IMPLEMENTATION OF THE ATLAS TRIGGER WITHIN THE MULTI-THREADED ATHENAMT FRAMEWORK

S.Cenk Yıldız
on behalf of ATLAS Collaboration

UCI University of California, Irvine

EPS-HEP Conference 2019, July 10-17, Ghent

OUTLINE

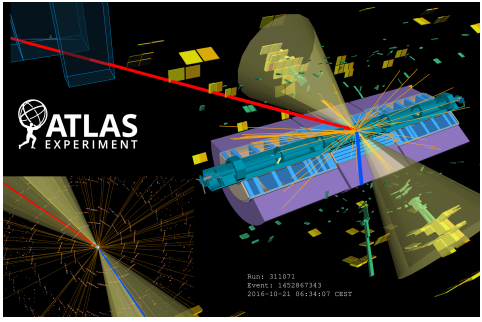
High Level Trigger(HLT) in Run 2

Athena Framework

- Athena: ATLAS offline framework for reconstruction and analysis
  - Developed in the beginning of the 2000, used over LHC Run 1 (2009-2013) and Run 2 (2015-2018)
  - Based on Gaudi framework[1]
  - Uses sequential processing
- High Level Trigger implemented in Athena via custom steering logic:
  - Schedules algorithm execution, manages decision logic
  - Offline algorithms require wrapper code for use with the trigger

Region of Interest (ROI)

- Geometrical region in ATLAS detector with interesting information
- During HLT processing partial event data in ROIs are used
- The HLT decision time budget is 0.5s, a full reconstruction can take up to 30s

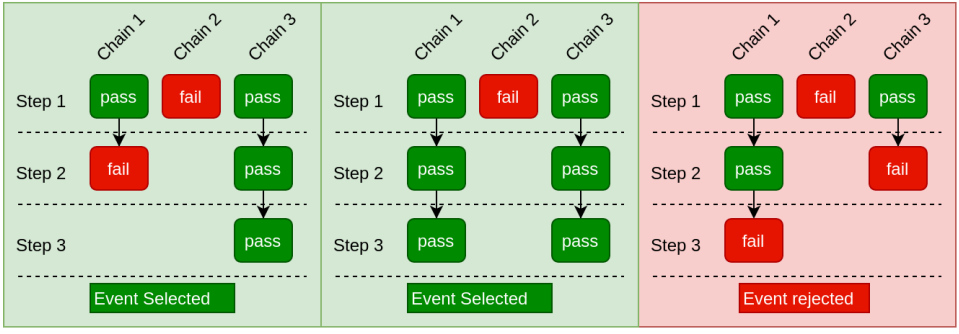


[1] G. Barrand et al. GAUDI - A software architecture and framework for building HEP data processing applications, Comput.Phys.Commun. 140 (2001) 45-55

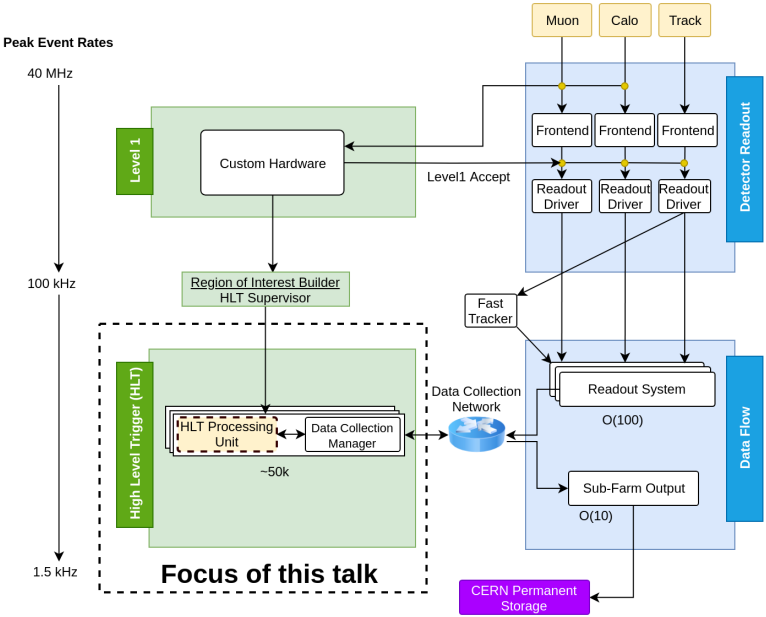
HIGH LEVEL TRIGGER(HLT) IN RUN 2

EVENT SELECTION

- Step: Sequence of feature extraction/reconstruction algorithms followed by a hypothesis testing algorithm.
  - Earlier steps do more coarse selection for **early rejection**
  - Each step is seeded by the previous step
  - Example step: reconstruct and identify muons within all Regions of Interest
- Chain: Sequence of steps
  - 1500 chains were active during Run 2
- Final HLT selection is positive, if event passes all steps in at least one chain



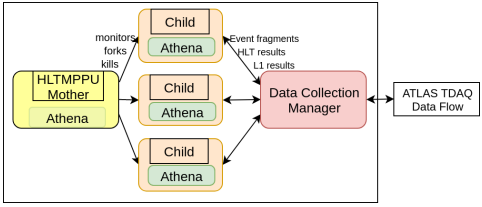
High Level Trigger(HLT) within ATLAS Trigger/DAQ

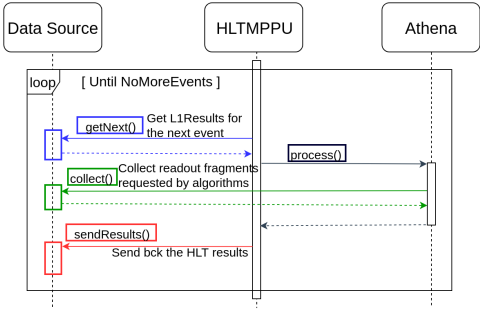


High Level Trigger(HLT) within ATLAS Trigger/DAQ

HLT Multi Process Processing Unit(HLTMPPU)

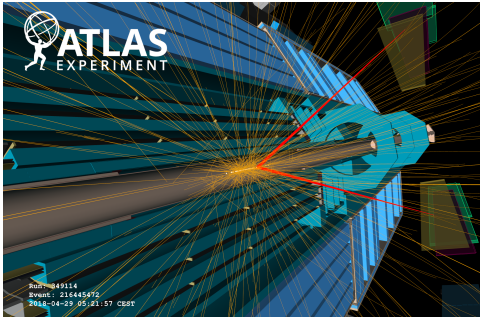
- TDAQ component that loads Athena HLT libraries which initialize algorithms and load conditions data
- Forks multiple children, and monitors them
- HLTMPPU manages event loop and processes events sequentially
- Uses Copy on-write(COW) mechanism to minimize memory overhead
- Communicates with Data Collection Manager to
  - Request L1 results for the next event
  - Request event fragments from different regions in detector
  - Send HLT decision and HLT results

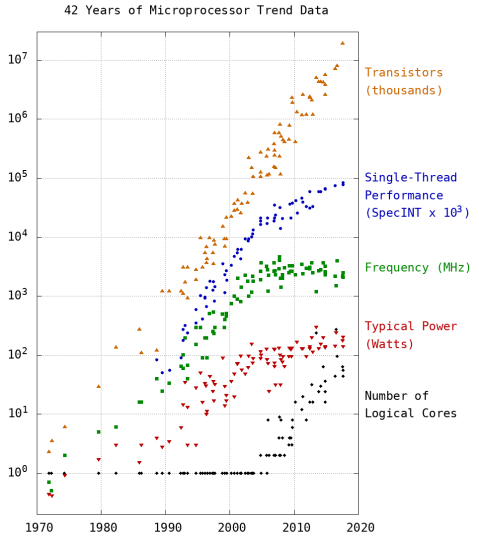




MOTIVATION FOR A NEW FRAMEWORK

- Hardware trends:
  - CPU frequencies are plateauing
  - Local memory available per core decreasing
  - Coprocessors are becomming more common
- Large Hadron Collider(LHC) expectations for Run 3 (14 TeV, maximum pileup of 70)
- Multithreaded event processing can use CPU resources better then the multi-process approach in Run 2

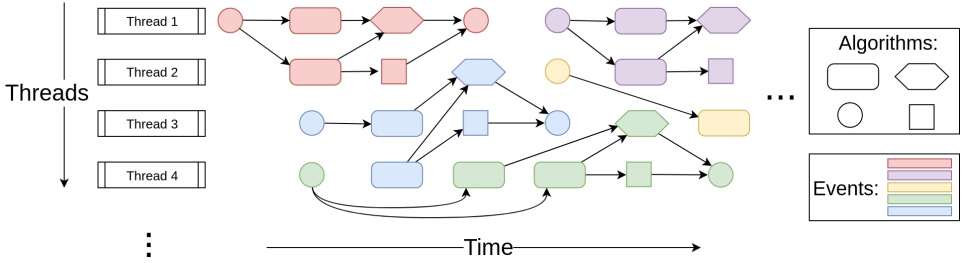




Data Source: https://github.com/karlrupp/microprocessor-trend-data

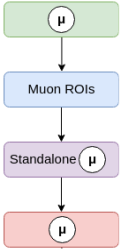
AthenaMT: Multi Threaded Athena Framework

- New offline framework from Run 3 (2021-2023) onward
- Built on Intel Thread Building Blocks (TBB), uses the Gaudi Hive Avalanche Scheduler
- Maximum code sharing with offline for use in High Level Trigger (HLT)
  - Direct use of offline algorithms without HLT specific wrappers
  - No HLT specific scheduler
  - EventView extension allows running algorithms once per Region of Interest
- Each AthenaMT instance is configured with number of event slots (number of concurrent events) and number of threads
- Types of parallelism
  - 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 vectorization



Types of Algorithms

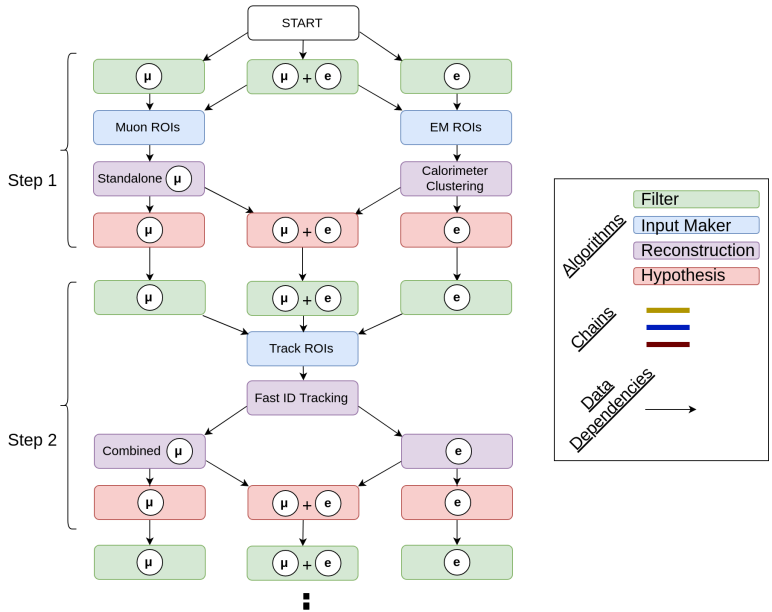
- Each step consist of 4 types of algorithms
- NEW Filter : Runs at the start of each step, aims to reject events as early as possible
- NEW Input Maker : Starting point for reconstruction, restricting it to ROIs
- Reconstruction : Reconstructs partial events from input-maker output
- Hypothesis : Performs hypothesis testing for all active chains, provides input to next steps filter



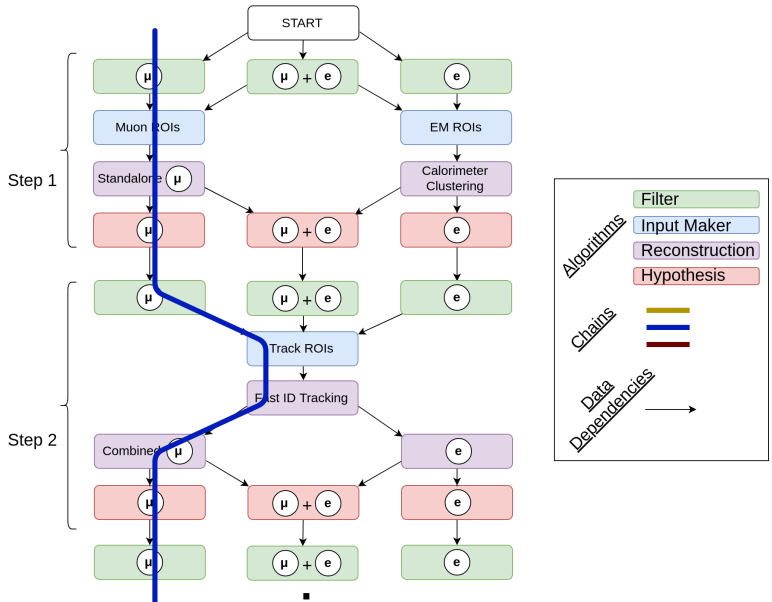
Execution of trigger chains

- At configuration-time, Gaudi Hive Scheduler builds a Data Dependency graph from Input and Output-Data Handles of algorithms
- Additionally, Control Flow graph defines sequences of algorithms and ensures early rejection
  - If a filter passes, it unlocks other types of algorithms that come after it
  - If all filters fail, the event processing terminates

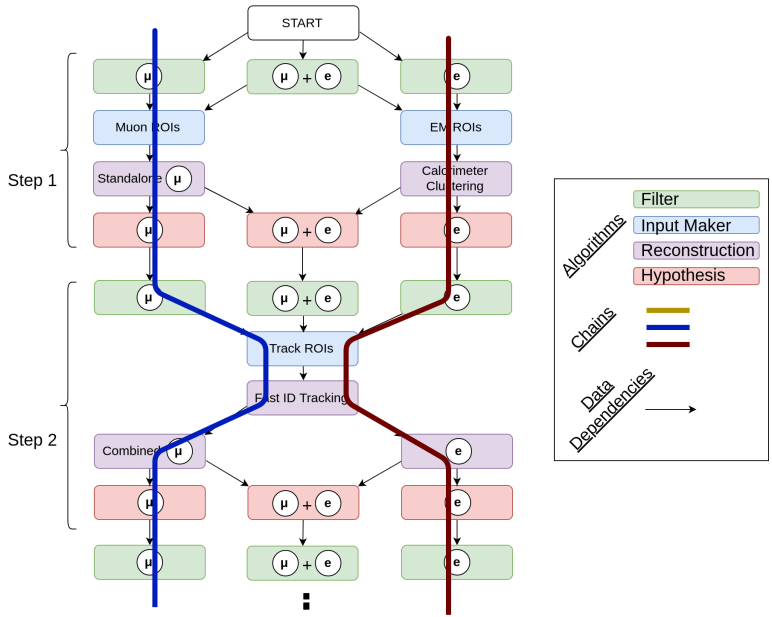
Data Dependencies



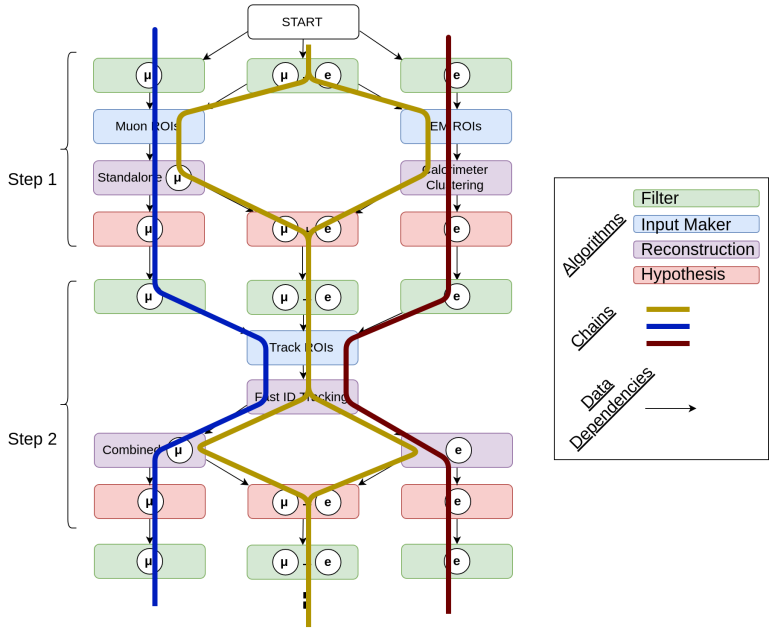
Data Dependencies



Data Dependencies

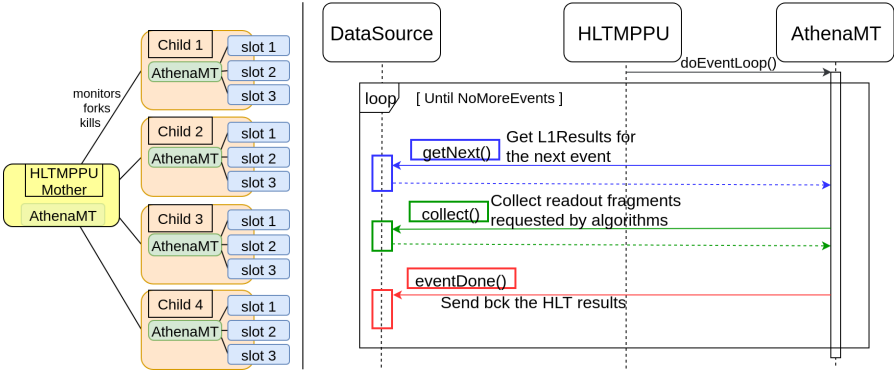


Data Dependencies



DATA FLOW IN RUN 3

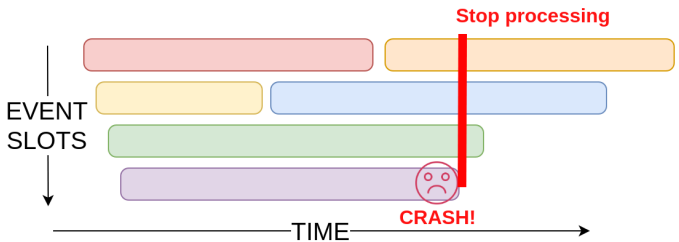
- TDAQ dataflow architecture remains the same
- HLT Processing Unit goes through changes to adapt to AthenaMT, event loop moves to AthenaMT
- Each Processing Unit is configured with number of forks, event slots and threads per fork



Error Handling/Recovery

Handling crashes or timeouts

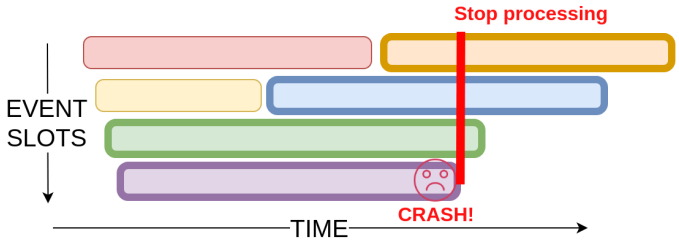
- **Debug stream**: Contains events for which the trigger was not able to make a decision
- If an algorithm crashes, or processing times out, all events being processed in the same fork are sent to debug stream
- Number of forks/event slots will be optimized to maximize performance and minimize event recovery overhead
- Having too many event slots per fork may increase the number of events in debug stream



Error Handling/Recovery

Handling crashes or timeouts

- **Debug stream**: Contains events for which the trigger was not able to make a decision
- If an algorithm crashes, or processing times out, all events being processed in the same fork are sent to debug stream
- Number of forks/event slots will be optimized to maximize performance and minimize event recovery overhead
- Having too many event slots per fork may increase the number of events in debug stream



Concluding Remarks

- The CPU frequencies are plateauing, local memory per core is decreasing and coprocessors are becoming more popular
- Multi threaded AthenaMT framework enables better use of computing resources with these hardware trends
- High level trigger can re-use the components and algorithms of AthenaMT
- High level trigger keeps the key concepts such as Region of Interests and early rejection
- Developments are ongoing on both TDAQ and offline software
- Validation campaigns and cosmic runs over next 2 years will be used to ensure readiness for Run 3

Backup Slides

EVENT VIEWS

DIFFERENCE BETWEEN ATHENAMT IN ONLINE AND OFFLINE

- Offline: Algorithms run once on full events
- Online: Algorithms run on partial events based on Regions of Interest(ROI), and they may run multiple times or never
- One needs a solution to run same algorithms without modification on partial events
- EventViews solve this problem by implementing EventStore interface
- They contain objects from specific RoI

IMPLEMENTATION

- A Creator algorithm creates the EventView stores within the primary store
- Algorithms run in their respective view, without modification
- Data from views can be merged back into the main event context

CONTROL FLOW GRAPHS

- Scheduler creates a Control Flow Graph from **OR** and **AND** nodes
- **OR** nodes: It's children is executed in parallel and result is OR of all children
- **AND** nodes: It's children are executed sequentially, and can exit early if a child returns FALSE