EURIZON Annual Meeting 2023

WP2 session

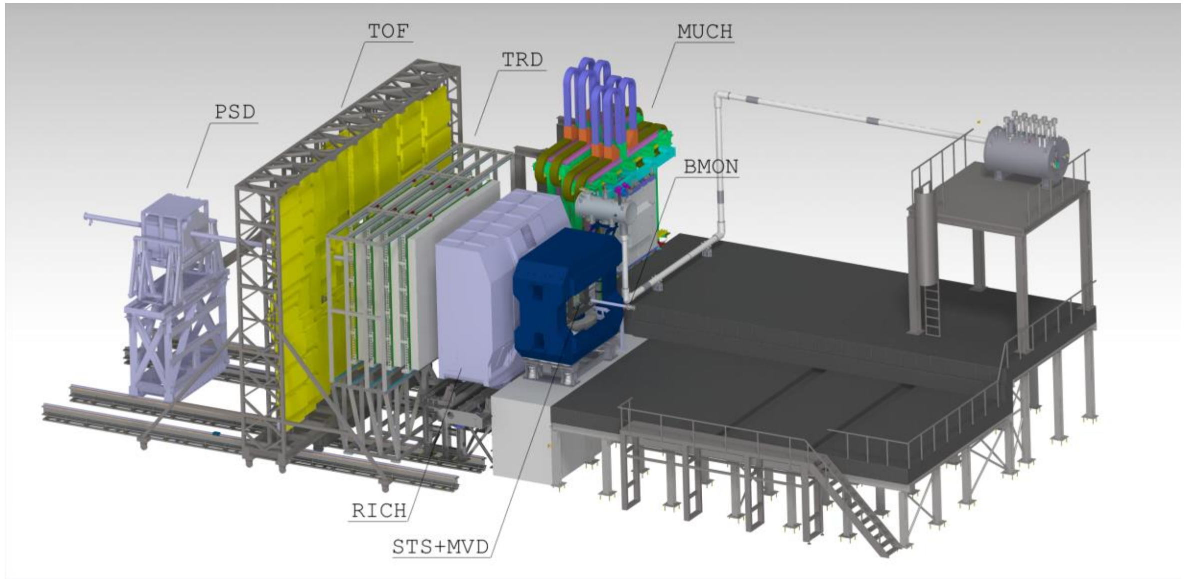
Central components of the CBM readout: Hardware and Software

- (m)CBM data Acquisition
- DAQ software components
- Online processing
- Controls

WP 2.2: Developments for the data acquisition chain, for data preprocessing and computing for mCBM and CBM at FAIR

eurizon
European network
for developing new horizons for RIs

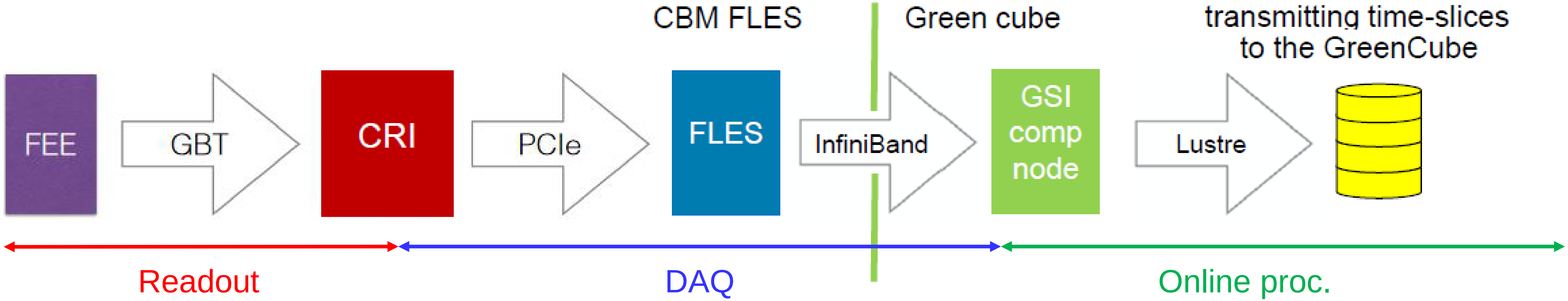
CBM data Acquisition



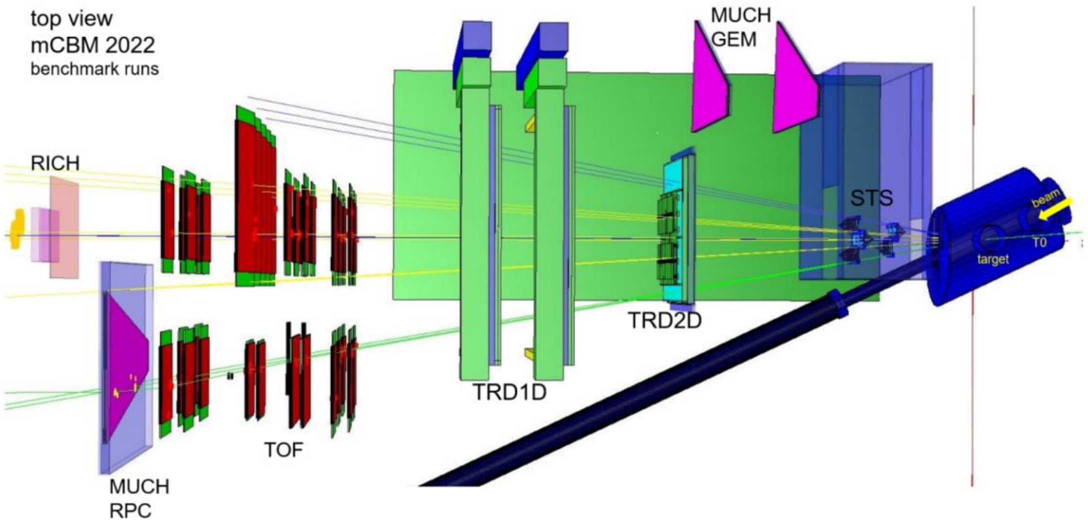
- 8 main detector systems
- Fully free-streaming acquisition (no HW trigger)
- Online processing computing farm in the GSI/Fair "Green IT Cube": O(100 k) cores

3 main stages:

- Readout chains: data generation and system specific transport
- DAQ: data organization and generic transport
- Online processing: data reconstruction and selection, archiving



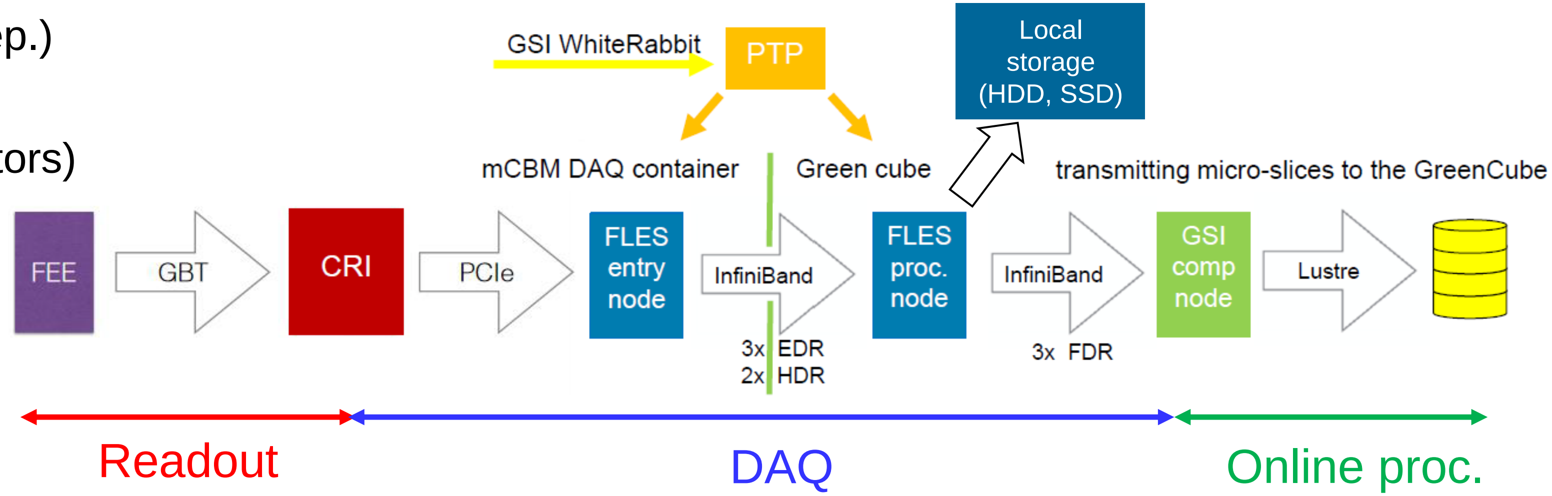
Prototype: mCBM (2nd phase, 2021-2022+)



- Permanent setup at GSI in dedicated cave
- SIS18 beamline
- Prototypes of 7 out of 8 CBM detector systems
- Up to 5 used at a time for physics analysis
- Reference measurement: $\Lambda \rightarrow p\pi^-$ decay
- Similar "HW/SW stress environment" as CBM
- O(100 TB) recorded in 2022 campaign

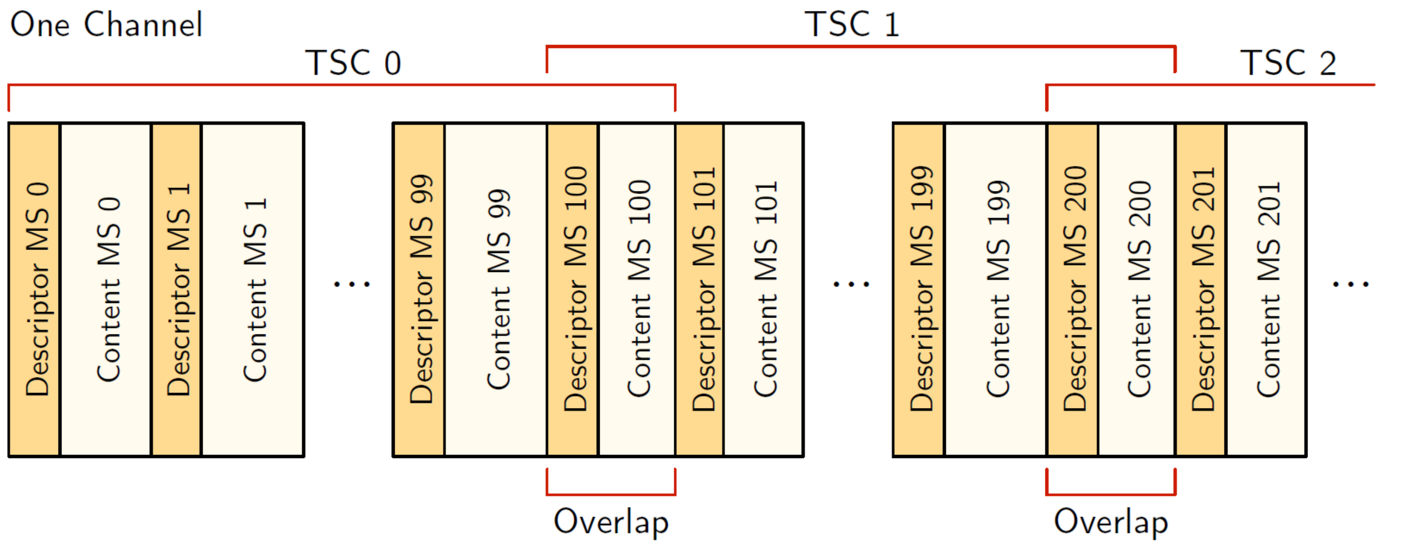
<u>Main differences to CBM in acquisition:</u>

- Reduced connectivity (3-10% CBM, detector dep.)
- Non final readout interface boards
- Evolving setup (each year closer to CBM detectors)
- In 1st and 2nd phases, no online data selection
- In 1st and 2nd phases, link from entry cluster to compute cluster not used in production
- Limited/prototype central coordination features

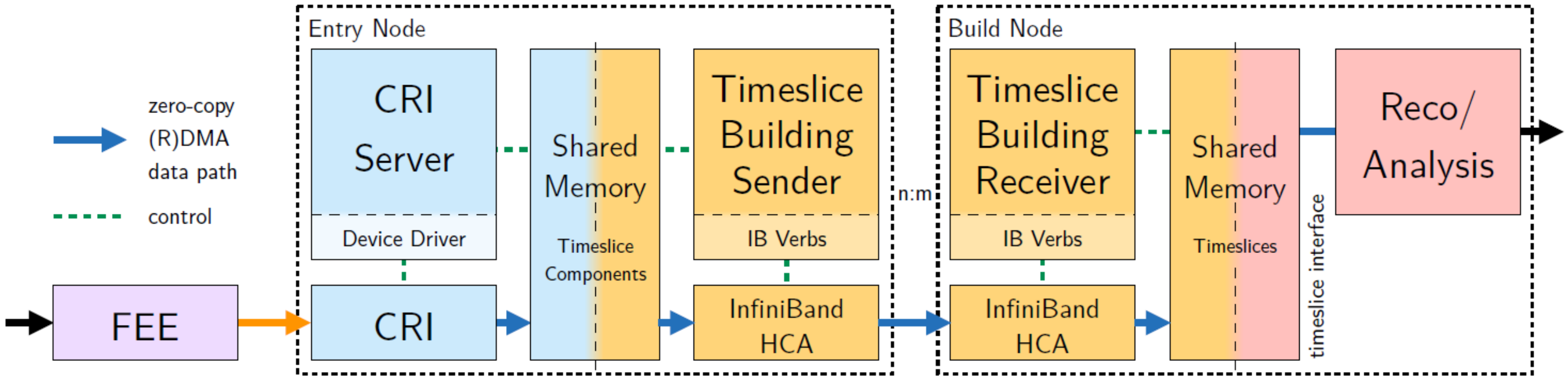


DAQ Software

- Data model:
  - microslices: self-contained blocks of data from a single source for a given period of time
  - timeslices: collections of microslices from multiple sources, overlapping over time
- FLES software (aka Flesnet):
  - Driver reading from CRI though PCIe
  - Timeslice builder sender/receiver
    - ⇒ Internal data transport & flow control
    - ⇒ Timeslice assembly



- mCBM specific FLES software:
  - Timeslice client
    - ⇒ Archiving of timeslices
    - ⇒ Publishing for monitoring
    - ⇒ Publishing for processing
  - InfluxDB + Graphana
    - ⇒ FLES monitoring



DAQ Software: operating mCBM in 2022

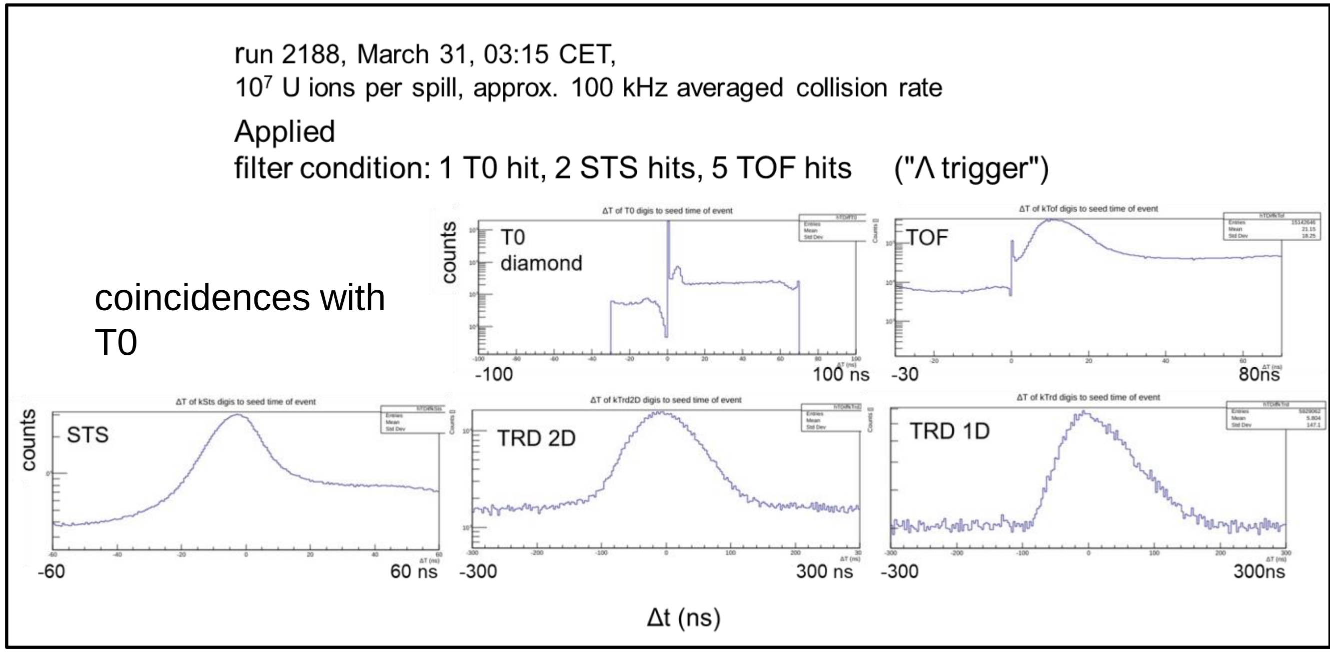
- No complete Experiment Control Software (ECS) available
- Additional layers added, with a CLI, to
  - Keep track of runs and their configuration
  - Start the cluster processes needed to perform the acquisition and archiving
  - Synchronize all systems before starting the readout
  - Enable/Disable data emission in a synchronized way for all systems

- Typical run sequence:
  - Ask detector shifts if they are ready and have their system configured
  - If needed, created a “configuration tag” selecting or excluding detector systems
  - Start the FLES run with a given recording/readout configuration tag
  - Common-start: re-synchronize the time counters of all participating systems, then communicate to all a common start point for data emission in future (5-6 s), strict for all
  - Monitoring of data rates in FLES, of data quality in data sample, …
  - Common stop: communicate to all system a common stop point for data emission in future, can be loosely obeyed
  - Stop the FLES run, which closes all processes in right sequence and archive some run logs

Online Processing: plans and mCBM

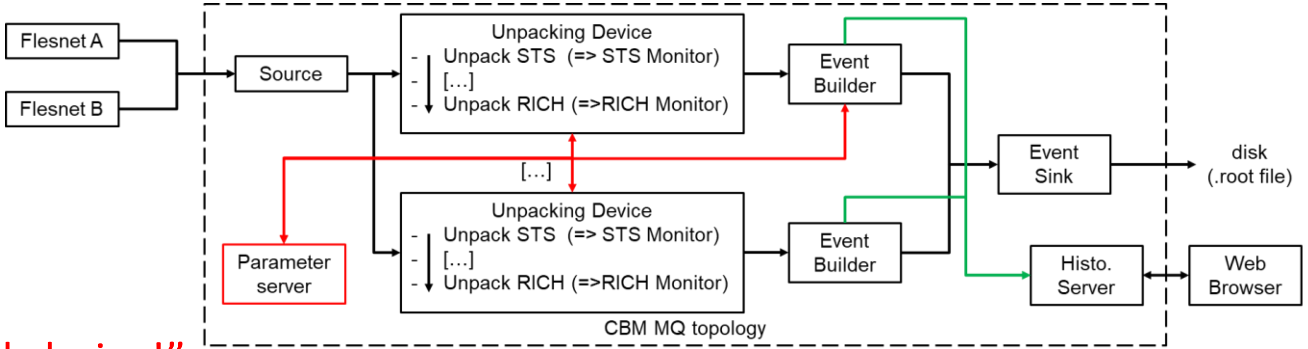
CBM plan

- Data unit for online processing = Timeslice
- Depending on physic cases
  - reconstruction from single detector to full setup tracking
  - event definitions may vary, ground for decision may not cover all signals
    - ⇒ "Event seeds" more than events
- Selection of data from looser time region around these event seeds for archiving
  - ⇒ allow more involved offline calibration if needed



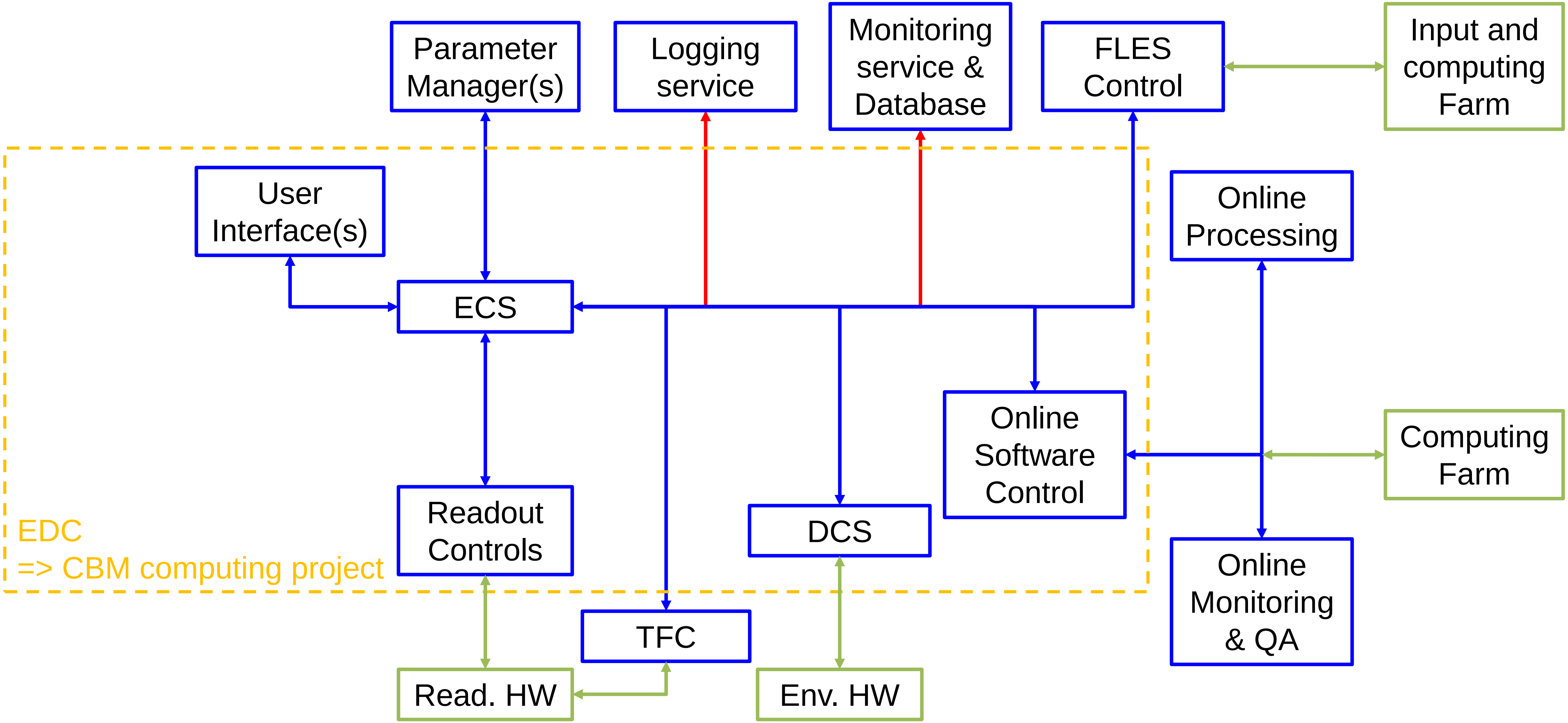
mCBM usage

- Bring stages of offline selection in FairMQ devices
- Online testing with coarse controls
- Used for 1st stage of mass offline analysis
- Now testing "replay" Online controls with ODC/DDS



⇒Disclaimer: "these solution are maybe/probably not our final choice!"

My main Worksite: Controls? EDC? ECS?

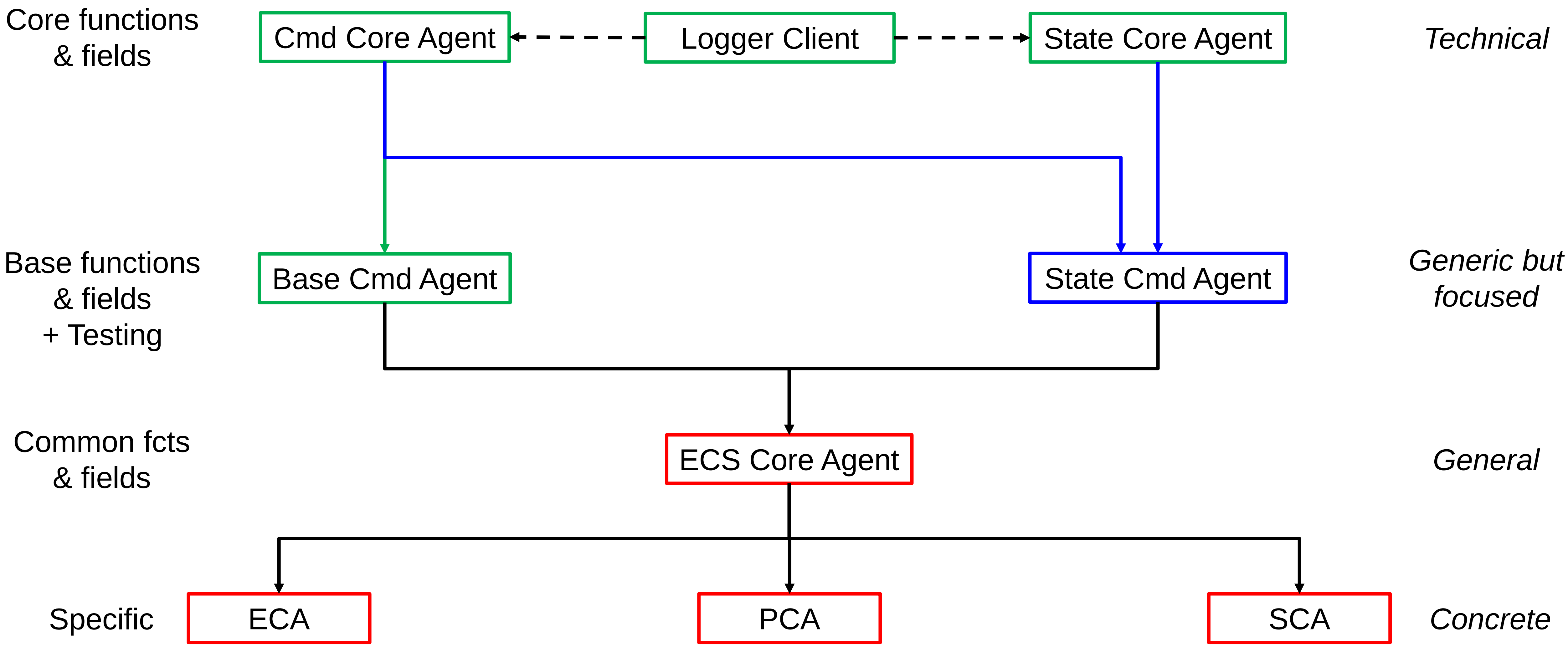


Readout controls

- C++ driver and low level management interfacing to the controls of the CRI FW and the full readout chain
  - “Device Control Agent” or DCA
  - At least one instance per entry node, at most one per CRI
  - Completely event (action) driven, from RPC calls by the controlling Python layer
  - Allows for atomic requests, queued requests and timed actions (e.g. periodic read for monitoring)
  - For now mostly providing low level register access
  - but later planned to hold most of the control logic to provide “atomic commands” and avoid access conflict
  - ⇒ Framework by FAIR colleague and system specific code by FPGA experts

- Python control libraries implementing the user interface
  - For now (2021-2022) also including most of the control logic and sending single register R/W requests
  - Later planned to mostly send “Atomic Command + parameters” , leaving register access to DCA
  - Based on client-server architecture to allow “common commands”,
    - e.g. asking all detector systems to (re-)synchronize
  - Currently stand-alone with either CLI interface or central scripts sending command to all systems
  - Will be the main interlocutor of the ECS agents controlling detector systems
  - ⇒ Framework by myself and system specific code either by me or detector experts

ECS: Modular concept



European network
for developing new horizons for RIs

Thank you for your attention

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 871072

For more information:

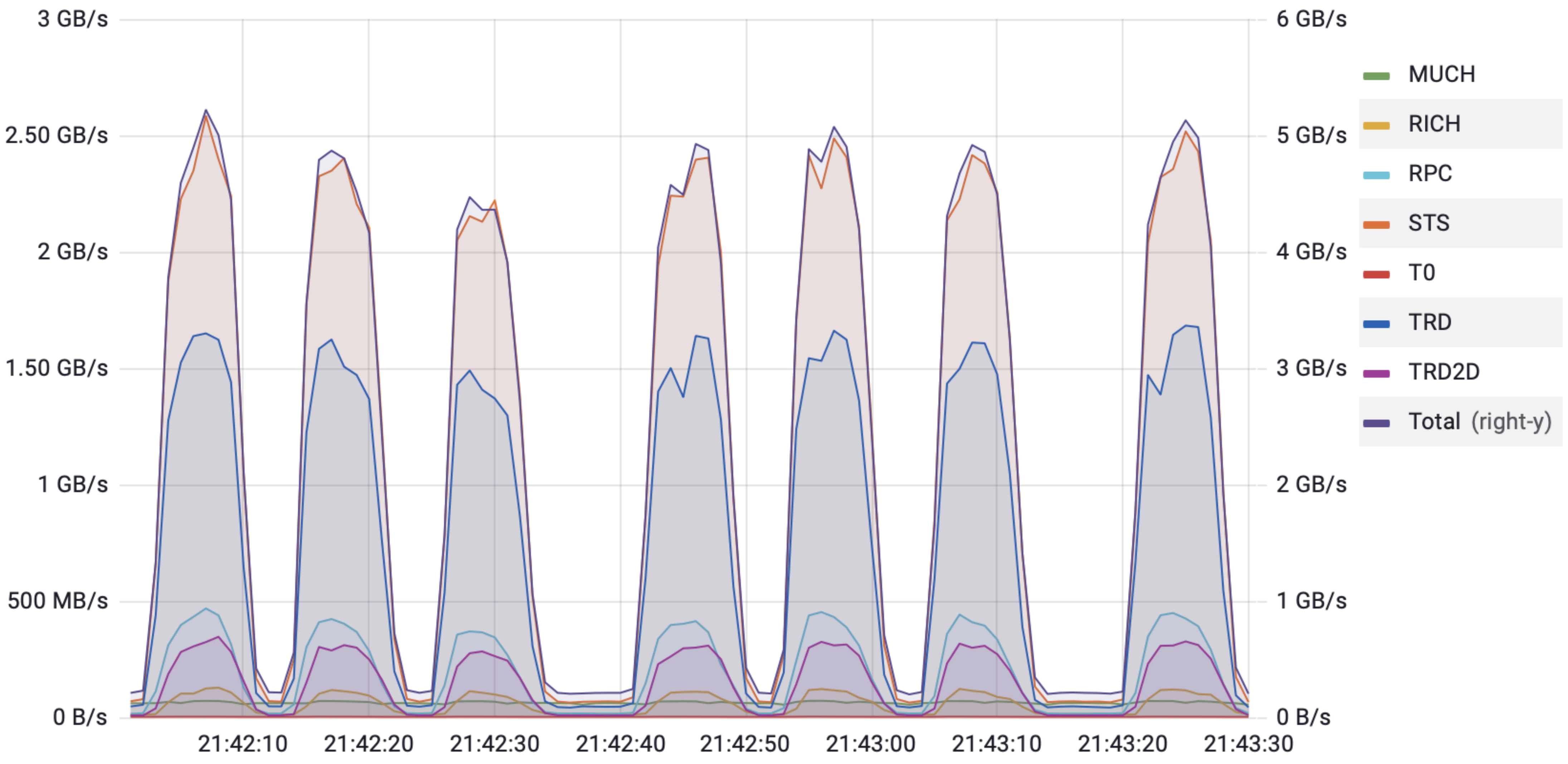
- *"Online Systems – Part I: DAQ and FLES Entry Stage", CBM TDR, currently under review and should be public sometime in 2023*
- *"CBM Progress Report 2021", 2022, https://repository.gsi.de/record/246663*
- *"mCBM proposal", CBM proposal to GSI G-PAC, 2017, https://repository.gsi.de/record/220072*

WP 2.2: Software for DAQ, Controls, Data analysis

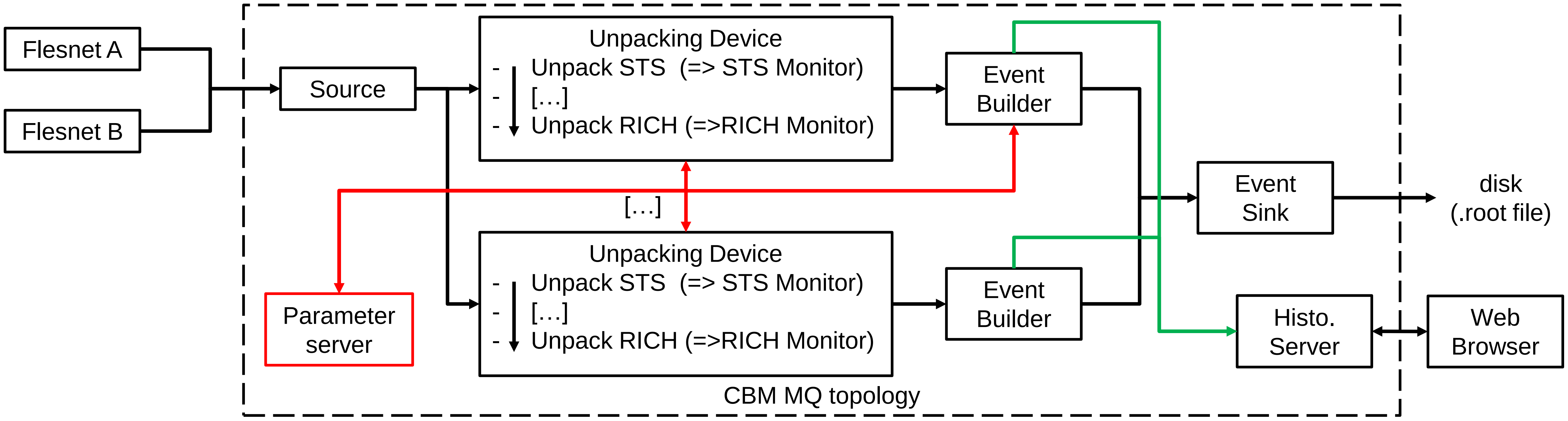
Assignee: P.-A. Loizeau

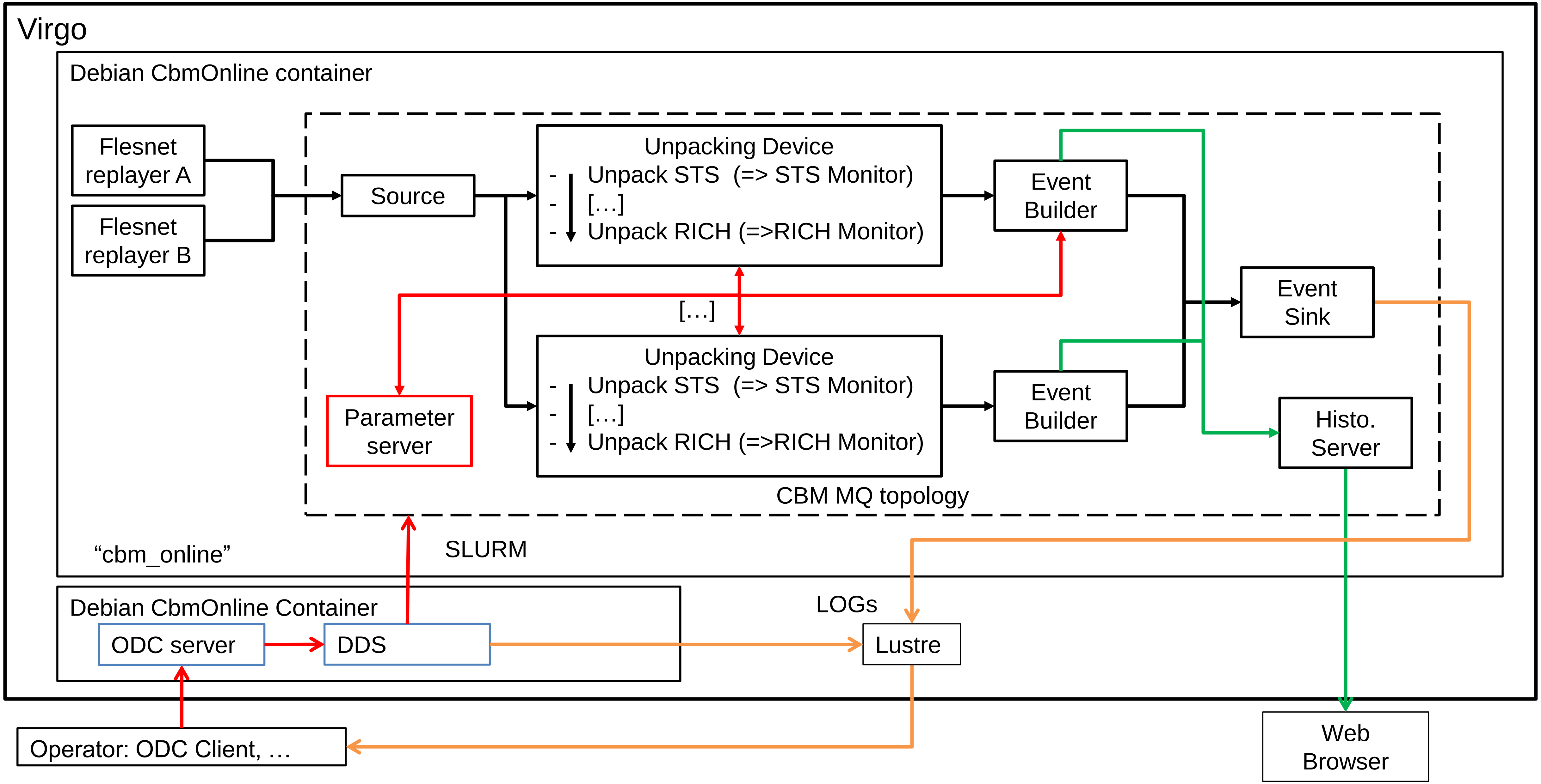
- Activities linked to WP2 in 2020-2022 period:
  - Finalization and support of the control SW for the AFCK based readout chain for the STS-XYTER (CBM STS laboratory, Kolkata CBM MUCH laboratory, mCBM 2020, BM@N tests)
  - mCBM
    - On-site support in the operation of the CBM DAQ prototype for mCBM 2020-21-22 campaigns
    - Development of a Python framework to control the CRI boards (High level part) and help for related detector-oriented developments, used for production in mCBM 2022 campaign
    - Development and support of online data monitoring tools for various mCBM detectors
    - Development and support of raw data unpackers and of event builders for mCBM
    - Test-deployment of these on the mFLES and Virgo clusters at GSI with the FairMQ framework to validate online processing concepts and test physical connections
  - Coordination of the “Experiment and Detector Controls” CBM computing project
  - Development of an “Experiment Control System” prototype for mCBM and CBM
  - General support and development with Cbmroot for CBM simulation and data analysis

Data rates in a mCBM DAQ high rate run (r2448)

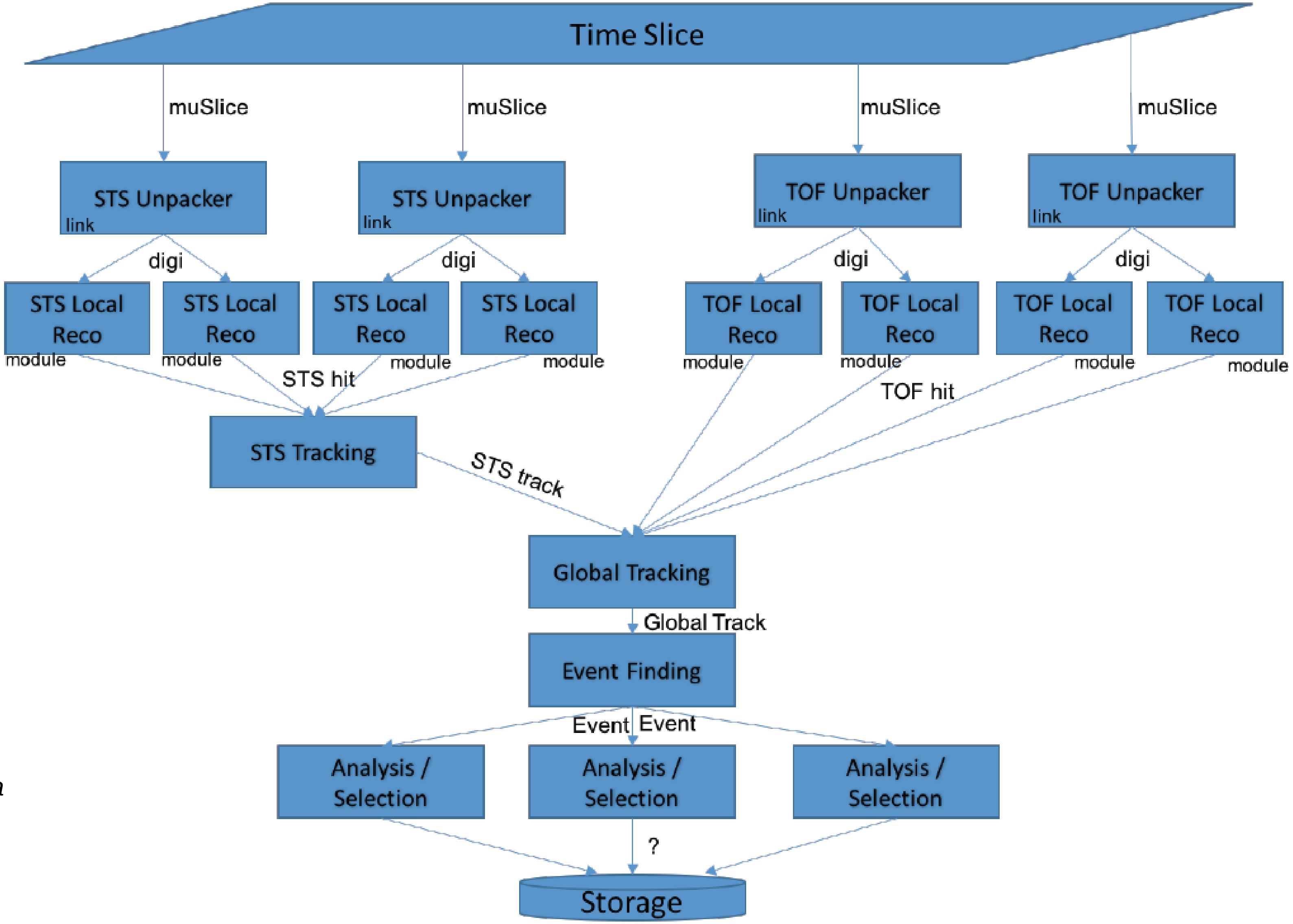


mCBM MQ test


Flesnet A
Flesnet B
Source
Unpacking Device
- Unpack STS (=> STS Monitor)
- […]
- Unpack RICH (=>RICH Monitor)
Event Builder
[…]
Unpacking Device
- Unpack STS (=> STS Monitor)
- […]
- Unpack RICH (=>RICH Monitor)
Event Builder
Parameter server
Event Sink
disk (.root file)
Histo. Server
Web Browser
CBM MQ topology


Virgo
Debian CbmOnline container
Flesnet replayer A
Flesnet replayer B
Source
Unpacking Device
- Unpack STS (=> STS Monitor)
- […]
- Unpack RICH (=>RICH Monitor)
Event Builder
[…]
Unpacking Device
- Unpack STS (=> STS Monitor)
- […]
- Unpack RICH (=>RICH Monitor)
Event Builder
Event Sink
Histo. Server
Parameter server
CBM MQ topology
"cbm_online"
SLURM
Debian CbmOnline Container
ODC server
DDS
LOGs
Lustre
Operator: ODC Client, …
Web Browser

Example: Simple CBM Process Graph (STS + TOF)



V. Friese
Theme Meeting on CBM
3rd February 2023, Jatni, India