# Triggering ATLAS with multiple threads

Based on ATL-DAQ-SLIDE-2019-850

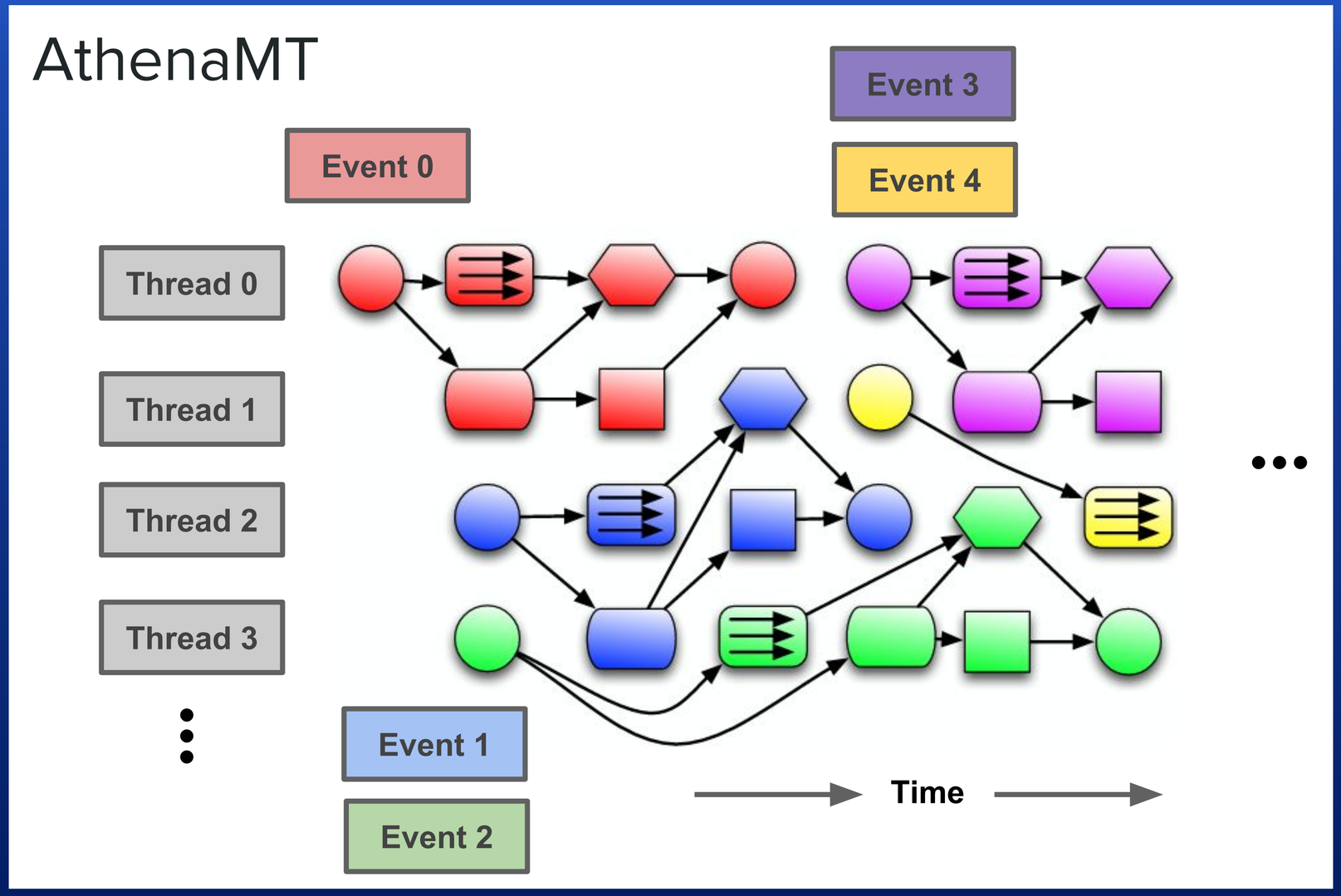
Teng Jian Khoo

Université de Genève

# Overview

- Computing resources don't scale with challenge of growing LHC luminosity — SW evolution necessary
- Large project to implement GaudiHive & AthenaMT framework upgrades and subsequently adapt algorithmic code
  - Framework elements common with LHCb and other experiments [Gaudi webpage]
  - Multithreading uses Intel Threading Building Blocks (TBB)
- Here give perspective based on adaptation of ATLAS jet trigger, missing ET reconstruction
  - Multithreading as a non-expert!

# PARALLELISM

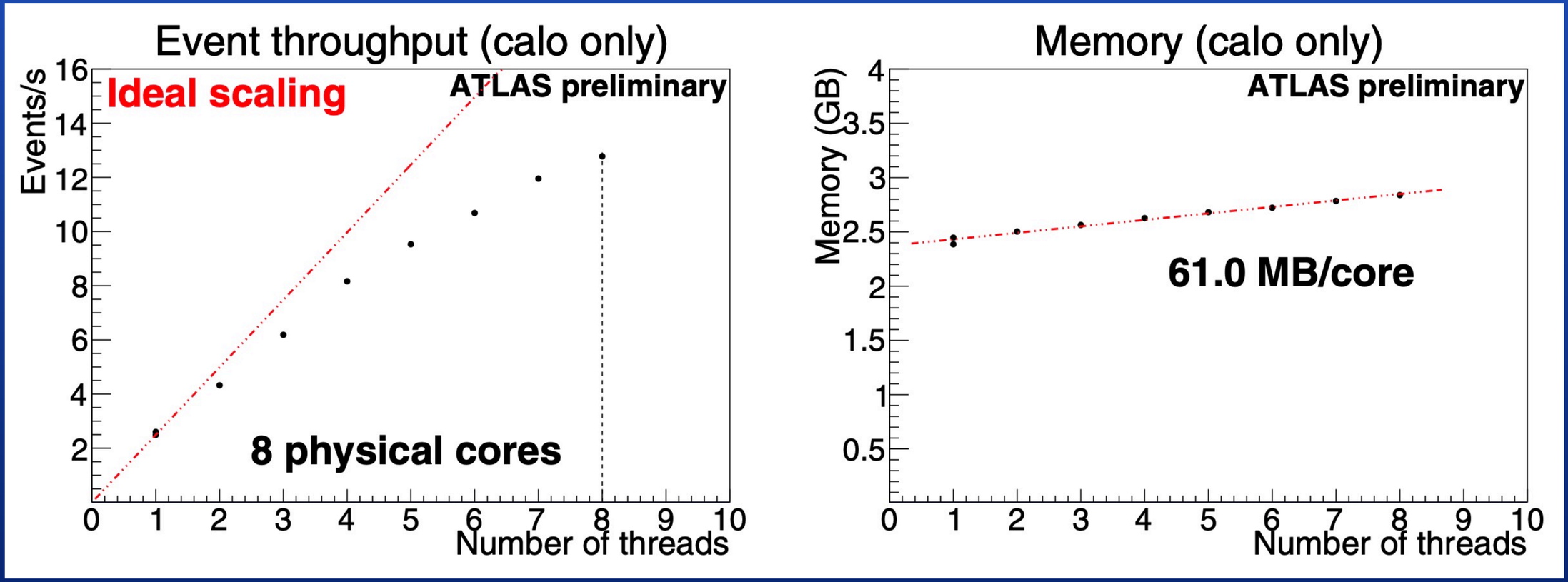



Inter-event ✓

Intra-event ✓

Sub-algorithm Possible

# PERFORMANCE SCALING




**Close to ideal CPU scaling per core with minimal memory growth**
**Some bottlenecks still to be addressed**

*S Snyder et al 2019, EPJ Web Conf., 214 (2019) 05018*

# CASE-STUDY: SW TRIGGER

- Multiple stages of reconstruction
  - Non-trivial data flow between elements
    - Tight latency/event rate constraints
      - Complex control flow for early termination
        - No room for execution failures during data-taking

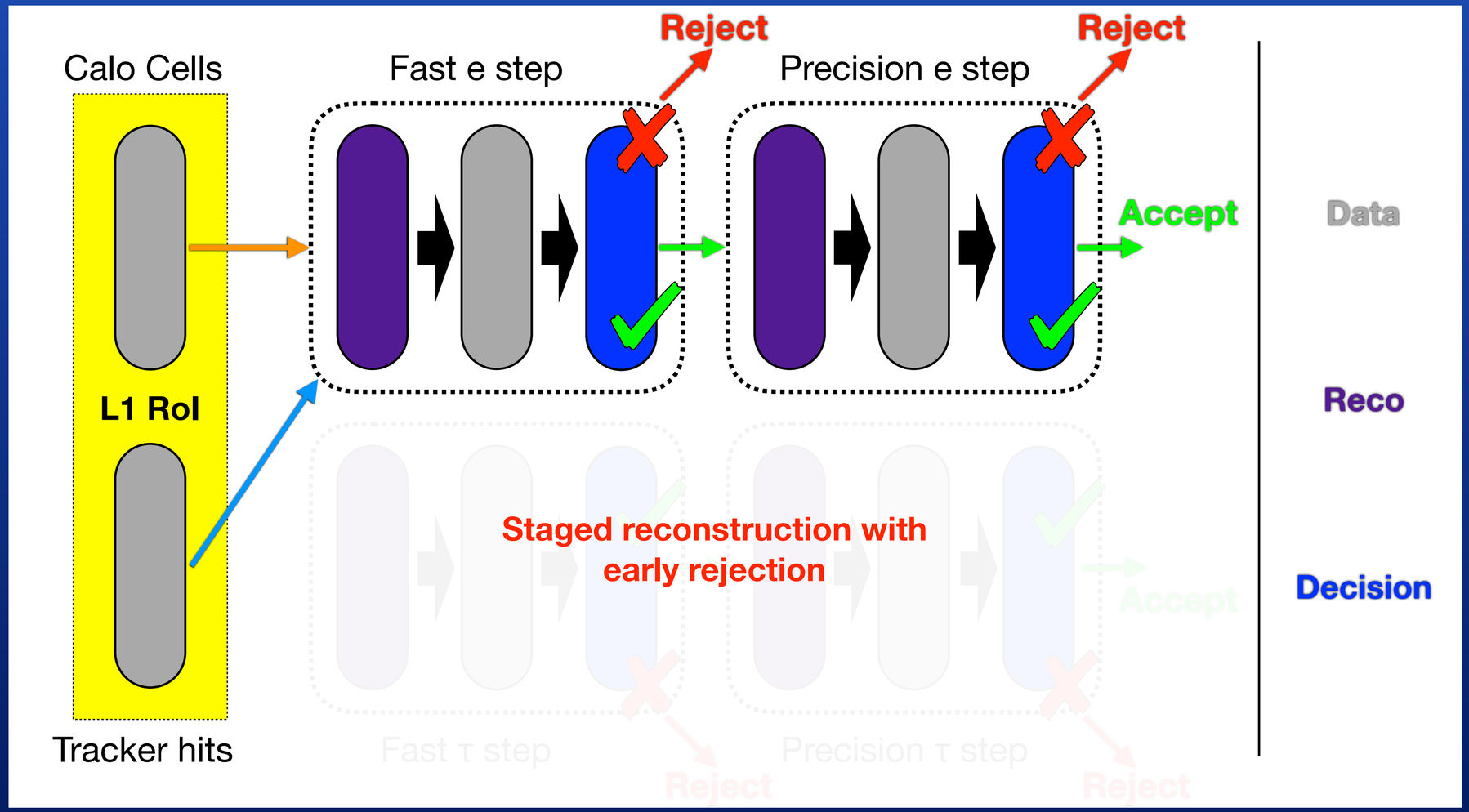
- Trigger is a stringent test for multithreaded workflows!

# Core HLT Elements

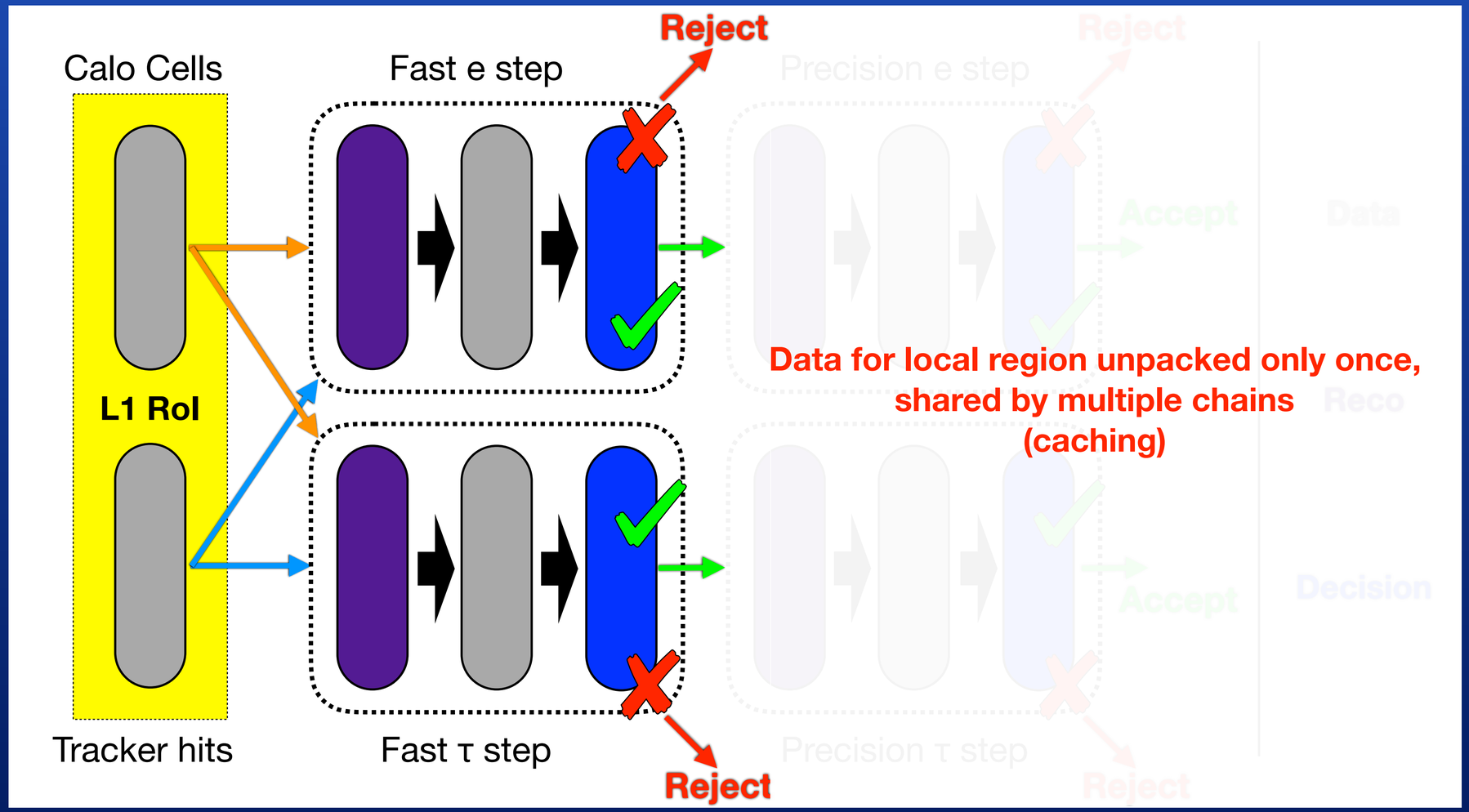
- Algorithmic code
  - Four-momentum reconstruction & Particle ID
  - Decision-making (“Hypothesis”)
- Regional reconstruction
  - Local detector readout where L1 trigger fired
    — “Regions of Interest” (all detector slices)
- Data flow & scheduling (“Control Flow”)
  - Reconstruct once, cache data for reuse
  - Early termination when event rejection is established
- Decisions recorded as event data

# TRIGGER DECISION SCHEMATIC

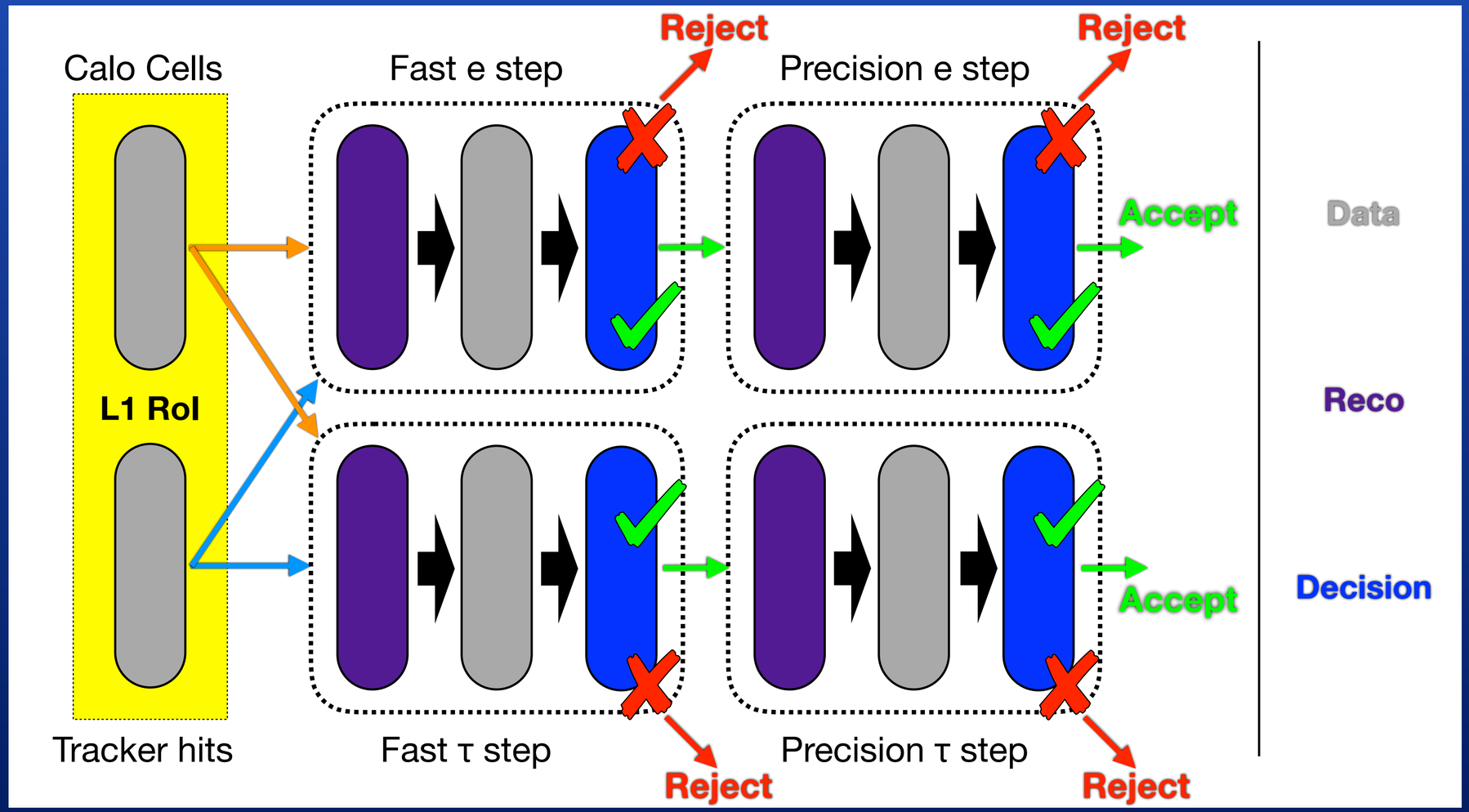
# TRIGGER DECISION SCHEMATIC



# TRIGGER DECISION SCHEMATIC



# TRIGGER DECISION SCHEMATIC



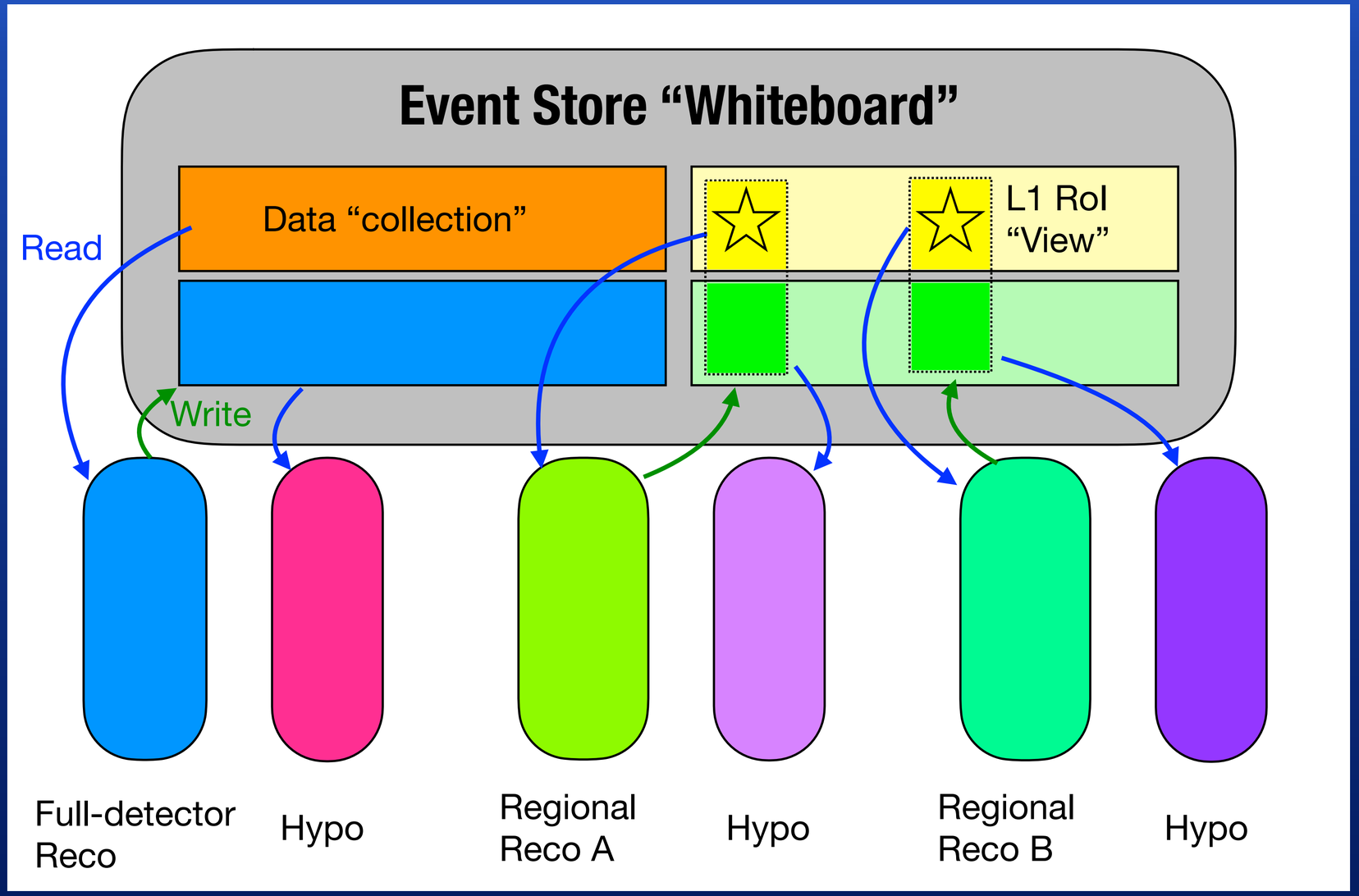
# MULTITHREADING COMMANDMENTS

- Thy code must be thread-friendly
  - Const data access
  - State-free execution
- Thou shalt have one sole scheduler
  - Integration of Control Flow elements into framework allowing execution to be stopped early
- Thou shalt have no callbacks
  - Conditions object containers in data store
  - Intervals Of Validity mapped to data objects
  - Conditions Algorithms populate store for new IOV

# CORE HLT ELEMENTS IN ATHENAMT

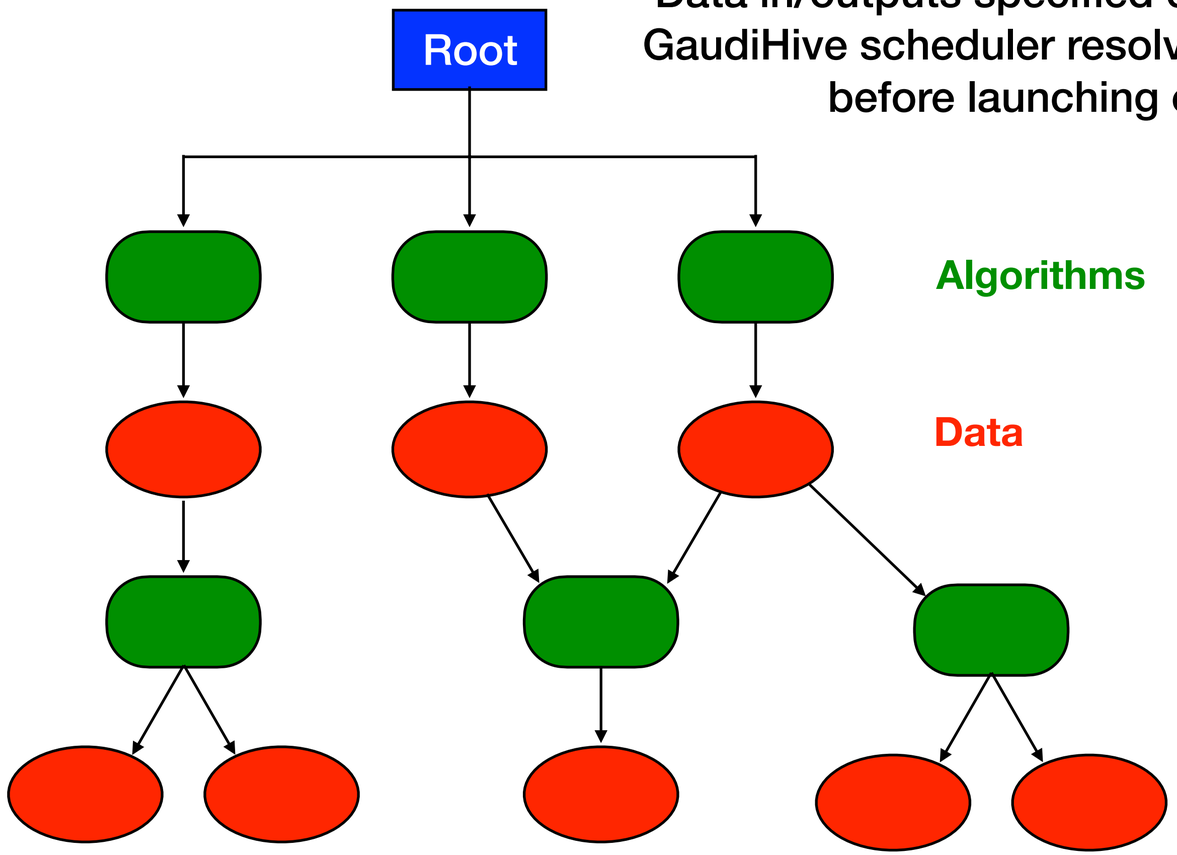
- Algorithmic code
  - Shared where possible with offline domain
- Regional reconstruction
  - Views in Event Store — restrict geometric acceptance transparently to algorithmic code
  - Parallel reconstruction of multiple Regions of Interest
- Data flow & scheduling
  - GaudiHive graph-based scheduler
    - Declarative data access
  - Control Flow sequences
- Decisions recorded as event data

# ATHENAMT DATA ACCESS



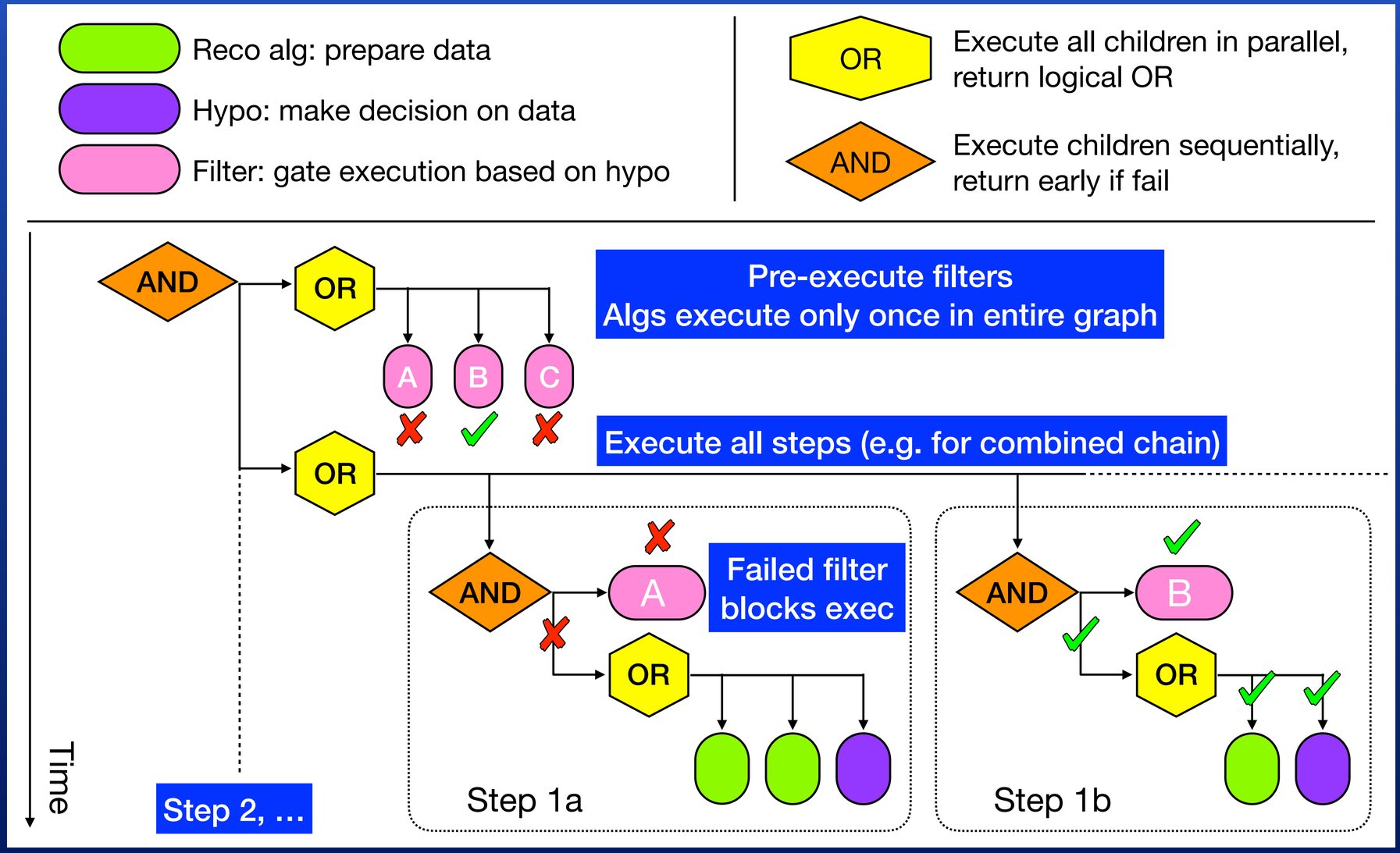
# ATHENAMT SCHEDULING

Data in/outputs specified during initialisation.
GaudiHive scheduler resolves execution graph
before launching event loop




Sub-algorithm components (tools)
propagate data inputs/outputs to parent

# TRIGGER SCHEDULING



# HIGHLIGHTED CHALLENGES

- Global objects, mutables, caching
  - Substantial rewrites needed in some areas to remove global maps etc
  - Mutable info in Event Data Model— create (thread-)local helper objects to manage
  - Caches in algorithmic code — create local objects and pass into method calls
  - Monitoring code: avoid direct user interaction with histograms
- Debugging rare problems
  - Thorough validation against single-threaded jobs needed
  - Issues can be subtle, some required partition of output files
  - Also use dedicated tools (Intel Inspector), static code checks, …
- Dedicated optimisation for certain data structures
- Non-thread-safe dependencies (fastjet-contrib)

# SUMMARY

- Common Gaudi framework extended to support multithreading
- AthenaMT extensions permit ATLAS HLT operation for O(500ms) reconstruction and event filtering
  - Regional reconstruction with Event Views
  - Early rejection with Control Flow gate nodes
- Some Run 2 chains fully implemented
- Validation & performance assessments to come

# References

- AthenaMT: ATL-SOFT-PROC-2017-019
- GaudiHive/Avalanche scheduler: http://concurrency.web.cern.ch/GaudiHive
- AthenaMT in ATLAS Trigger: ATL-DAQ-SLIDE-2019-850
- Multithreaded simulation: ATL-SOFT-SLIDE-2019-796
- Concurrent data structures: ATL-SOFT-SLIDE-2019-780
- Monitoring multithreaded reconstruction: CHEP 2018

# BACKUPS

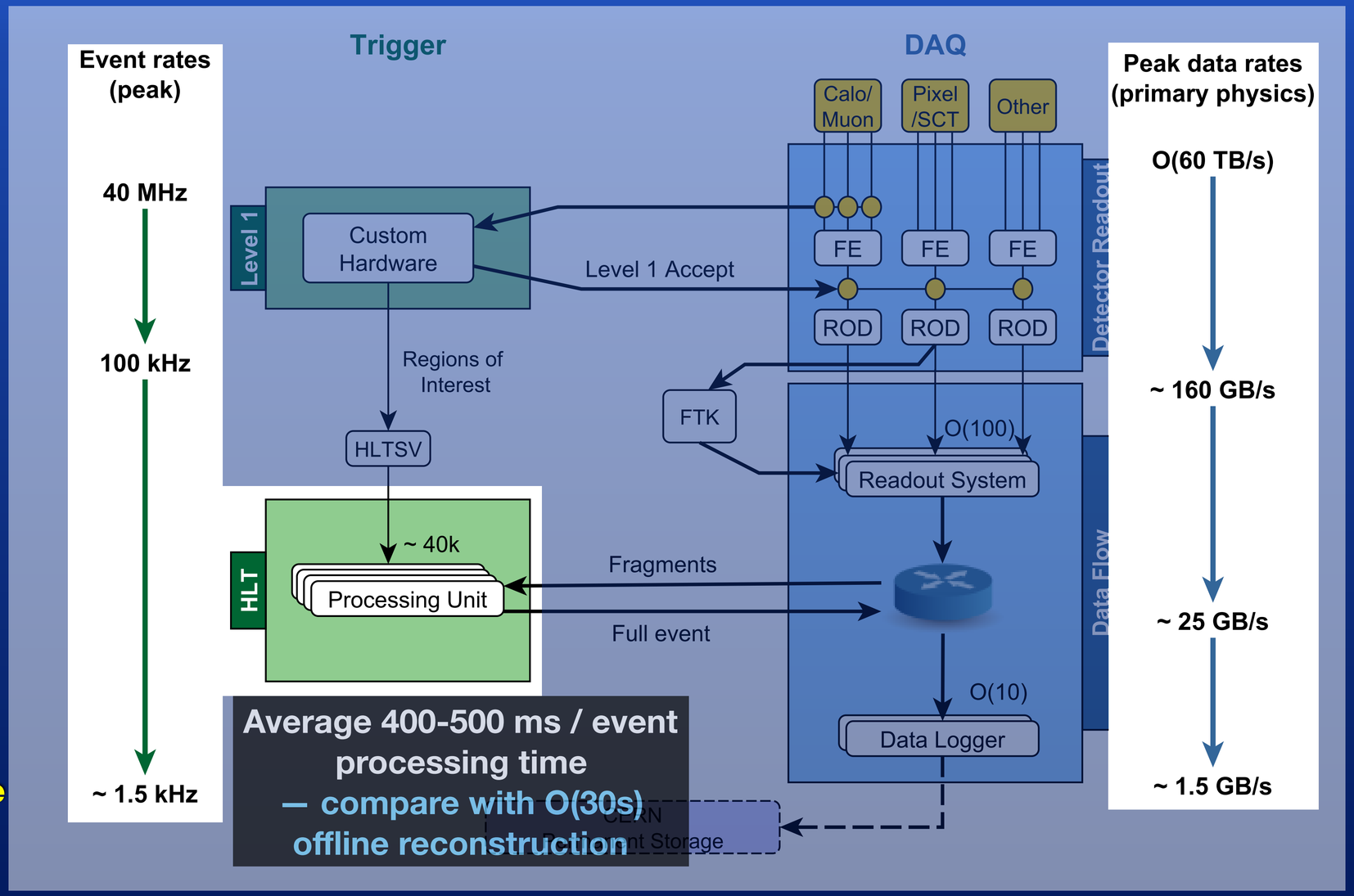
# Why Multithreading?

- "Embarrassingly parallel" no longer enough
  - Processor speed plateau —> growth of multicore
  - Memory price floor —> need for memory sharing
- Run 2 approach: multiprocess execution
  - Fork workers after initialisation (or first event)
  - Share large static data structures
  - Slow memory growth unavoidable
  - Intra-event parallelisation limited
- Pathway to future computing architectures
  - E.g. GPUs or FPGAs for hardware acceleration

# HLT COMPUTING CONSTRAINTS

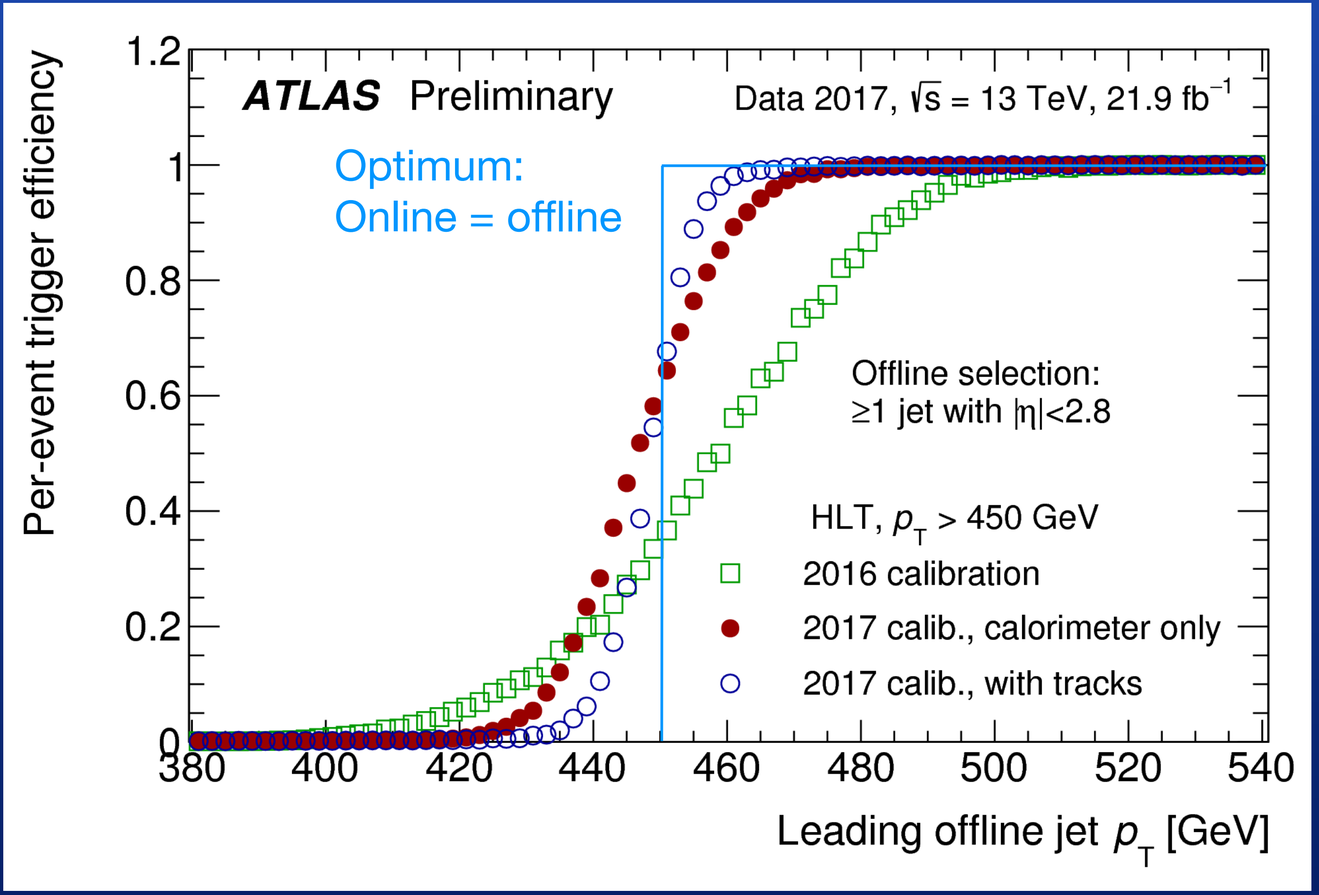
L1 input rate

Maximum rate to storage




Event size ~1MB

# PHYSICS PERFORMANCE




Improvement from implementing offline corrections at cost of CPU

*ATLAS-DAQ-PUB-2018-002*

# OVERVIEW OF ATHENA ARCHITECTURE