of interest

Reconstruction algorithms
process detector data to extract features

Hypothesis algorithms
execute hypothesis testing (e.g. $p_T > 10$ GeV) for all active chains

- Control flow graph is created in initialisation
- The steps are then executed based on the data available in an event
- If a filter passes, continue through the next steps

Processing an event



Data dependencies
define how algorithms are scheduled

Trigger chains
correspond to different paths through the fixed control flow diagram

Filter algorithms
run at the start of each step and implement the early rejection

Input maker algorithms
restrict the following reconstruction to a region of interest

Reconstruction algorithms
process detector data to extract features

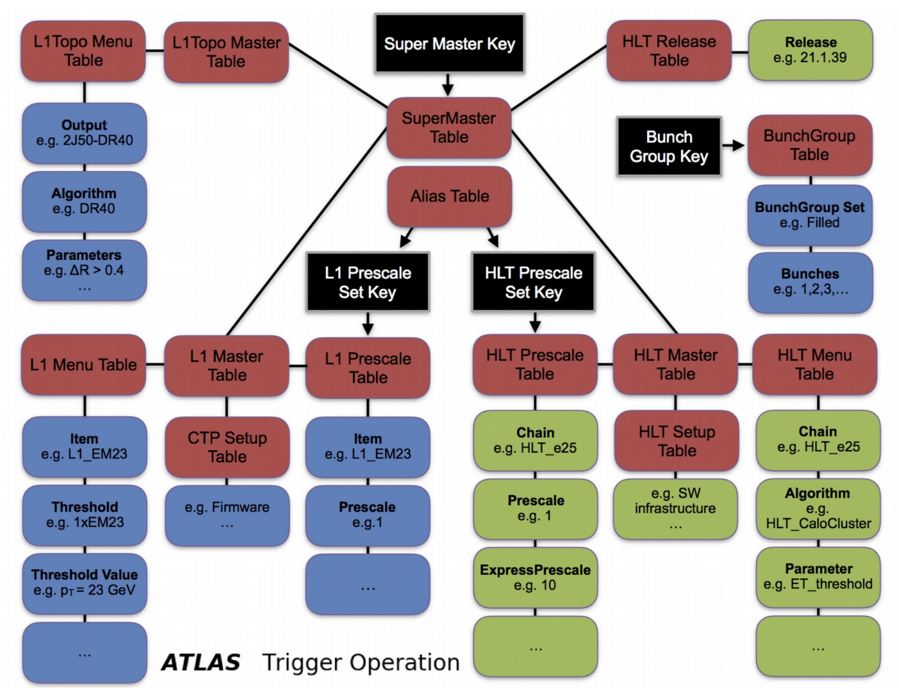
Hypothesis algorithms
execute hypothesis testing (e.g. $p_T > 10$ GeV) for all active chains

- Control flow graph is created in initialisation
- The steps are then executed based on the data available in an event
- If a filter passes, continue through the next steps
- If it fails, stop processing steps
- If reach the last step with a Chain passing all steps, accept the event

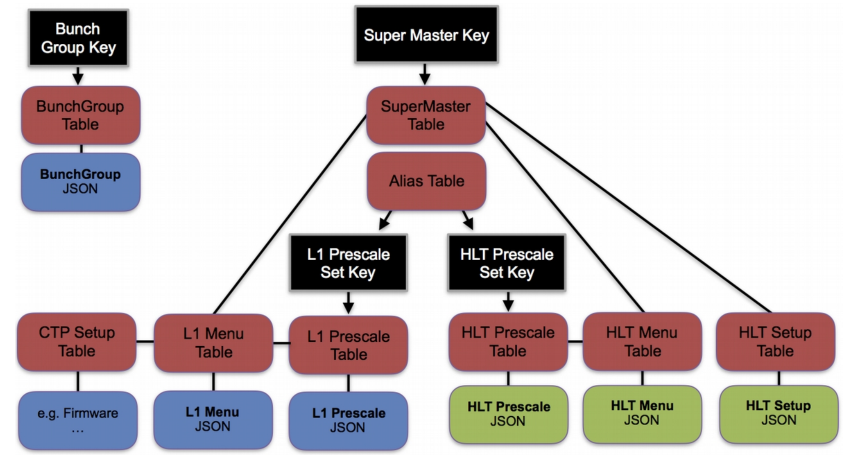
Trigger Database Updates

- The chain configuration is stored in the trigger database (DB) and is loaded during the HLT initialisation

- In Run 2 this DB structure represented a table per object ~90 tables (filled by parsing xml files)
  - Information accessed by four keys, i.e. the primary keys of the relevant parent tables



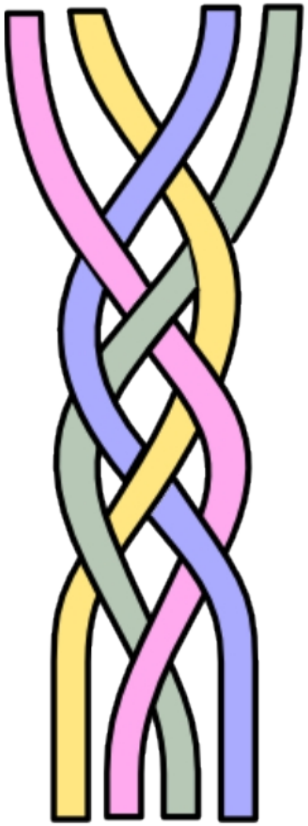
- In Run 3 this structure is simplified as most of the DB schema will be replaced by directly storing JSON files



- Each of these files contains objects holding the information of the previous tables:
  - Makes the DB schema and interaction simpler O(10) tables
  - Easier to extend the files during data taking period rather than updating DB schema
  - Increased data duplication, but reduces time consuming lookups for every object
  - Eliminate another conversion of the data for offline metadata storage (used when processing of events without DB access)

NEW Ref: https://atlas.web.cern.ch/Atlas/GROUPS/PHYSICS/PAPERS/TRIG-2019-04/

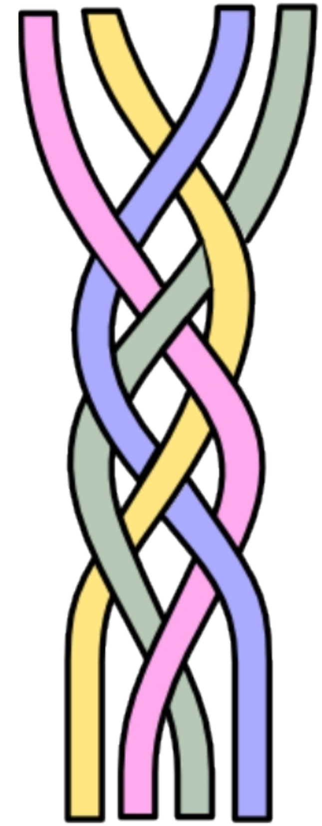
Summary



AthenaMT is being developed to prepare for future data-taking requirements and available computing resources

Core functionalities to be able to run the HLT are in place
Full set of algorithms to deploy Run 3 menu are being developed
Validation campaigns ongoing both offline and online (using MC, Run 2 data or cosmic/random triggers)

Performance studies have started, but the final configuration of MT usage will be measured at the start of Run 3



Other related material presented at ICHEP 2020:

- The ATLAS trigger menu: from Run 2 to Run 3
- Tim Martin (Warwick)

- Triggering in the ATLAS Experiment
- Javier Montejo Berlingen (CERN)