SOFTWARE BASED CONTROL AND MONITORING OF A HARDWARE BASED TRACK RECONSTRUCTION SYSTEM FOR THE ATLAS EXPERIMENT

Simone Sottocornola on behalf of ATLAS collaboration

CHEP 2018

OUTLINE

- Triggering at LHC and tracking challenge
- FTK overview
- FTK online software status
- FTK monitoring status
- FTK commissioning status
- Conclusions

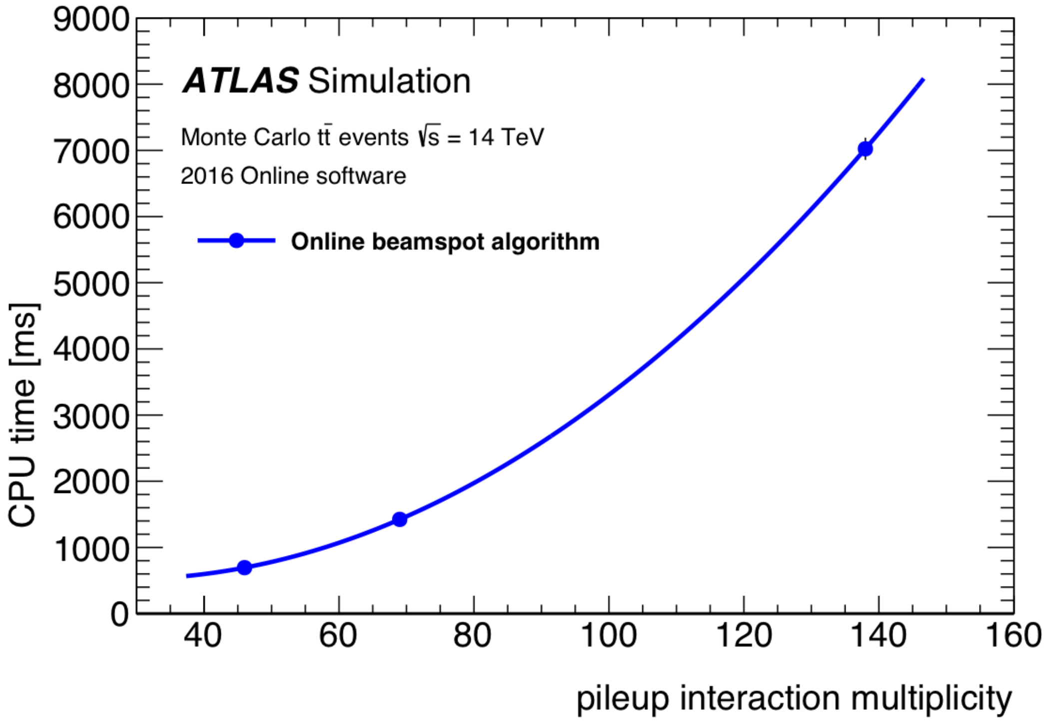
ATLAS TRIGGER @ LHC

- Rate reduction required ~ 40k
  - Through multilevel architecture
- Level 1 (L1):
  - Hardware based
  - Low resolution data
    - no Inner Detector (ID) data
  - Rate reduction ~ 400 in $2.5\mu s$
- High Level Trigger (HLT):
  - Software based
  - Full resolution data from all the detectors
  - Rate reduction ~ 100 in $200ms$ avg

THE TRACKING CHALLENGE

- Track-based trigger selection fundamental
  - Less pileup dependent
  - Mandatory for jet reconstruction, MET, b and τ tagging …
- Tracking is a combinatorial problem
  - CPU processing time does not scale (linearly) with pileup
  - Tracking information used in small Region of Interest (RoI) for subset of events
  - Reconstruction times:
    - O(10)ms for a single RoI
    - O(10)s for full scan
- New hardware track trigger:
  - Designed to provide good resolution global track reconstruction at full L1 rate
    - Reconstructs all tracks (pT >1 GeV) for all L1 accepted events
    - Tracks provided at HLT in ~ 100 μs at the L1 rate (100 kHz)

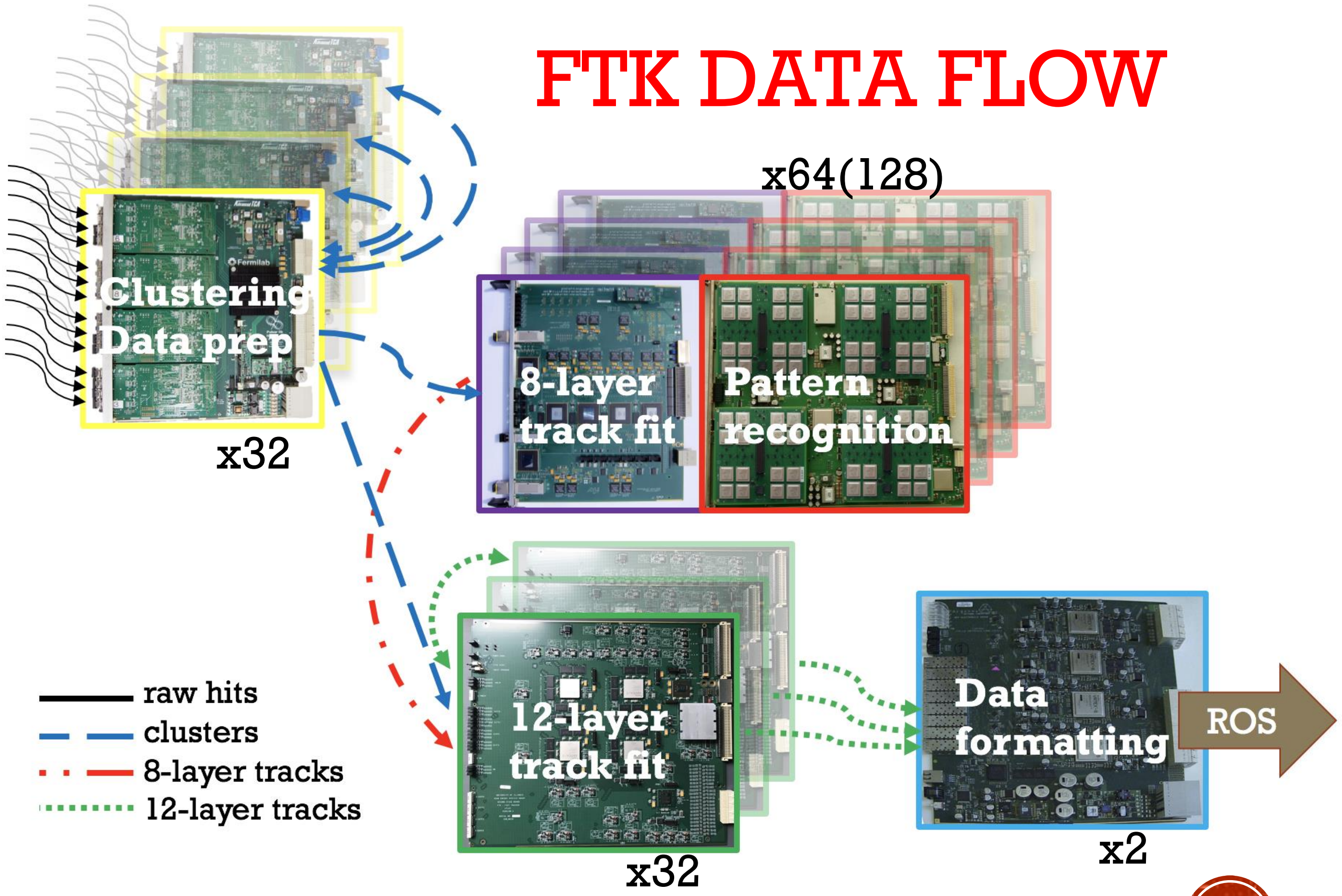
For more details see Julie's talk



THE FAST TRACKER

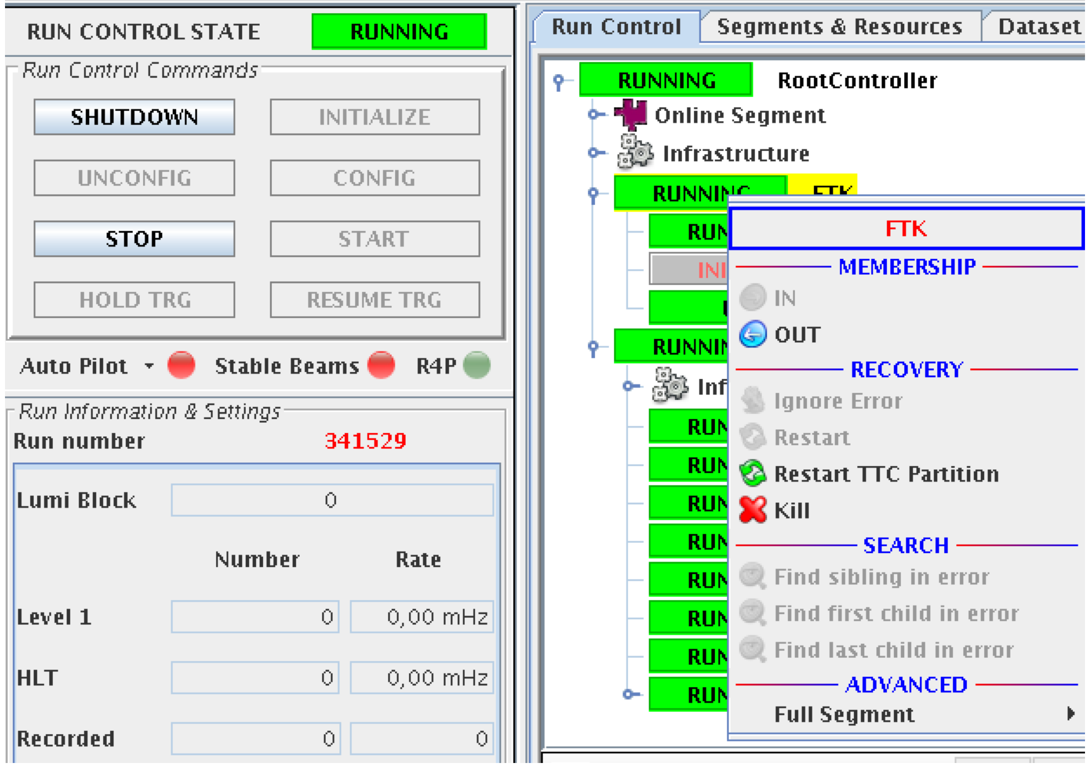
For more details see Todd's talk

- FTK: hardware system based on pattern recognition able to find and reconstruct tracks in the ATLAS ID
- Data driven and massively parallel processing system
  - 8k Custom Associative Memory (AM) chips for pattern matching
  - 2k FPGAs for track fitting, data preparation, ambiguity resolution …
- Very complex system!
  - > 450 boards and many interfaces
    - ~400 input links (from ID)
    - O(10)k links in total
  - Data flows among boards
    - Both vertically and horizontally
    - Synchronization between all the boards crucial
  - Two different standards used: VME and ATCA



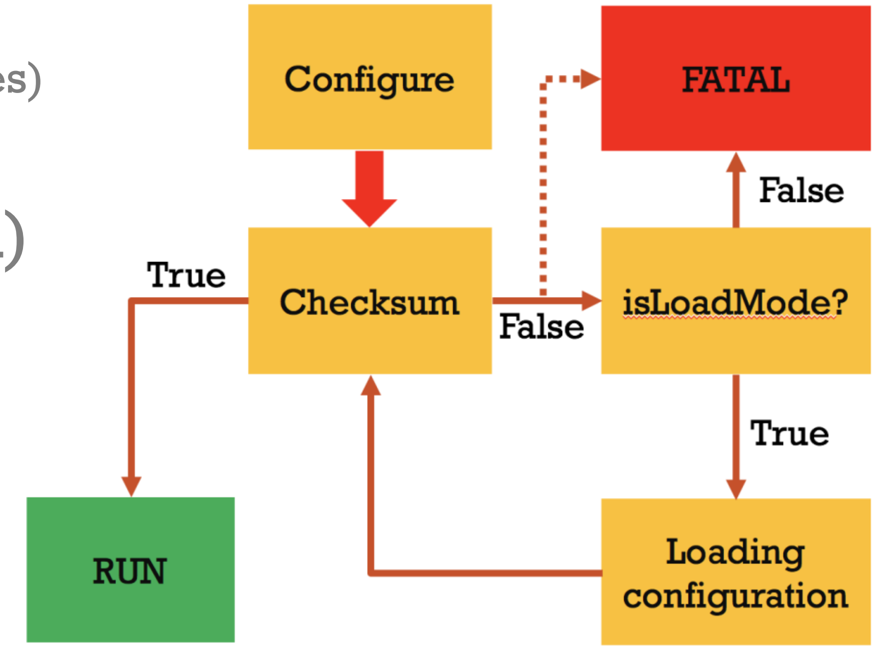
FTK CONTROL SOFTWARE

- Current FTK goal: achieve stable processing providing tracks to HLT in a $\eta - \phi$ region of ATLAS
  - FTK Online SW strategy: make use of the available ATLAS tools while spotting problems and bottlenecks
    - Custom solutions will be introduced during Long Shutdown
- ATLAS control SW based on Finite State Machine
  - FTK configuration needs to fit in FSM transitions
  - System configuration very complex
    - Many links to be set-up in the correct order
      - Definition of FTK specific sub-transitions required to cope with configuration complexity
    - Many operations required, some to be repeated in case of errors



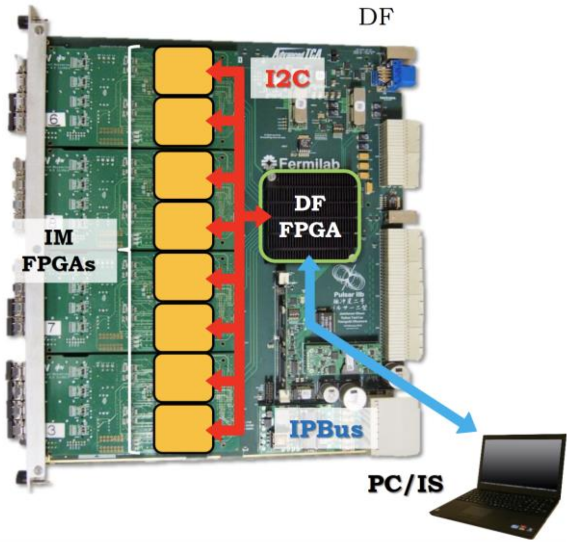
FTK CONFIGURATION

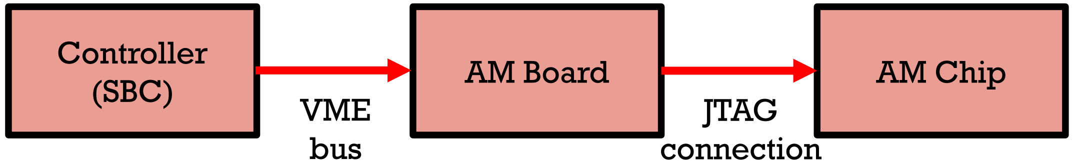
- FTK boards configuration:
  - Pattern banks and constants to be loaded on AMChips and FPGAs
    - Not stored in permanent memory
  - SBCs with limited memory (4GB) and CPU power (1.9 GHz - 4 Cores)
- FTK configuration loading time larger than ATLAS configure transition (O(1)h vs O(1)min)
  - Configuration data size ~ 500 MB/board
  - Configuration time ~ 5 min/board
- Data loading moved to interfill period
  - During ATLAS configure transition just verify the validity of the configuration
    - Using checksums computed by both FW and SW on the whole memory content
  - Tool for automatic generation of FTK configurations developed



FTK BOARD ACCESS INTERFERENCES

- Big effort to make board access thread safe
  - Race conditions observed during board configuration
- SW solutions implemented
  - Mutexes to serialize board access from same process
  - System semaphores for multiproces accesses
  - Cons:
    - Board monitoring slowed down
    - Control software less responsive
- Work ongoing on concurrent access management at FW level
  - Where possible



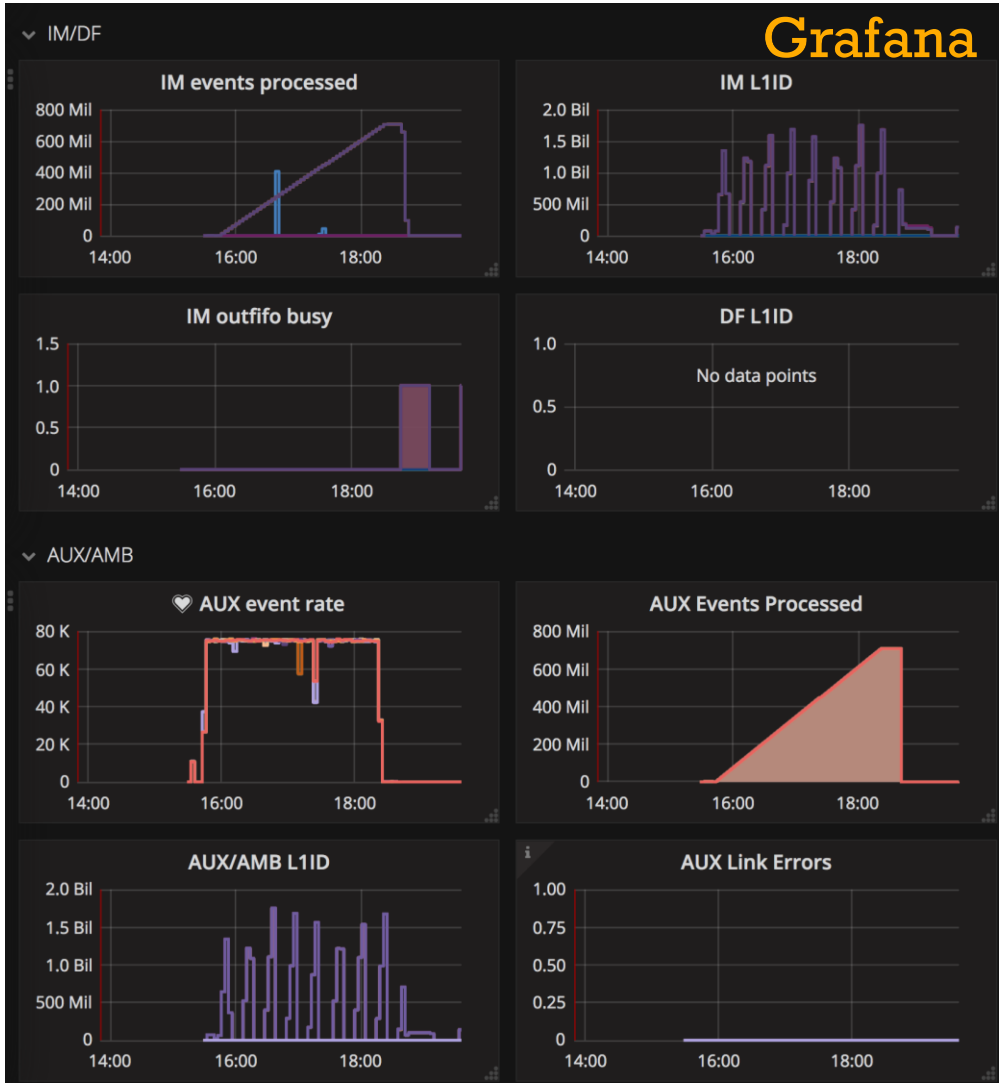


FTK REMOVAL AND RECOVERY

- ATLAS provides procedures to remove and recover faulty subsystem parts while running
  - Challenging for FTK
- Removal: allows excluding part of the readout blocking the trigger
  - Output replaced with empty fragments
  - FTK Removal requires deep knowledge on possible failing conditions
    - Currently triggered when too many fragments are missing
- Recovery: allows re-activating components previously disabled
  - Recovering the system behind the scenes and stopping the L1 accepts only when FTK re-joins the data taking session
  - FTK needs to recover while ATLAS is taking data
    - Recovery operations require definition of FTK internal FSM
    - Application to manage internal FSM and commands dispatching developed

FTK MONITORING

- FTK monitoring sprinted out by the advancement of the commissioning
- Currently focused on board monitoring
  - Status of the internal dataflow
- Information stored in board status registers
  - monitoring thread in Controller application
    - Data collected via VME/IPbus
    - Published in ATLAS Information and Histogram services and displayed in high level monitoring applications
- Work ongoing on online validation of FTK output
  - Required in view of stable FTK integration
  - Fragments sampled from ATLAS dataflow
  - Online comparison with simulations

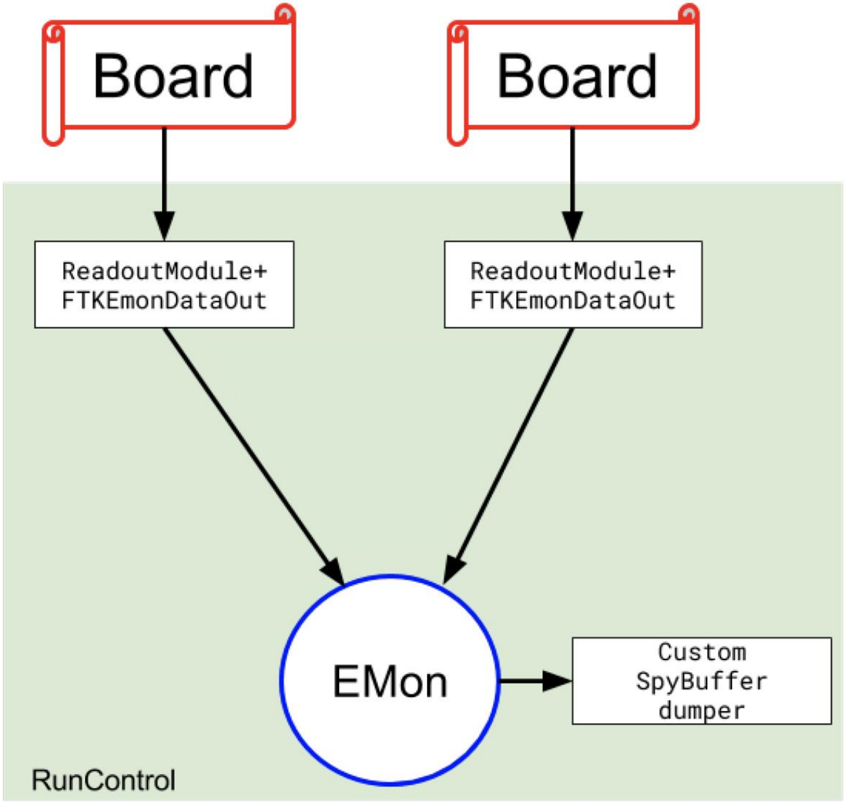


FTK SPYBUFFERS MONITORING

- FW debugging requires collection of board input/output data
  - Require data of the same event for all the boards
  - Blocking the monitoring registers writing as soon as an interesting error occurs
- Spybuffer:
  - large register in which monitoring data are stored
  - Typical use-case is monitor input/output of a FW module prior to an error
  - FTK Spybuffers are circular buffers
- Freeze:
  - Operation that triggers the board's FW to deliberately stop updating its Spybuffers
  - Affects only the monitoring registers, not the processing FIFOs
- Online Spybuffer readout system provided
  - Based on ATLAS Event monitoring (EMon)

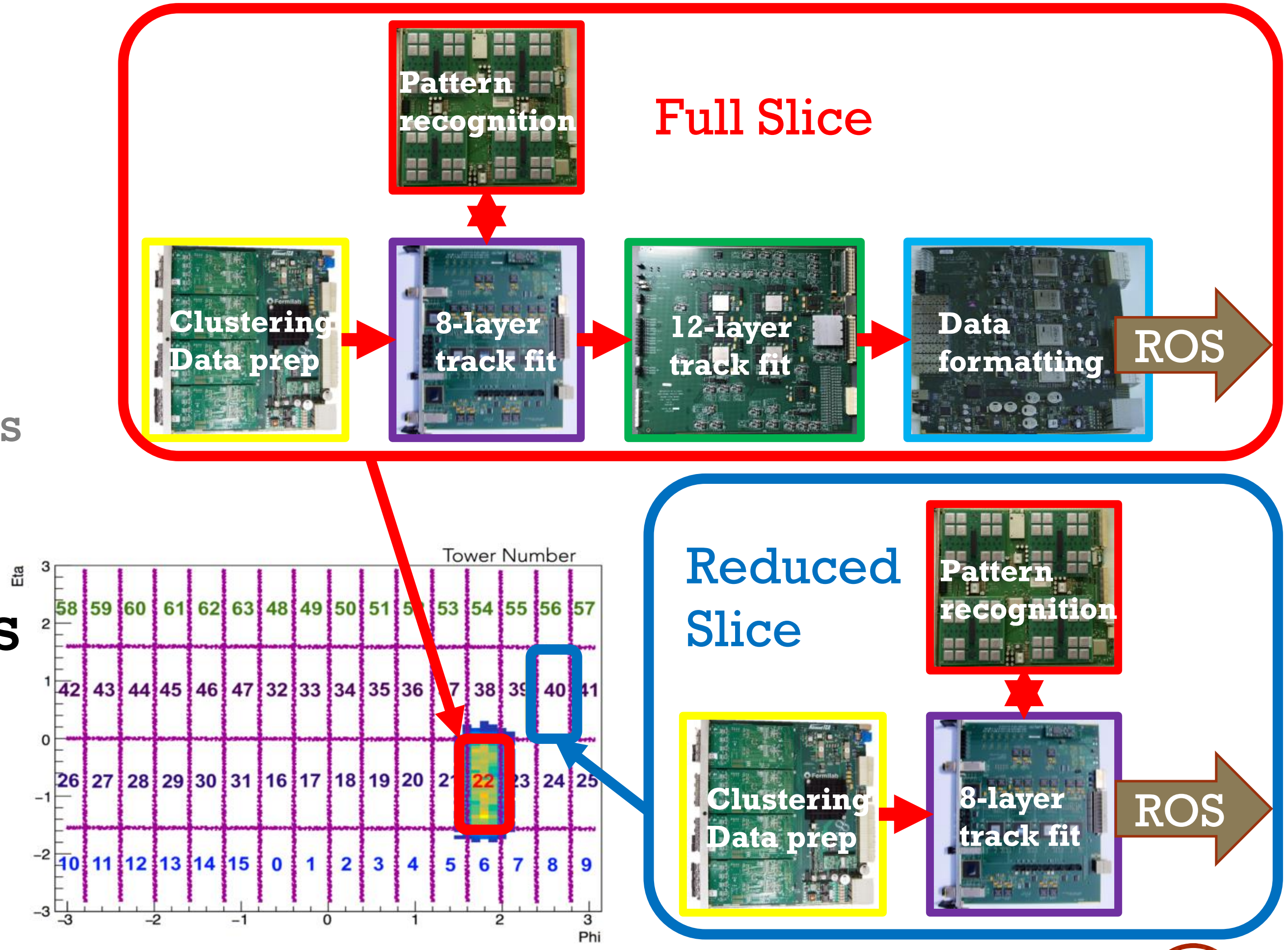
FTK SPYBUFFERS MONITORING

- Current readout logic:
  - Spybuffers collected from the boards by the control application
    - Periodically, through monitoring thread
  - Data made available through the ATLAS EMon service
- Spybuffer readout is slow
  - Serialization mutex locked too long
  - Dataflow monitoring stopped during the Spybuffer readout
    - It should have higher priority
  - Useful information only from Spybuffers containing problems
- Readout logic optimization
  - Spybuffers read in case of error only
  - Decision driven by a dedicated status register
  - Pre-defined set of selection criteria defined in configuration



FTK STATUS: COMMISSIONING

- Commissioning focuses on two slices
  - Full slice outputting 12-layer tracks
  - Reduced slice outputting 8-layer tracks
    - Used for commissioning upstream boards
  - Slices stably integrated in ATLAS data taking architecture
- First validated FTK 12-layer tracks written to the ATLAS bytestream
- In parallel, working on scaling up the system



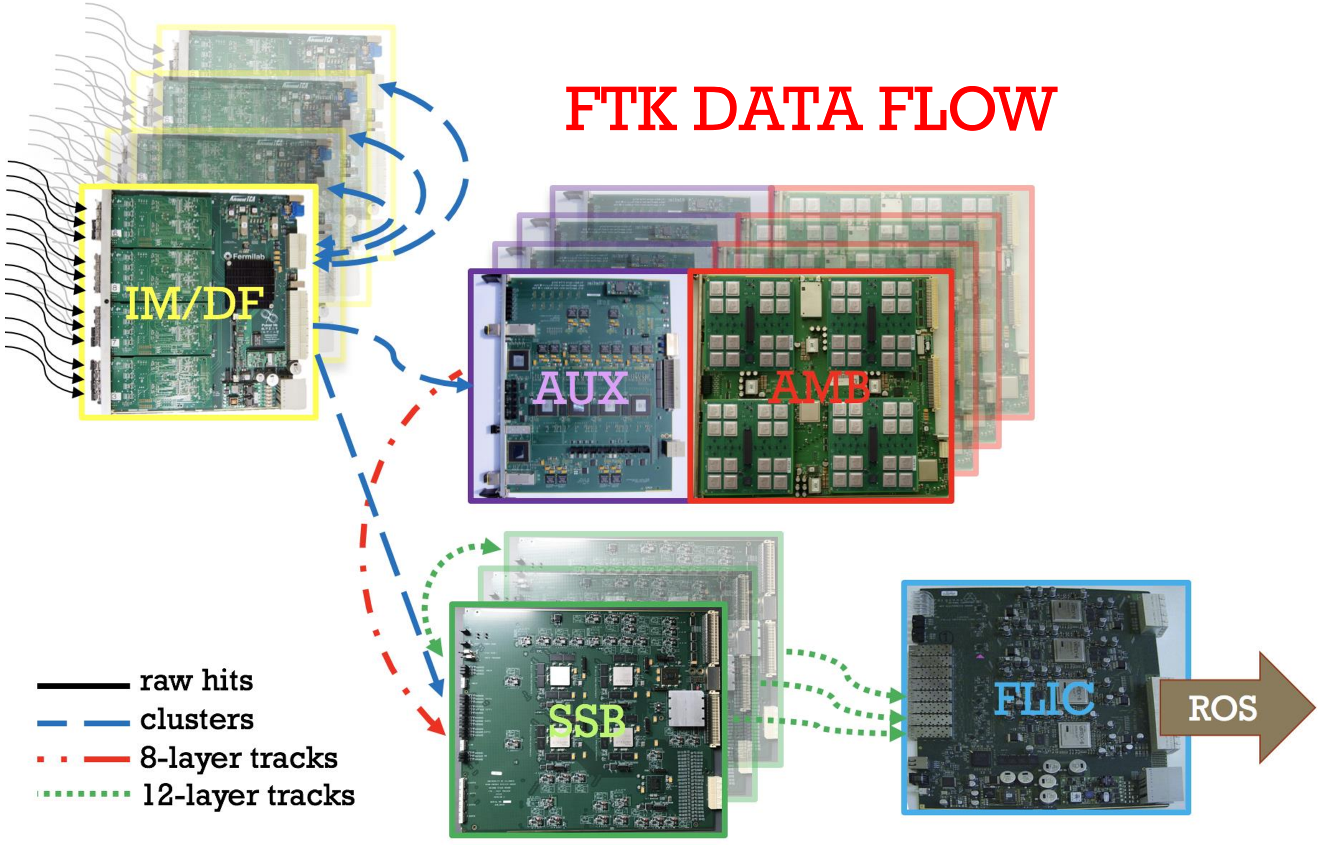
CONCLUSIONS

- FTK system currently under commissioning
  - Made steady progress in the last year
  - First 12-layer tracks written in the ATLAS bytestream
  - FTK system well integrated within ATLAS data taking
- Integration of FTK in ATLAS Control SW challenging
  - SW stable and in continuous development
  - Current architecture has some limitations and bottlenecks
    - Custom solutions will be introduced during LS2
- FTK monitoring sprinted out by the advancement of the commissioning
  - Big effort in boosting up dataflow and Spybuffer monitoring to ease debugging

BACKUP

OVERVIEW

- FTK system
  - **IM**: receive, cluster hits from Inner Detector
  - **DF**: distribute data to FTK towers
  - **AUX**: reduce data for pattern recognition, 8-layer track fits
  - **AMB**: 8-layer pattern recognition
  - **SSB**: 12-layer track fitting
  - **FLIC**: data formatting for HLT



FTK CONFIGURATION GENERATOR

- ATLAS configuration defined in xml files stored on CVS
  - Usually managed individually through custom configuration editors in Java
  - Cumbersome operation for FTK especially during commissioning
- A software package has been deployed to generate the FTK configurations
  - Python tool able to generate configuration files for FTK
  - Configured via a python dictionary with editable options
  - Used to generate dynamically the whole FTK configurations

MONITORING STATUS

- Board DataFlow: check that the sub-systems are not stuck – processing is ongoing
  - Status register data published in IS
  - Widely used by FTK experts (e.g. Grafana, custom tools, …)
  - Big effort to define optimal variables for monitoring purpose
- Board DataQuality: check that sub-systems produce a reasonable output
  - Information from SpyBuffers published in emon
  - Currently available for 2 boards
- FTK output DQM: check FTK output is reasonable
  - using FTK fragment in ATLAS event
  - Framework available, not currently in use
- High-level FTK output monitoring: compare FTK output to FTK emulation run on the same ID hits
  - ATLAS events containing FTK and ID fragments
  - Advanced/stable prototype available, to be speed up

WHERE CAN FTK HELP?

- Trigger and DAQ Run3 goals:
  - Reduce HLT rate by rejecting events which are background
  - Add sensitivity in channels or open sensitivity to new channels by improving the HLT
- FTK will help reaching these goals
  - Jet reconstruction and pileup reduction using data from all pile-up vertices in event
  - Better isolation requirements avoiding inefficiency from pileup
  - MET: tracking can reduce effects of pileup
  - τ : FTK tracking at early stage will avoid inefficiency from calo only selection
  - b : FTK tracks will save HLT time allowing L1 threshold lowering
  - …