# ATLAS Trigger use-case

T. Bold - on behalf of ATLAS HLT s/w upgrade team

# ATLAS HLT working point

- The filtering at HLT is performed with quite limited resources

- Farm size and readout capacity .vs. rate allows only for partial event processing (~5%)

# Savings

- **RoIs** the data read & reconstructed for fragments of the detector
  —> shapes set initially by the L1 RoIs
  —> updated by the HLT as we go (eg.: track extrapolation)
  —> reshaped/constrained for last stage precise reconstruction (eg.: tau tracking)

- **Early rejection**
  —> the selection divided in parts of (reco-cut-reco-cut-reco-cut-…)
  —> failing cuts == break the chain
  —> that is done at the granularity of RoIs

# The ATLAS HLT now

- Custom control algorithm (steering) + thousands of sub algorithms (HLT FEX & HYPO)

- Reconstruction is effectively performed on demand for RoIs & only if we are selecting for a signature (i.e. no processing for prescaled out signatures)

- Lightweight monitoring to histograms (available online with about ~*O*(min) delay)

- Full control over the error handling
  —> monitored for operators crew
  —> if serious issue: direct to a special stream for debugging/recovery
  —> an issue in data or in code not enough to end the job (keep calm & carry on)

- Handles about 2k trigger chains (lines)

- HLT is reconfigurable during the run only by tweaking prescales (need to envisage scenarios)

# Why upgrade

- Memory on multi-core machines
  - Possible speedup due to better cache utilisation
- Modernisation of the offline code (precision sel. stages)
  - And HLT will use even more offline code
  - Common algorithm interface
- The existing fwk. inherently non-MT compatible
  - https://cds.cern.ch/record/2268736

New HLT design is the result of requirements capture, design, review over the past 1-2 years

# Ingredient no. 1

# Processing in RoIs through the Views

- In current HLT algorithm obtain „the RoI context” (Trigger Element)  when they are invoked.
  - Used to obtain input data, and store output
  - Contexts linked in d-graph (tree like structure) saving record of execution order, intermediate decisions, objects etc.
  - Typical algorithm executed over available contexts one after another (blind to other contexts)
- The replacement in the upgraded system are EventViews

# Processing in RoIs through the Views

- The EventView (SG::View) has identical API as the ATLAS StoreGate (whiteboard)

- An algorithm can be „cheated” by redirecting DataHandles to an instance of the View rather than to the main store

- Who can do that? The scheduler.

  - This is because the scheduler passes anyways the context to algorithms

# Implementation

- We have demo implementation in which:
  —> detector data is inspected by and EventViewCreator —> creates viewa for each RoI
  —> launch (by explicitly call) sub-algs.

- Plan is to be able to create views and let the scheduler do the rest

  - parallelism

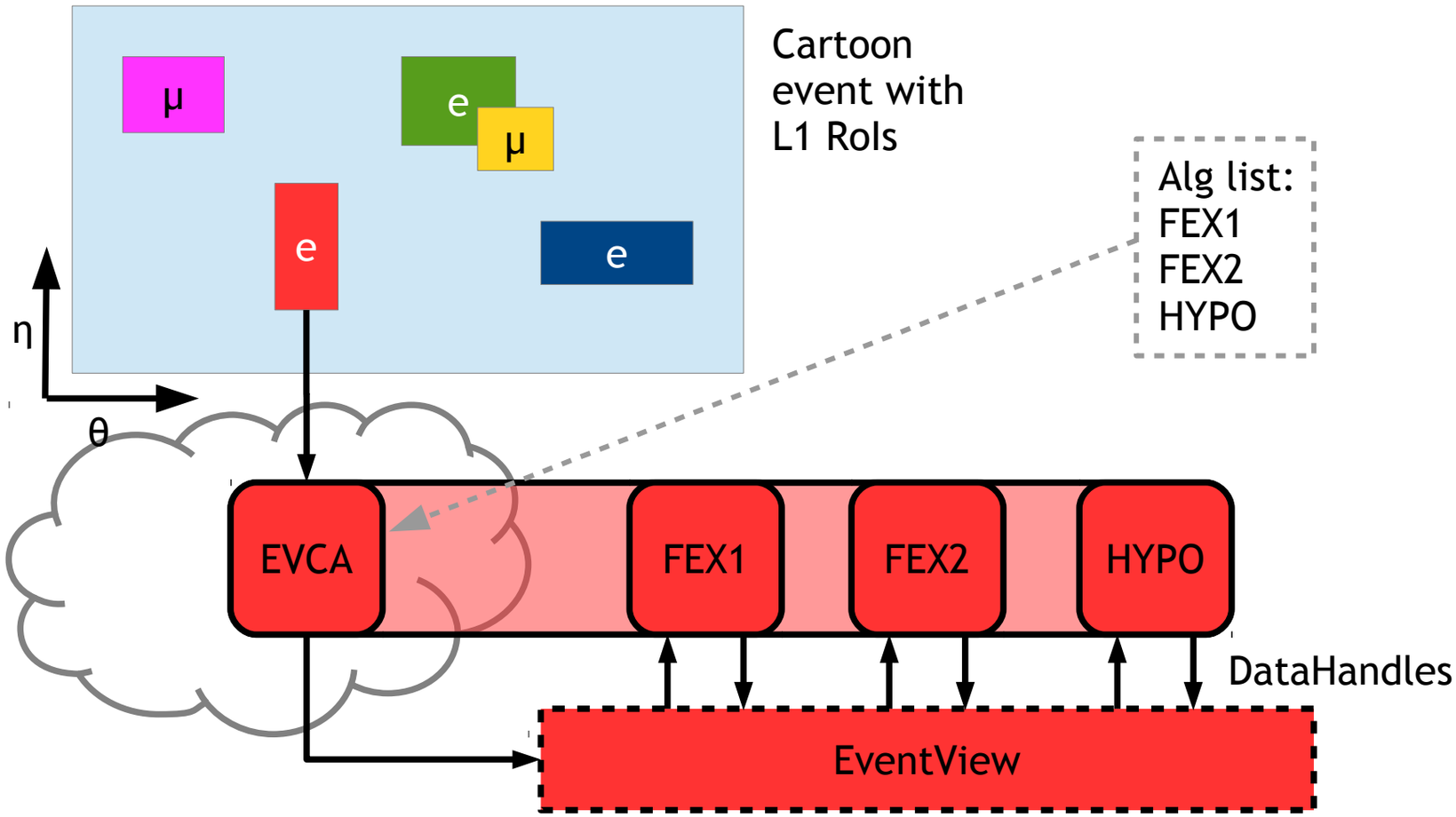
  - data dependencies resolution

# Slide from Ben Wynne

## HLT development

We have working HLT examples using EventViews in the package Trigger/TrigValidation/TrigUpgradeTest

However, these don't use the scheduler to run algorithms in views – the view creator algorithm just executes a list of algorithms itself

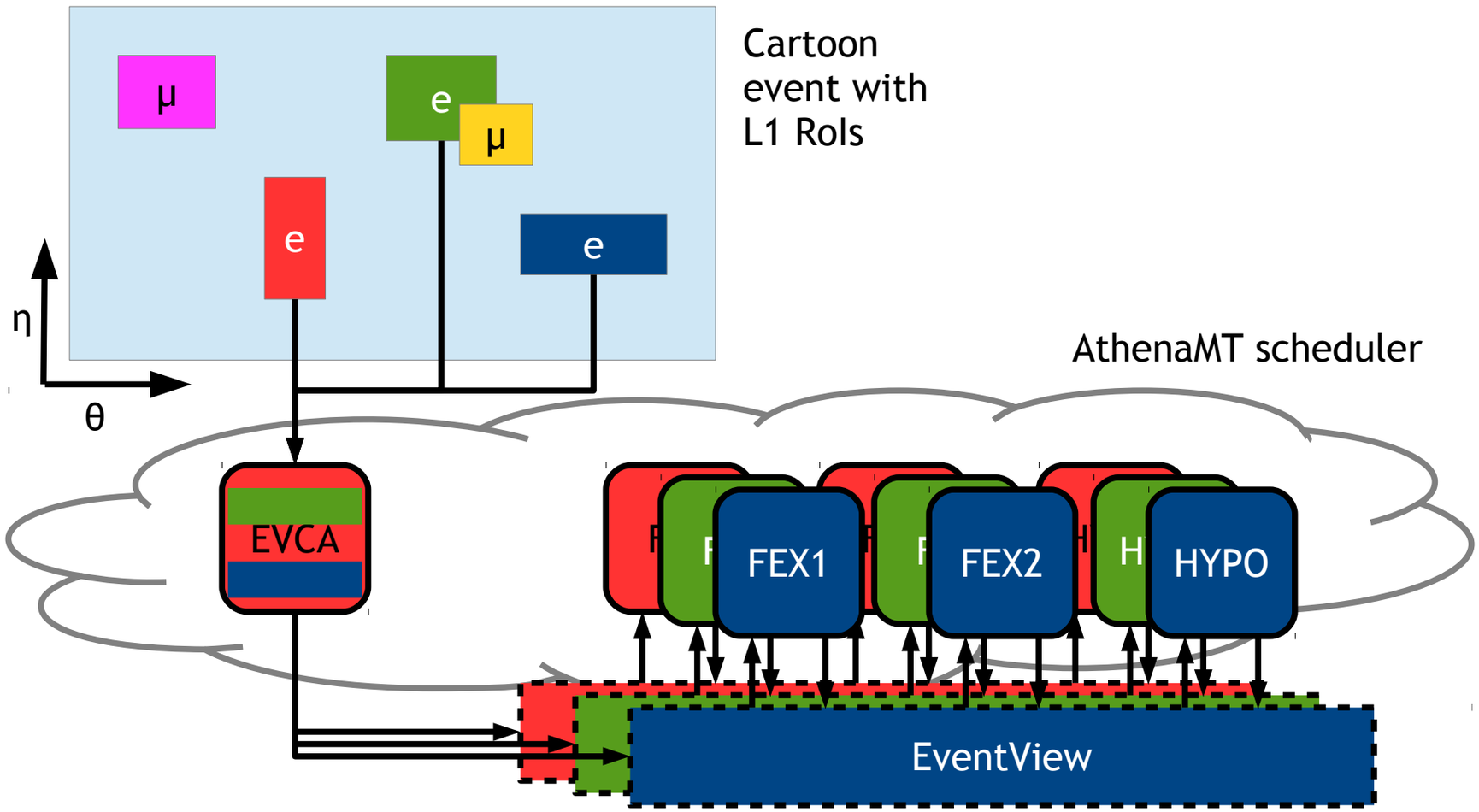


Slide from Ben Wynne

# Multiple EventViews

Clearly we expect some events to contain multiple RoIs

Consequently we might create multiple EventViews, each with the same sequence of algorithms processing it




Cartoon event with L1 RoIs

4

# An efficient implementation

- Processing of the whole event is determined by the entire (event-wide) DF/CF while processing in views shall use a small portion of the whole DF configuration.

- The implementation should not need to traverse whole event CF/DF once an algorithm in a view completes

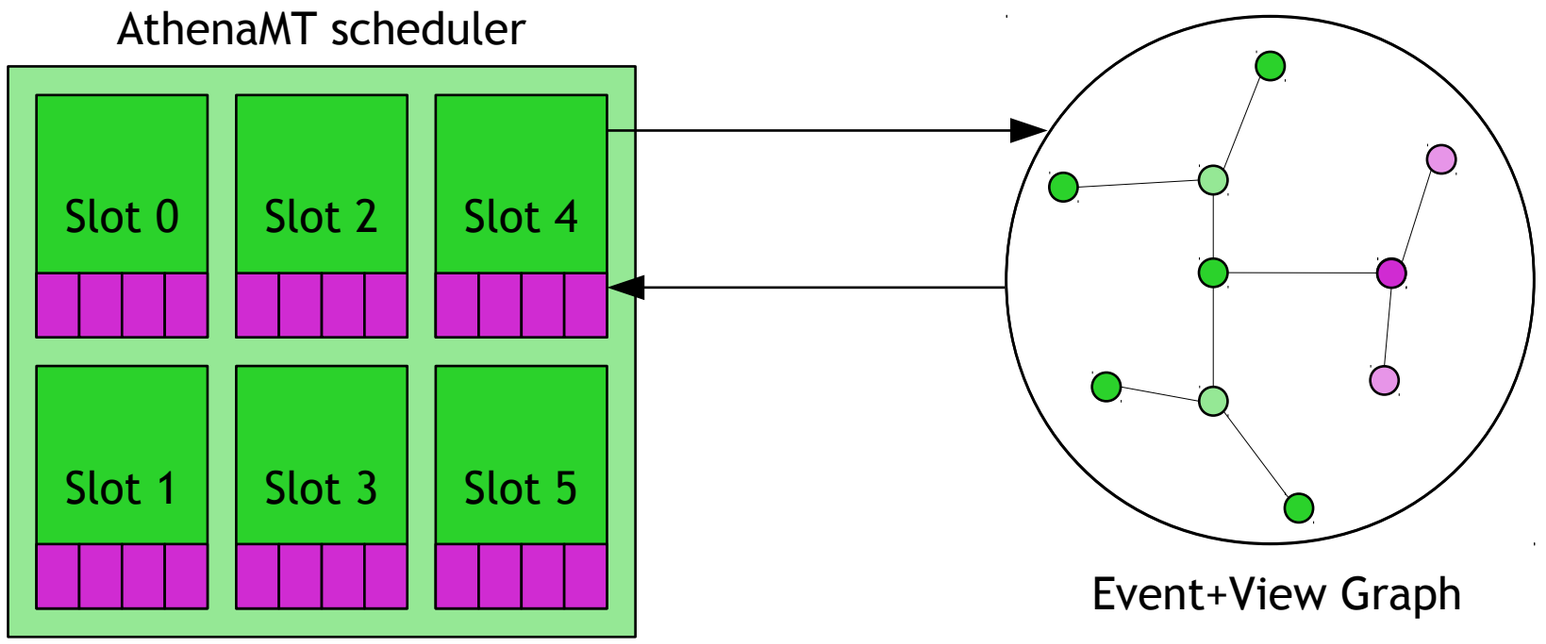
# Slide from Ben Wynne

## Nesting slots

Ideally the Precedence service updates the state of slots, and the scheduler simply stores and consumes that state

So, if we store the view slots as members of the corresponding event slot, they can be passed to Precedence service for a co-ordinated update

Requires an extension of the graph traversal in how the EventView subgraph regions are walked through (including multiple views per event) and entered/escaped by graph visitors



# Ingredient no. 2

# The HLT early rejection with ControlFlow

- What we are after is **conditional reconstruction**
- CF can be easily used for that however
  - Missing producer algorithm —> stall at the later stage if all subsequent sequencers are not disabled

Think of it: tracking for e&mu signatures, needs to know regions to run on. None of electrons pass the earlier selection.

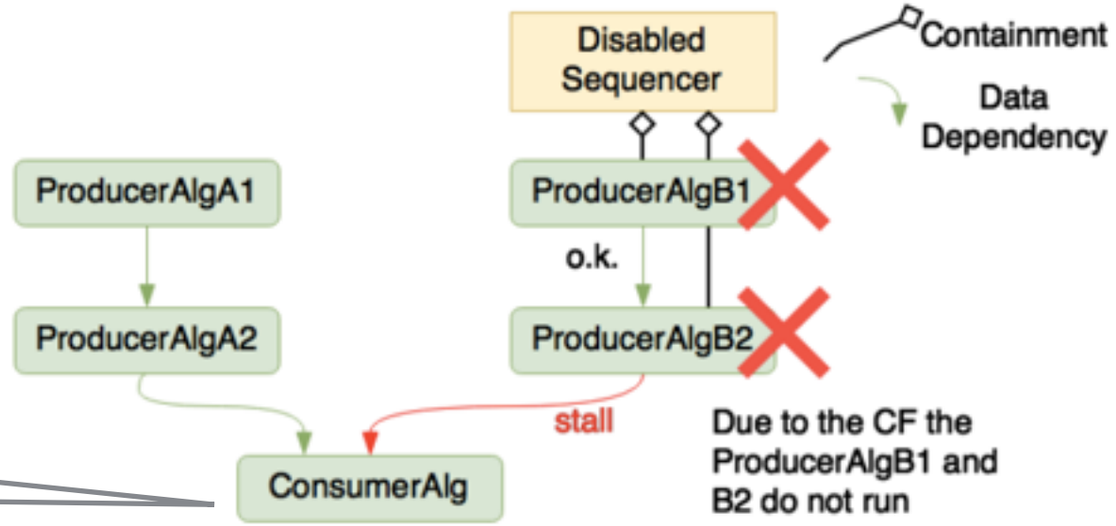


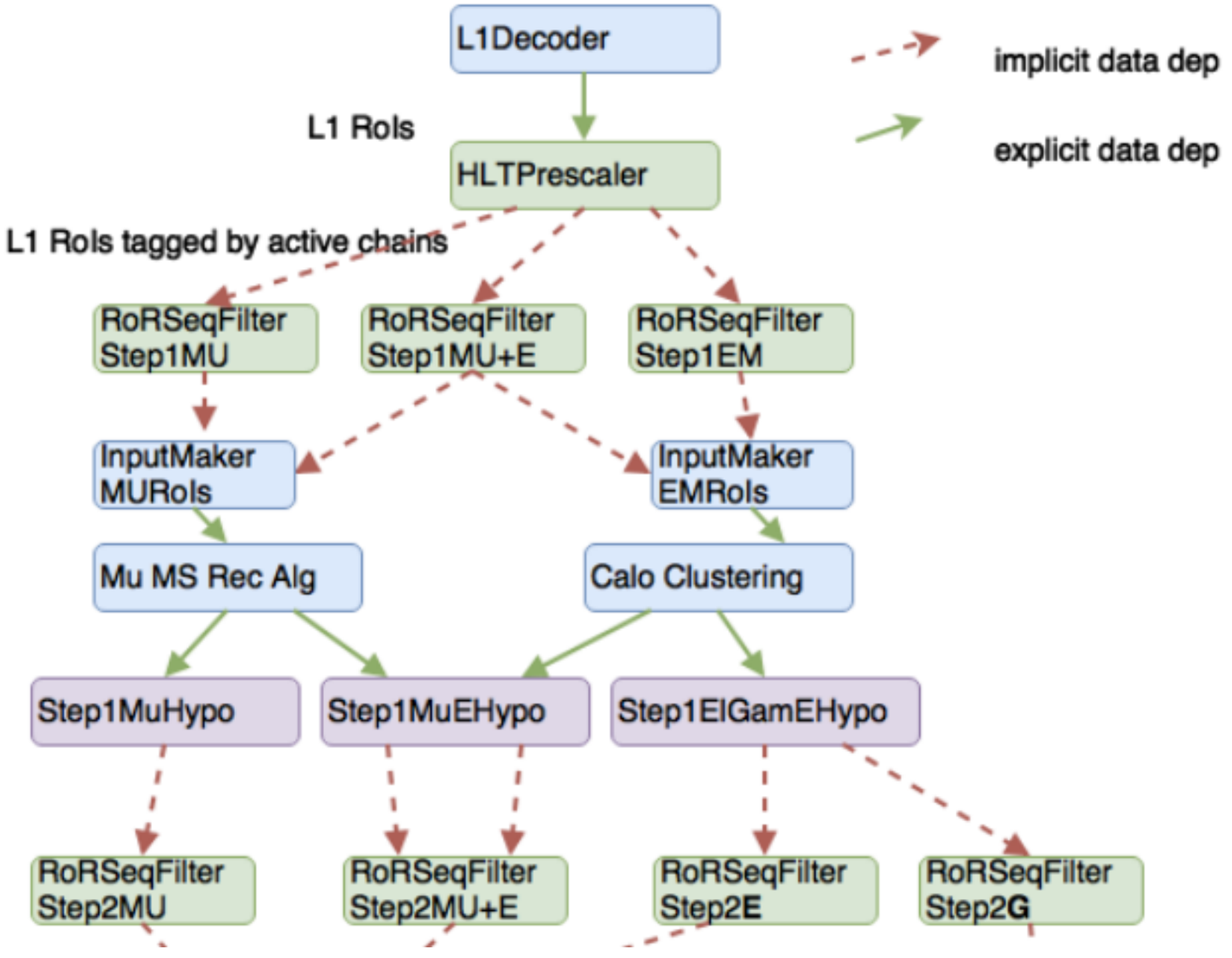

Figure 7.3: An illustration of a negative interaction between Control Flow and Data Flow that leads to a stall in the execution of the ConsumerAlg.

# The HLT early rejection with ControlFlow

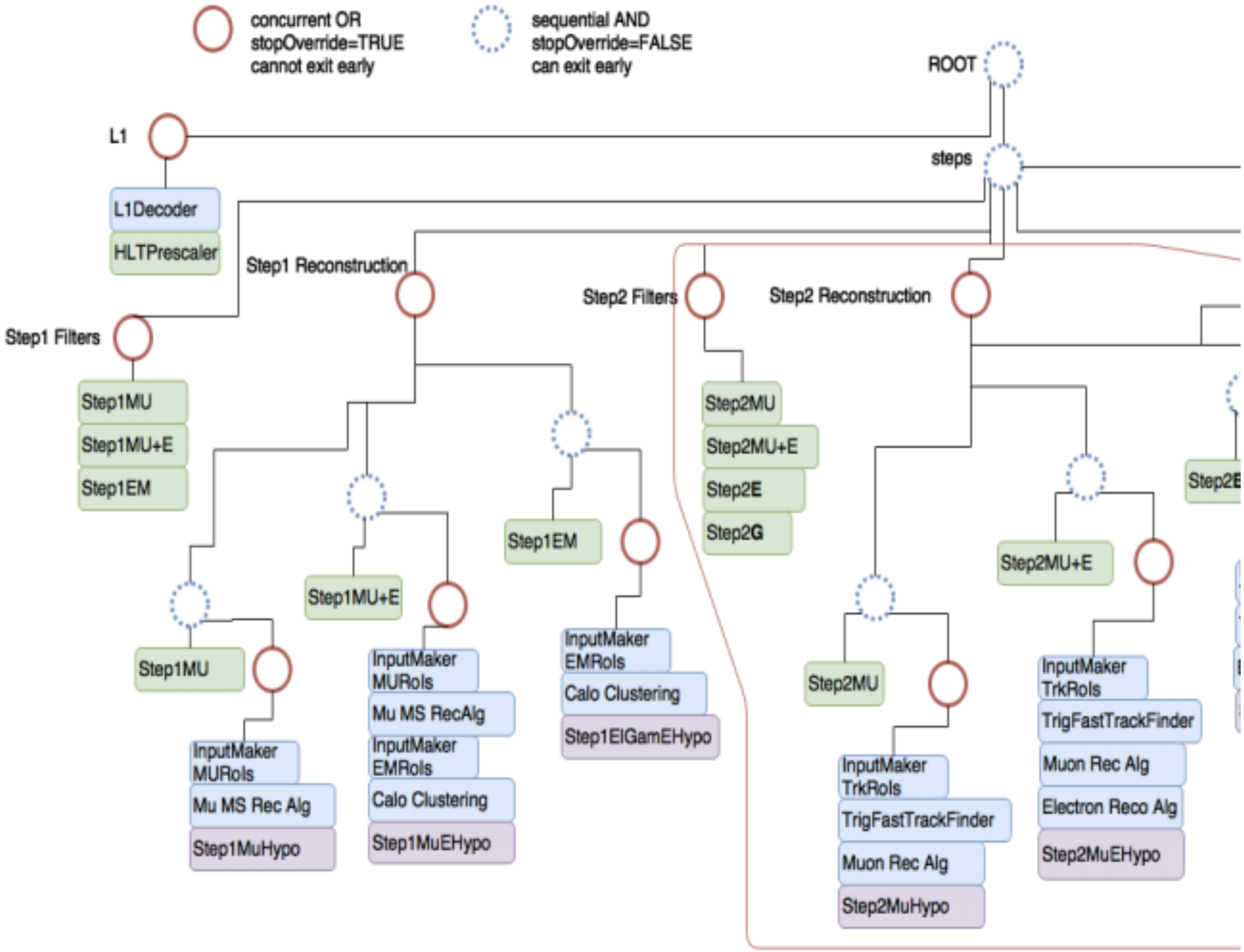
- The „hidden” dependencies help solve the issue —> in practice renounced DH checked for validity explicitly by the consumer algorithm (implicit DH)

- However, we need to assure that all potential producers have completed or will never run in this event

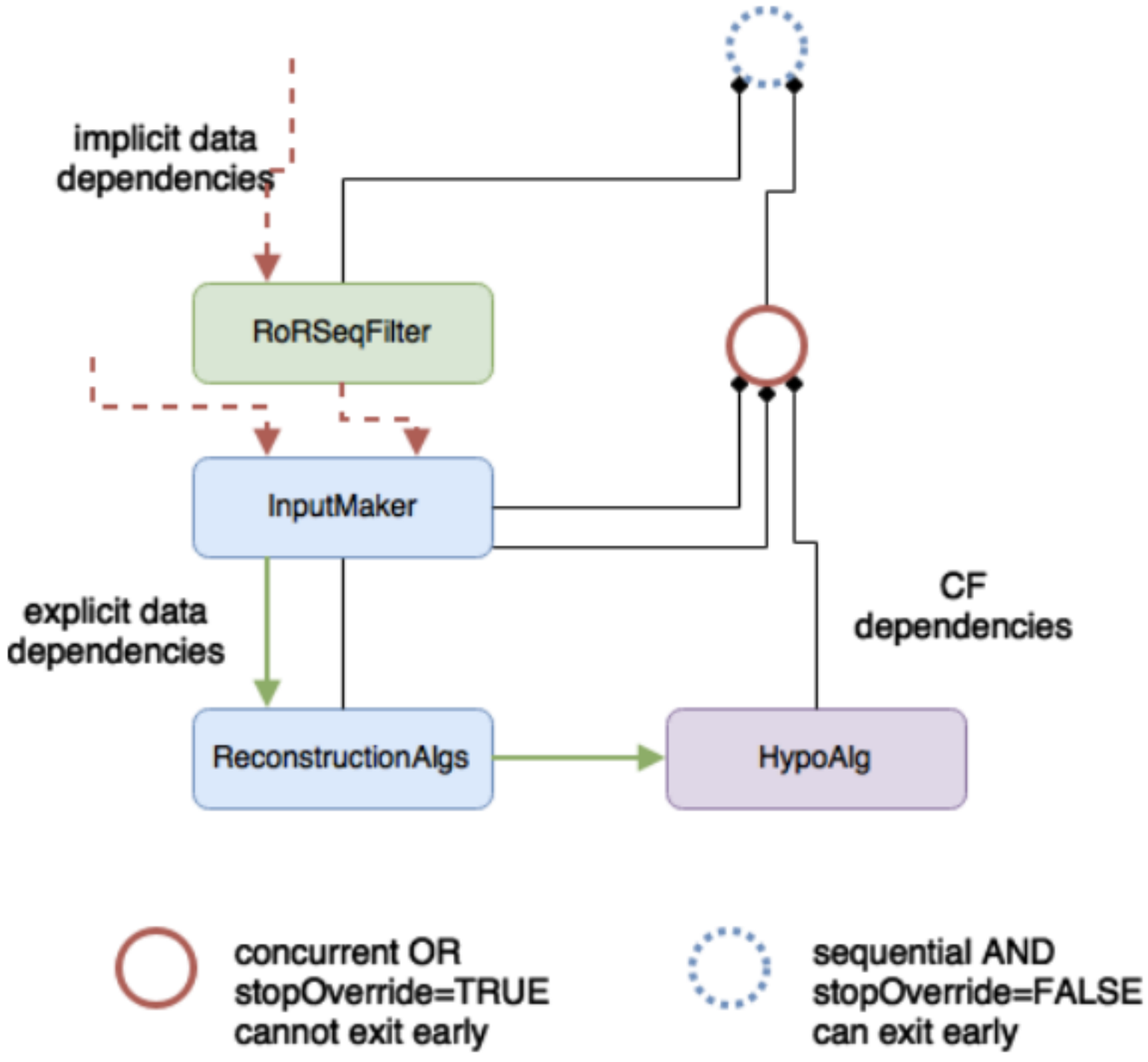
# The DF


L1Decoder
implicit data dep
L1 RoIs
explicit data dep
HLTPrescaler
L1 RoIs tagged by active chains
RoRSeqFilter Step1MU
RoRSeqFilter Step1MU+E
RoRSeqFilter Step1EM
InputMaker MURoIs
InputMaker EMRoIs
Mu MS Rec Alg
Calo Clustering
Step1MuHypo
Step1MuEHypo
Step1ElGamEHypo
RoRSeqFilter Step2MU
RoRSeqFilter Step2MU+E
RoRSeqFilter Step2E
RoRSeqFilter Step2G

# The CF


concurrent OR
stopOverride=TRUE
cannot exit early
sequential AND
stopOverride=FALSE
can exit early
ROOT
L1
steps
L1Decoder
HLTPrescaler
Step1 Reconstruction
Step2 Filters
Step2 Reconstruction
Step1 Filters
Step1MU
Step1MU+E
Step1EM
Step2MU
Step2MU+E
Step2E
Step2G
Step1MU+E
Step1EM
Step2MU+E
Step1MU
InputMaker MURois
Mu MS RecAlg
InputMaker EMRois
Calo Clustering
Step1ElGamEHypo
Step2MU
InputMaker TrkRois
TrigFastTrackFinder
Muon Rec Alg
Electron Reco Alg
Step2MuEHypo
InputMaker MURois
Mu MS Rec Alg
Step1MuHypo
InputMaker MURois
Mu MS RecAlg
InputMaker EMRois
Calo Clustering
Step1MuEHypo
InputMaker TrkRois
TrigFastTrackFinder
Muon Rec Alg
Step2MuHypo

# Elementary HLT control unit



# Milestones

| | Milestone | Date |
|---|---|---|
| M1 | **Simple electron chain:** (1 thread)<br>• demonstrate multiple thresholds<br>• Rejection using Control Flow,<br>• Creation of EventViews; First implementation of execution in views. | Q2 17 |
| M2 | **First Implementation:** (1 thread)<br>• Complete electron chain. Add gamma, Jet & Muon chains<br>• HLT workflow fully integrated into scheduler. | Q4 17 |

2

- Comments:
  - The last point depends on Gaudi
  - HLT would need to process multiple-events-in-flight for the Run 3
    - In case we do not manage to cross —threads=1 boundary we need to be able to run with PT mode at the start of the Run 3
      —> MT framework has to allow to fork after init.

# Status

- The ATLAS HLT s/w upgrade is underway
- Lots of work on HLT as we adapt towards the offline model
- There new HLT will heavily rely on the Gaudi scheduler infrastructure for:
  - processing in Views (demonstrator in place), parts of implementation in private branch, full implementation still missing (collab. of Ben & Illya)
  - reject early via the CF (tested, it seem to work just as expected)
- The support of this features is really crucial for the next generation HLT s/w