Workshop X on Streaming Readout

TriDAS

Laura Cappelli – INFN-CNAF

Co-authors: *Tommaso* Chiarusi[1], *Francesco* Giacomini[2], *Carmelo* Pellegrino[2]

[1] INFN Bologna, [2] INFN-CNAF

Supported by Italian Ministry of Foreign Affairs (MAECI) as Projects of great Relevance within Italy/US Scientific and Technological Cooperation under grant n. MAE0065689 - PGR00799

May 19th, 2022

Agenda

- The framework

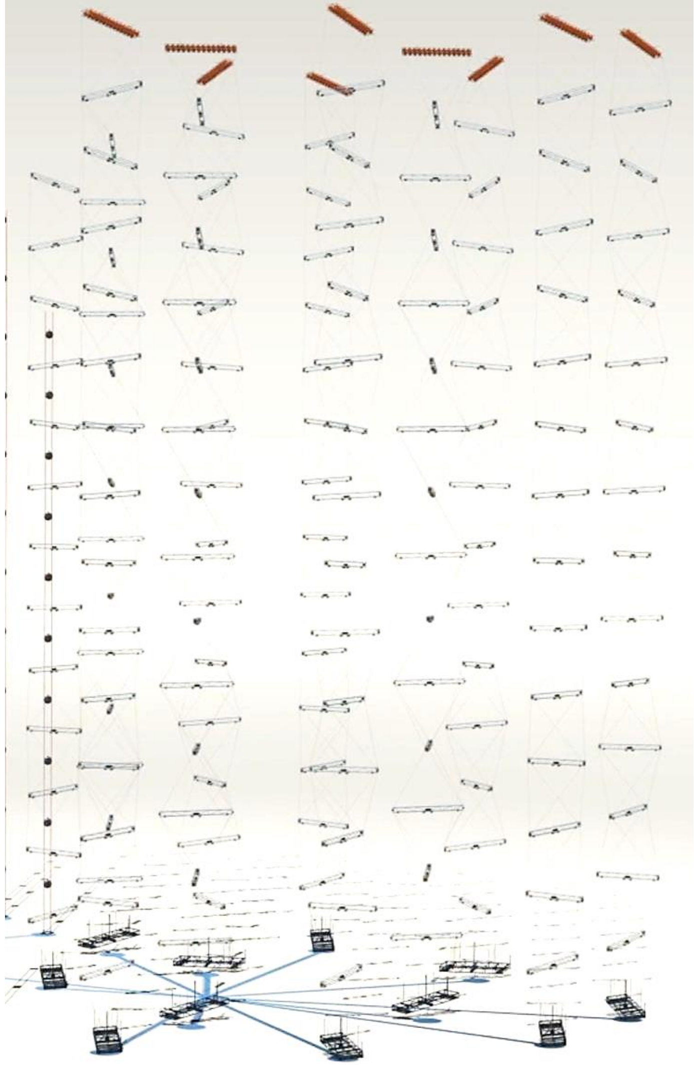
  What it is and how it works

- TriDAS@Jlab

  The project and the tests

The framework

The birth of TriDAS


The KM3Net Towers

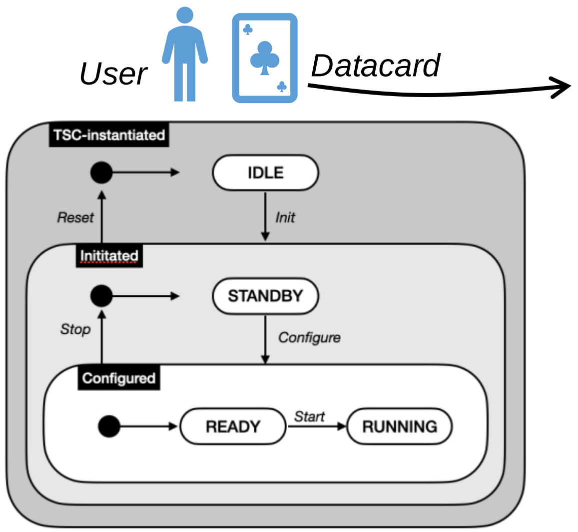
- Designed for streaming read-out of Astro-particle Physics events
  - NEMO project
  - KM3NeT-ITA (Towers) project

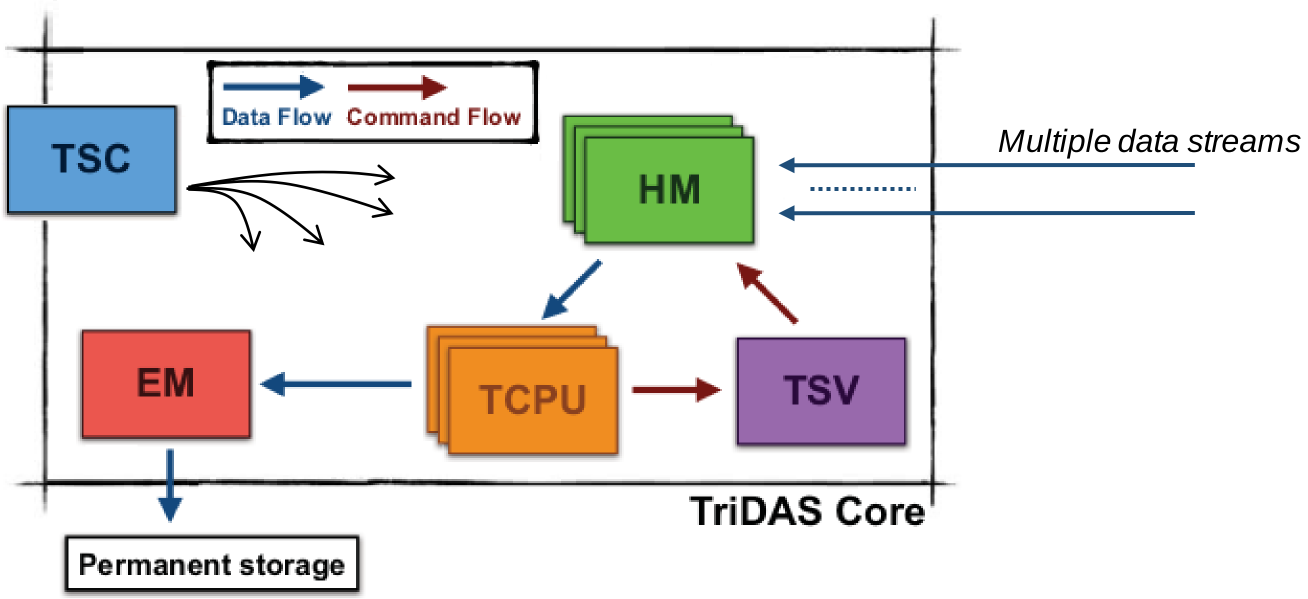
- ***All-data-to-shore*** approach: the software at the shore station
  - Reads the data streams
  - Reconstructs the events
  - Filters the data to collect only the interesting ones

The TriDAS framework

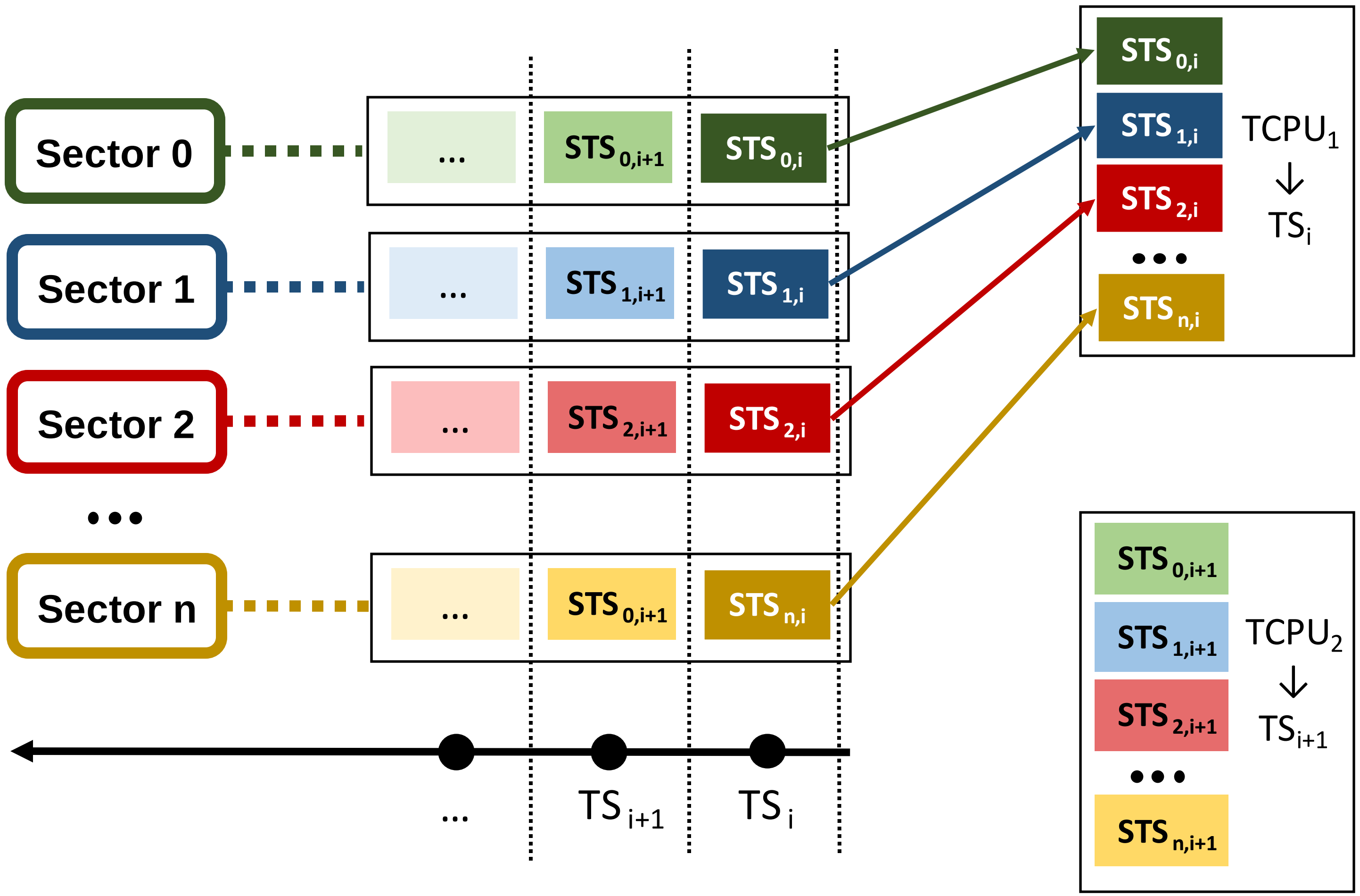
- TriDAS characteristics:
  - C++17 multithreaded software framework
  - Dependencies: CMake, ZeroMQ, Boost
  - State machine driven process
  - Flexible design:
    - Configurable via datacard (e.g. detector geometry)
    - L2 trigger algorithms in standalone plugins
    - Data format

- Composed by 5 modules:
  - HM (*Hit Manager*)
  - TCPU (*Trigger CPU*)
  - TSV (*TriDAS SuperVisor*)
  - EM (*Event Manager*)
  - TSC (*TriDAS System Controller*)
- The TriDAS code is available here



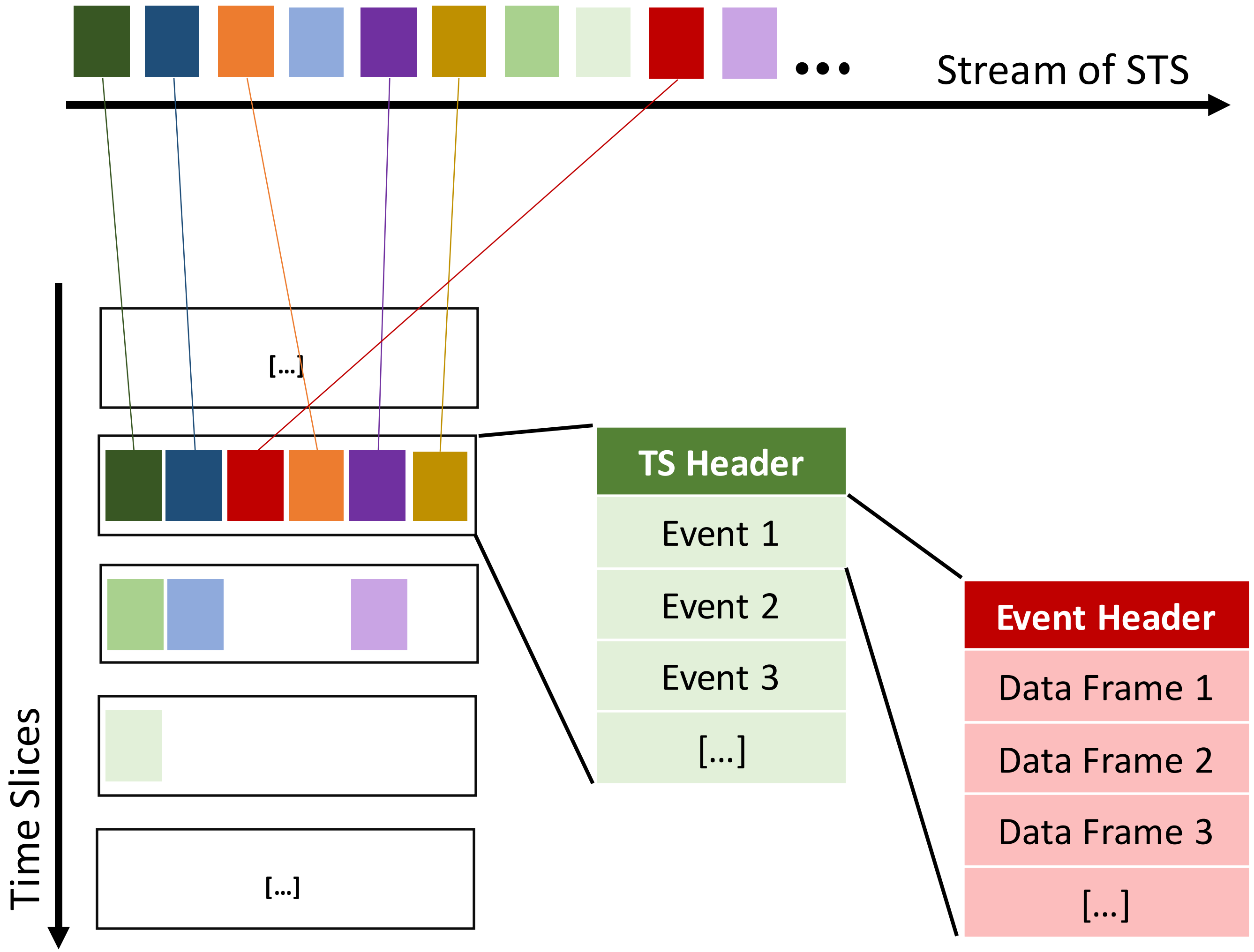


Data Flow: the HM aggregation



- Each HM:
  - Collects data from a specific sector of the detector
  - Subdivides the data into a sequence of time-ordered bunches called **Sector Time Slices (STSs)**
    - Fixed time duration called **Time Slice (TS)** chosen at run start time via datacard parameter (50ms in CLAS12)
  - Sends the STSs to a TCPU according to the token received from the TSV
- A TCPU receives all the STSs of a TS

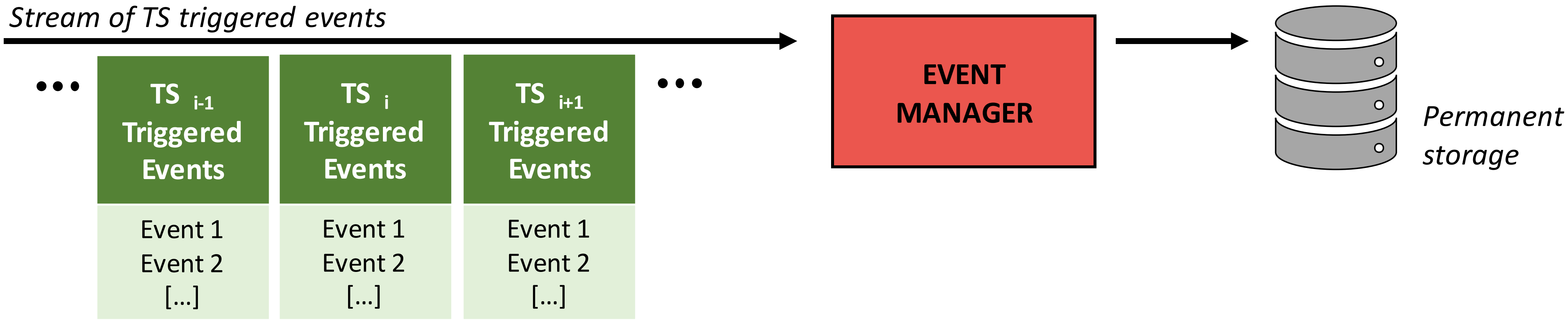
Data Flow: event building & trigger



- The TCPU:
  - Receives a stream of STSs
    - The STSs temporal order isn't guaranteed
    - There are many threads, each one arrange the data of a TS
  - Reconstructs events per time window
  - Applies one or more trigger algorithms
    - **External plugin** selected in the datacard
- At the end of the process, the TCPU has obtained a list of interesting events per TS
  - One event is composed by multiple hits
  - Events and hits found in a TS are time ordered

Data Flow: Event Manager

- The TCPUs send to the EM the triggered events for each TS
- The EM:
  - Writes the TSs into some post trigger files
- All the useful event information are stored for further analysis

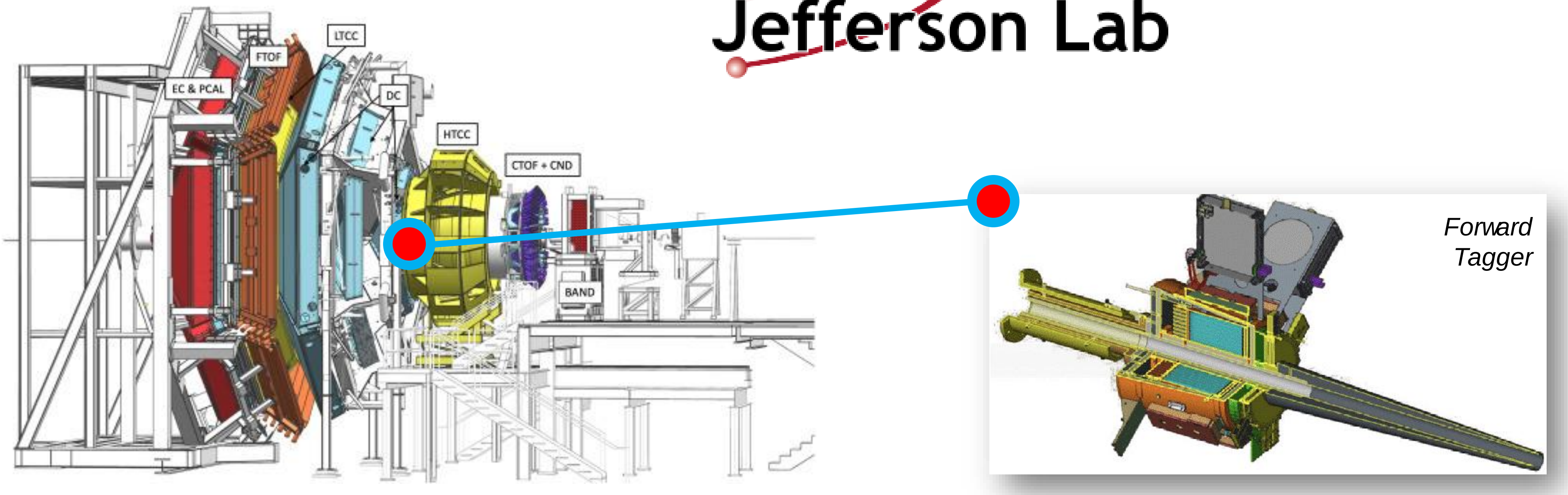


TriDAS @ JLab

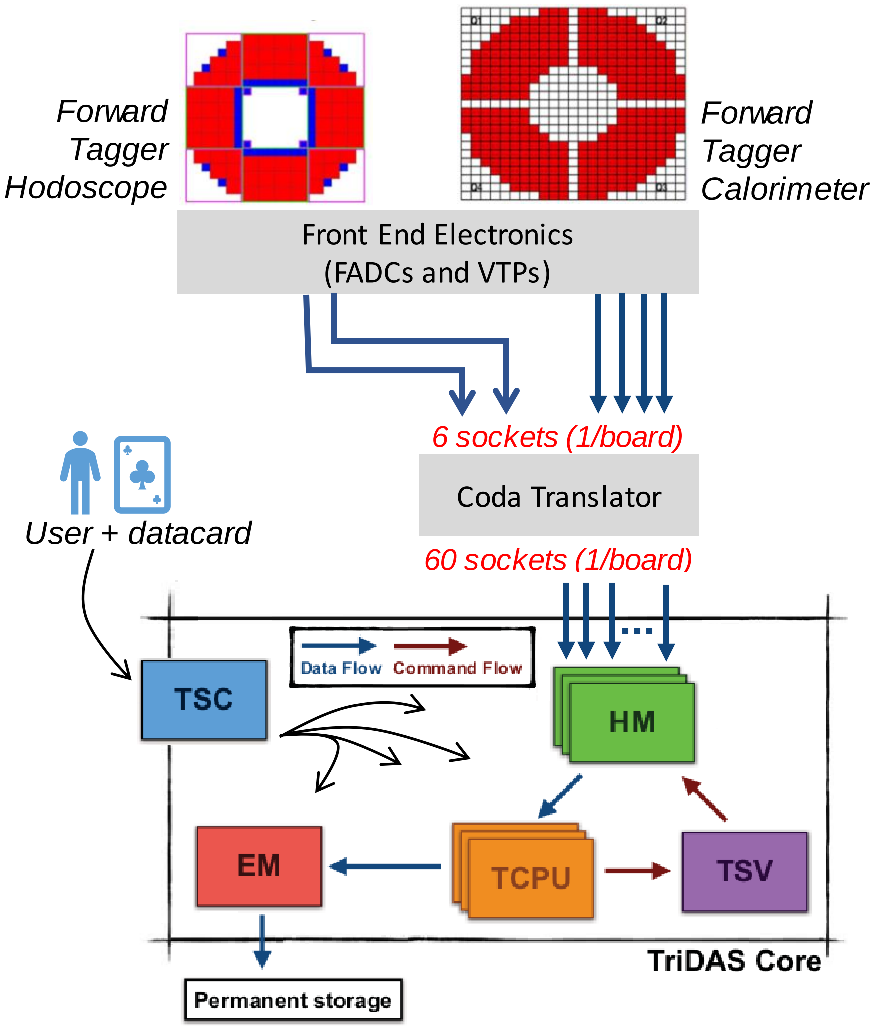
A new use-case

- Could TriDAS work efficiently in the DAQ of a nuclear Physics experiment on beam?
  - 2020: "as it is" TriDAS with the CLAS12 Forward Tagger
  - Operating in **triggerless** mode

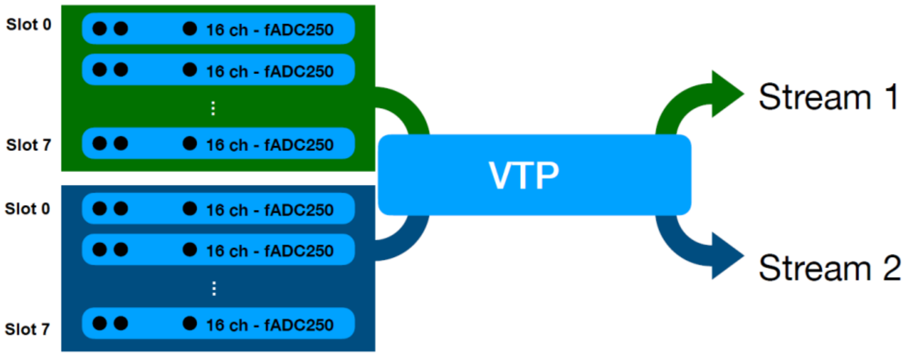
Jefferson Lab



Test in 2020 – Setup

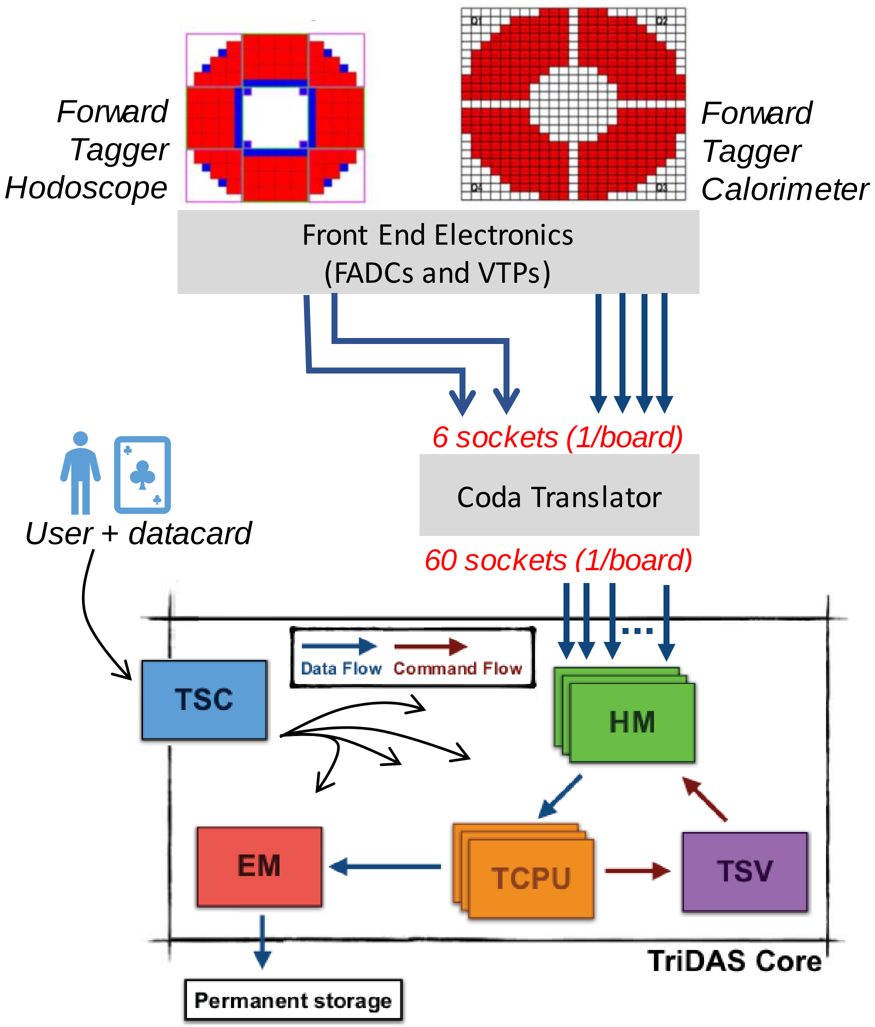


- System tested on the Forward Tagger sub-detectors
  - 6 streams: 2 streams from 3 VXS Trigger Processors (**VTP**)
    - Throughput: max 4 GBps per stream, set from few tens of MBps to 100 MBps
  - 16 Flash ADCs 250 (**FADC**) per VTP, 8 per stream
  - 16 channels per FADC
  - Total channels: 768 (16 ch * 8 FADC * 2 streams * 3 VTP)



- Between the front-end electronics and TriDAS there was the **CODA translator** layer

Test in 2020 – Results



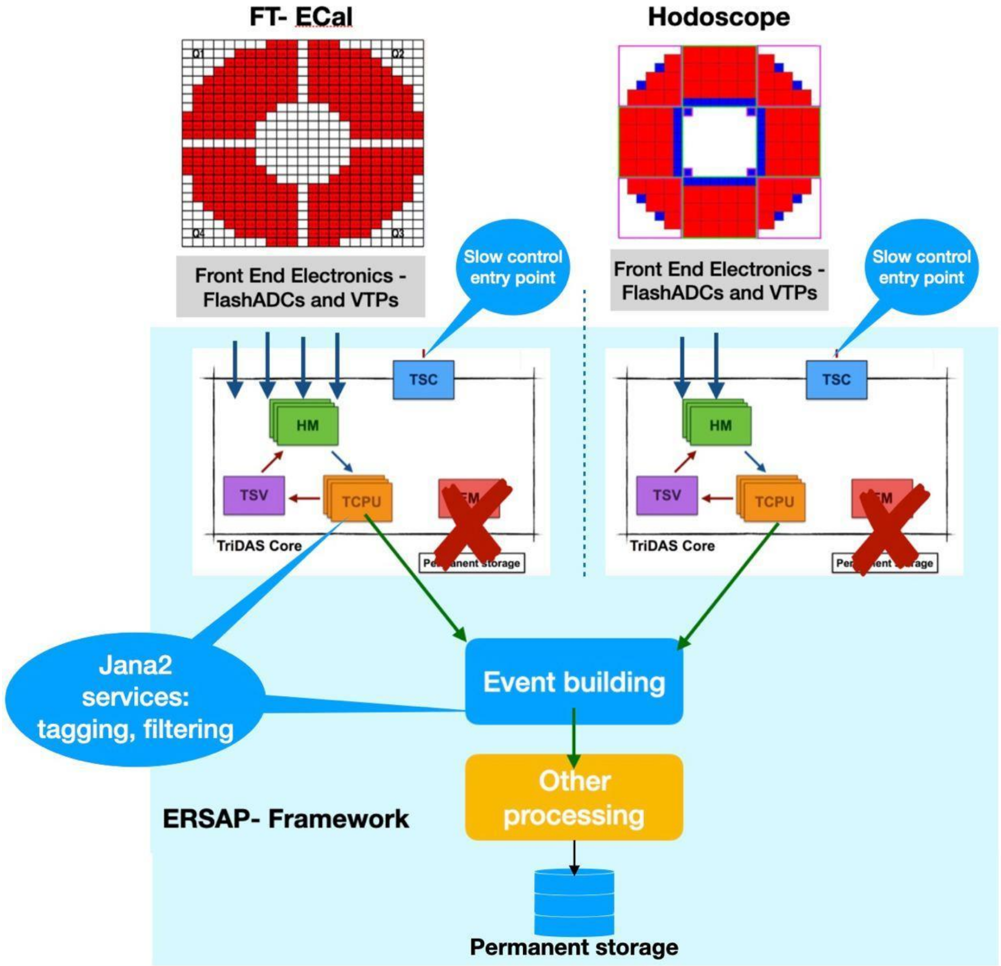
- Linux servers used:
  - 48 cores, 1GHz each, 64 GB RAM
  - 3 servers used for all modules
- HM instances: from 5 to 20
  - CPU consumption linear with the number of instances (500% – 1600%)
  - Memory occupancy constant (12-13 GB per run)
- TCPUs instances: 10 instances on 2 servers = 20 instances
  - 5 Time Slices at the same time on each instance
  - Trigger: **Jana2 plugin** (rudimental reconstructions and clustering)
  - CPU consumption: depending on the trigger algorithms (400% – 1600%)
  - Memory occupancy: 20-24 GB

From:

F. Ameli et al. "Streaming readout for next generation electron scattering experiments". arXiv: 2202.03085

TriDAS-ERSAP integration

- Next objective: TriDAS – ERSAP integration
  - ERSAP, or *Environment for Realtime Streaming Acquisition and Processing*: JLab micro-services architecture for data-stream acquisition and processing (see Vardan Gyurjyan's talk)
- Code improvements:
  - Update C++ version and code dependencies
  - General review of each component
- TriDAS reviews for ERSAP integration:
  - Remove CODA translator and manage directly the VTP data format
  - The HMs became servers
  - Use multiple instances of TriDAS
  - Remove the EM to directly send the events to ERSAP



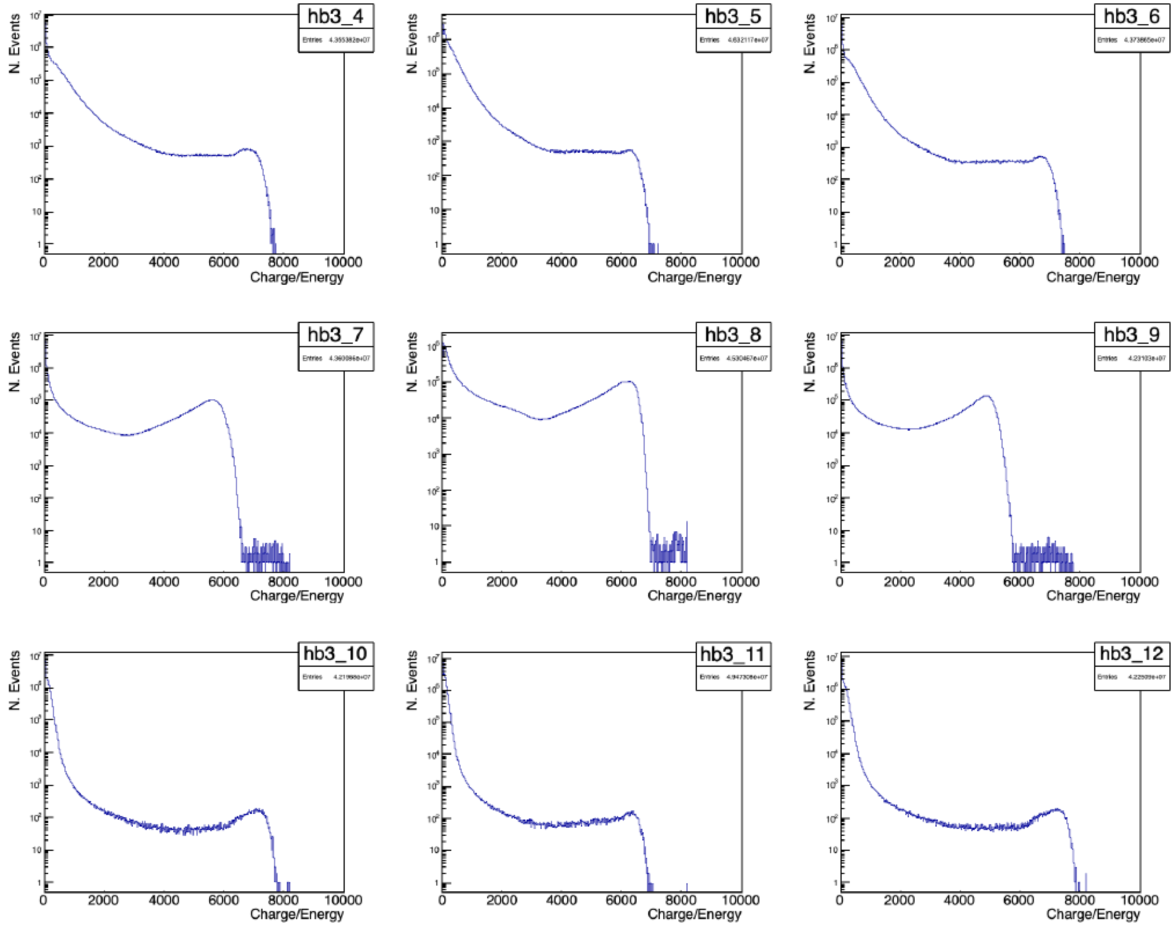
Test in 2022

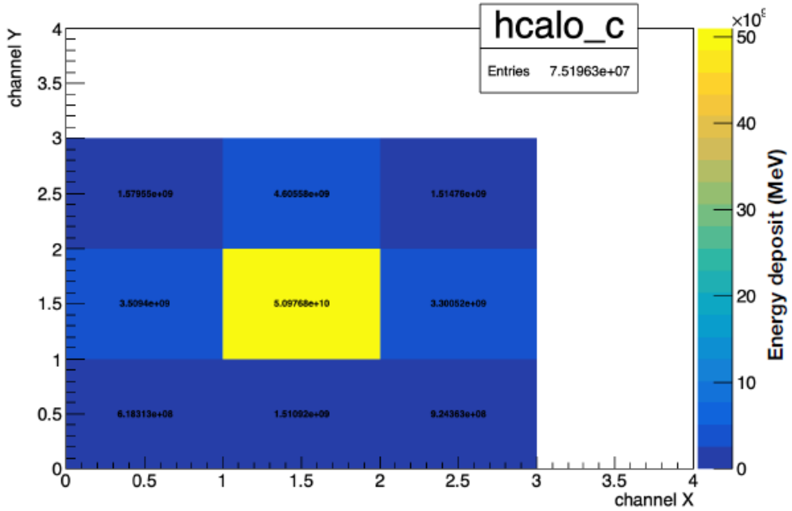
- Tests divided in 2 phases:
  1. Validate the new TriDAS implementation without ERSAP
     - Use the VTP data format (instead of CODA translator) and the Event Manager
     - Three tests done:
       a. **Run TriDAS with a two channel pulser to validate the VTP data format**
       b. Run TriDAS with a 3x3 calorimeter to validate the new implementation in a limited environment
       c. Run TriDAS with the Forward Tagger sub-detectors
  2. Validate the TriDAS-ERSAP integration
     - Use TriDAS as ERSAP microservice (without CODA translator and EM)
     - One test on the Forward Tagger Calorimeter with one TriDAS instance

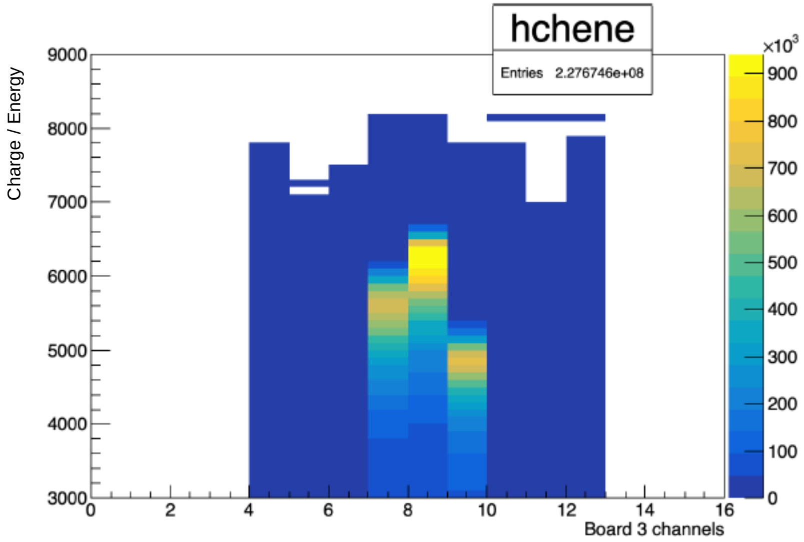
Test in 2022

- Tests divided in 2 phases:
  1. Validate the new TriDAS implementation without ERSAP
     - Use the VTP data format (instead of CODA translator) and the Event Manager
     - Three tests done:
       a. Run TriDAS with a two channel pulser to validate the VTP data format
       b. Run TriDAS with a 3x3 calorimeter to validate the new implementation in a limited environment
       c. Run TriDAS with the Forward Tagger sub-detectors
  2. Validate the TriDAS-ERSAP integration
     - Use TriDAS as ERSAP microservice (without CODA translator and EM)
     - One test on the Forward Tagger Calorimeter with one TriDAS instance

Results: TriDAS & 3x3 calorimeter







Test in 2022

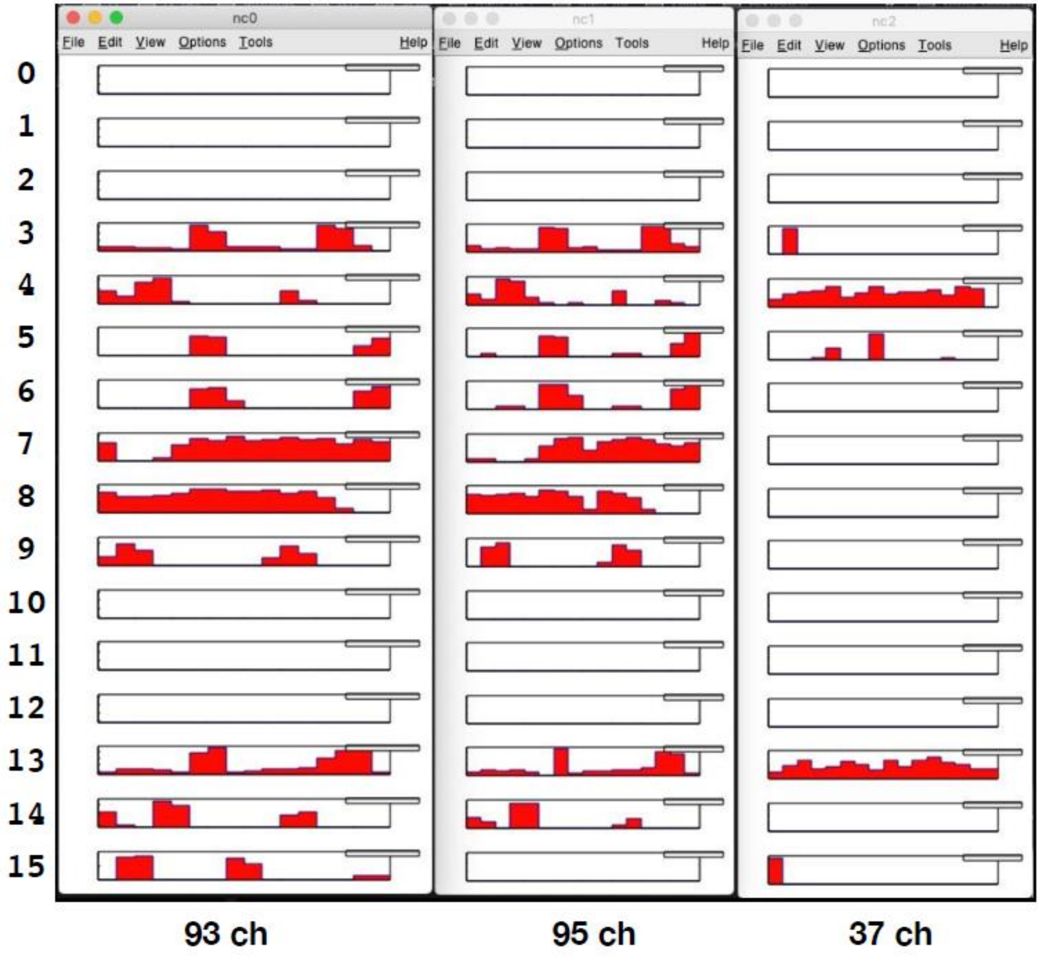
- Tests divided in 2 phases:

  1. Validate the new TriDAS implementation without ERSAP
     - Use the VTP data format (instead of CODA translator) and the Event Manager
     - Three tests done:
       a. Run TriDAS with a two channel pulser to validate the VTP data format
       b. Run TriDAS with a 3x3 calorimeter to validate the new implementation in a limited environment
       c. Run TriDAS with the Forward Tagger sub-detectors

  2. Validate the TriDAS-ERSAP integration
     - Use TriDAS as ERSAP microservice (without CODA translator and EM)
     - One test on the Forward Tagger Calorimeter with one TriDAS instance

Results: TriDAS & Forward Tagger

- One TriDAS instance on one Linux server:
  - 6 HMs
  - 3 TCPUs (computed 6 parallel TS per TCPU)
  - 1 EM
- Performance:
  - 225 channels detected
  - 5 cores used
  - Memory occupancy constant

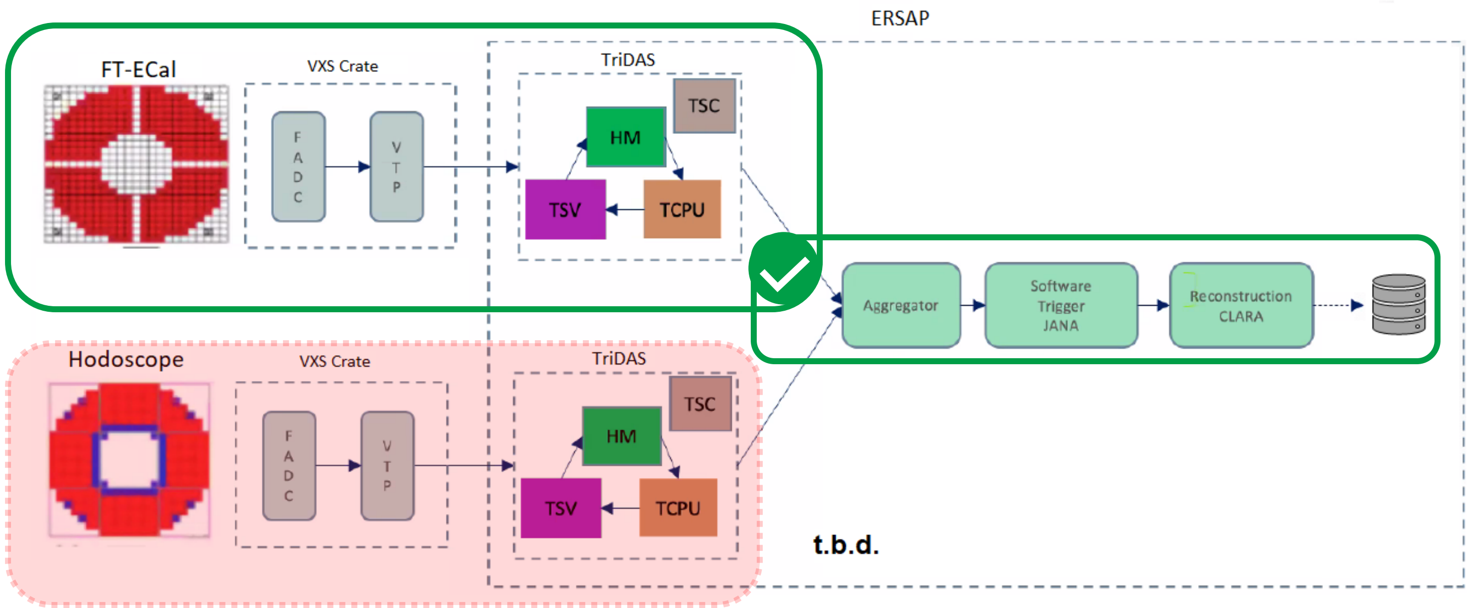


Test in 2022

- Tests divided in 2 phases:
  1. Validate the new TriDAS implementation without ERSAP
     - Use the VTP data format (instead of CODA translator) and the Event Manager
     - Three tests done:
       a. Run TriDAS with a two channel pulser to validate the VTP data format
       b. Run TriDAS with a 3x3 calorimeter to validate the new implementation in a limited environment
       c. Run TriDAS with the Forward Tagger sub-detectors
  2. **Validate the TriDAS-ERSAP integration**
     - **Use TriDAS as ERSAP microservice (without CODA translator and EM)**
     - **One test on the Forward Tagger Calorimeter with one TriDAS instance**

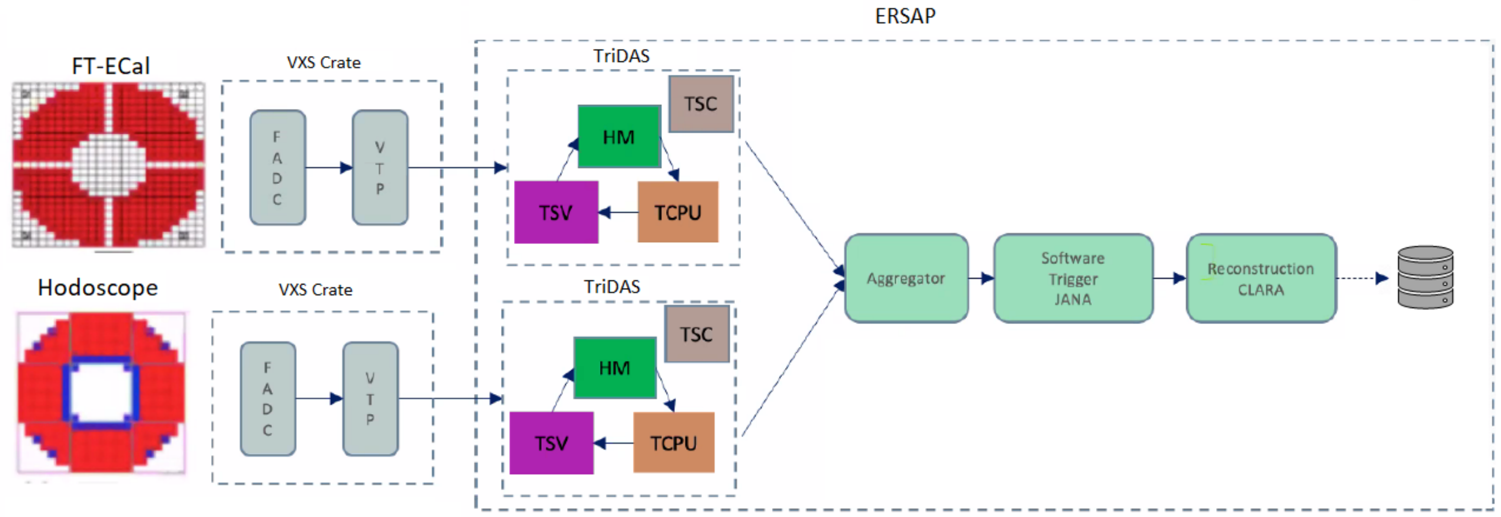
Results: TriDAS & ERSAP

- One TriDAS instance on one Linux server:
  - 6 HM instances
  - 3 TCPU (6 parallel TS per TCPU)
  - NO EM
- TriDAS – ERSAP interface:
  - ZeroMQ library is used
  - The Event-TAG is correctly identified
  - Some issues detected: event payload interpretation to be further checked



Conclusion

- The TriDAS-ERSAP integration is in progress
  - Communication interface under review
- Used also a GEMC-based simulation code to reproduce the CLAS12 data streams
  - New tests will be performed to activate all DAQ components
  - TriDAS shows expected results, and it could be used as cross check validator
  - TriDAS performance and results need to be compared with those obtained from the TriDAS-ERSAP integration
- Goal: use multiple instances of TriDAS as ERSAP microservices



QUESTIONS?