Workshop IX on Streaming Readout

TriDAS

Laura Cappelli – INFN-CNAF

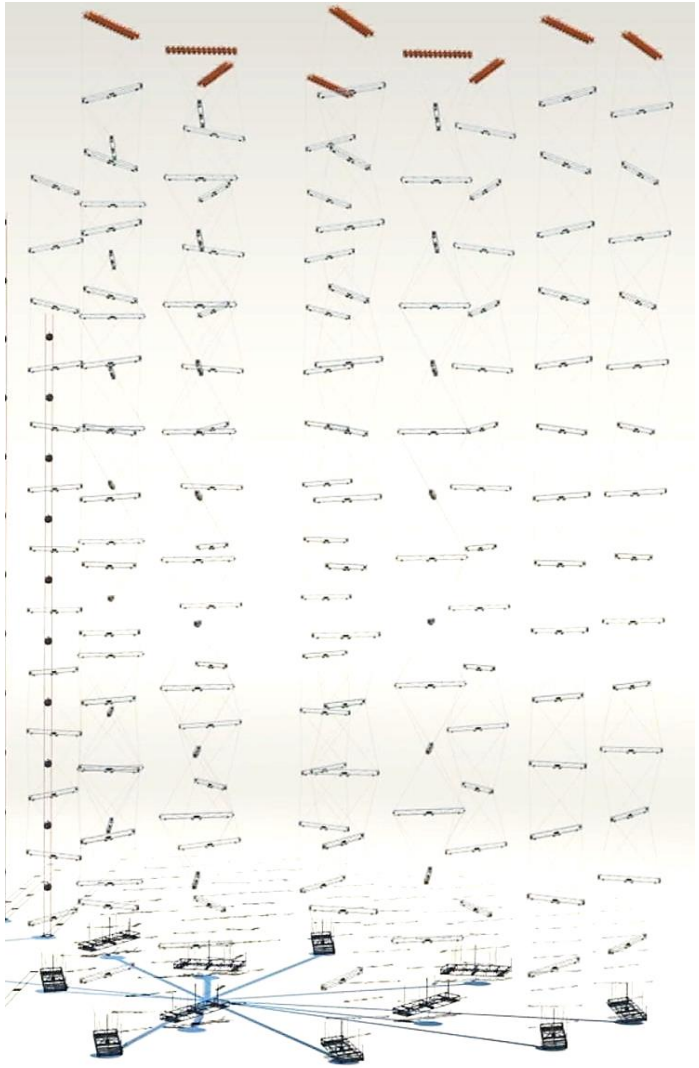
Co-authors: *Tommaso* Chiarusi[1], *Francesco* Giacomini[2], C*armelo* Pellegrino[2]

[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

December 10th, 2021

The birth of TriDAS

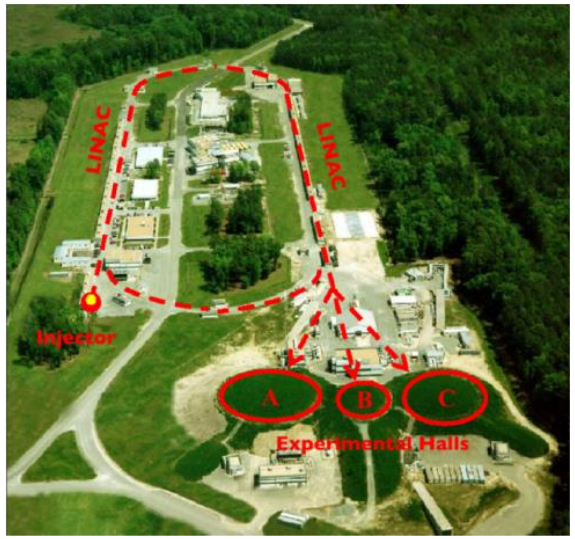


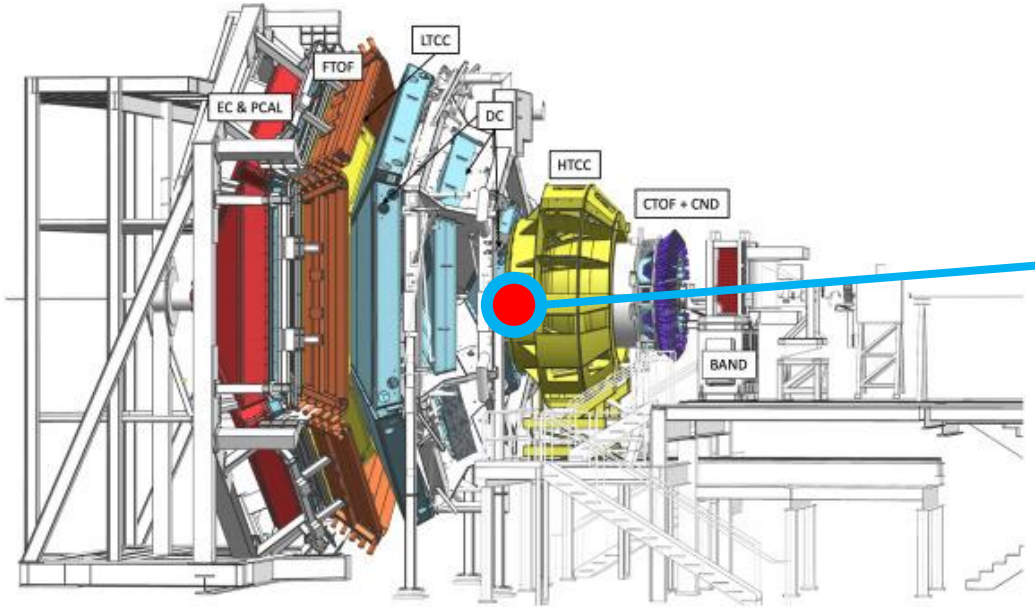
- Designed for streaming read-out of Astro-particle Physics events
  - NEMO project
  - KM3Net-ITA (Towers) project
- ***All-data-to-shore*** approach: the software at the shore station
  - Reads the data streams
  - Reconstructs the events
  - Filters the data to collect only the interesting ones

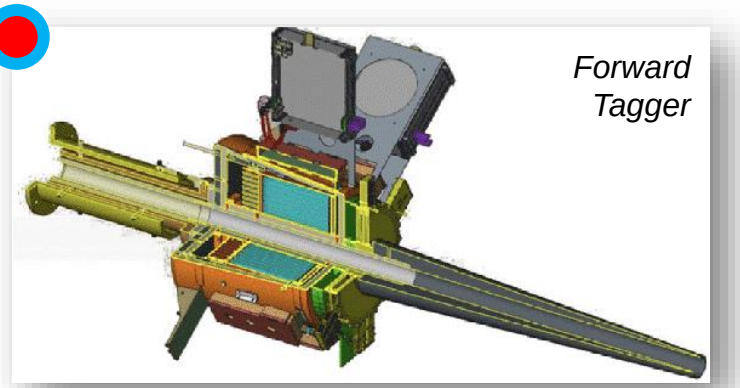
The KM3Net Towers

TriDAS @ JLab

- TriDAS was appended to the readout system of CLAS12:
  - It provided a proper customization of the front-end electronics firmware to cope with the **triggerless** approach
  - Collect data from Forward Tagger sub-detector
  - In summer 2020 that software prototype system was successfully tested



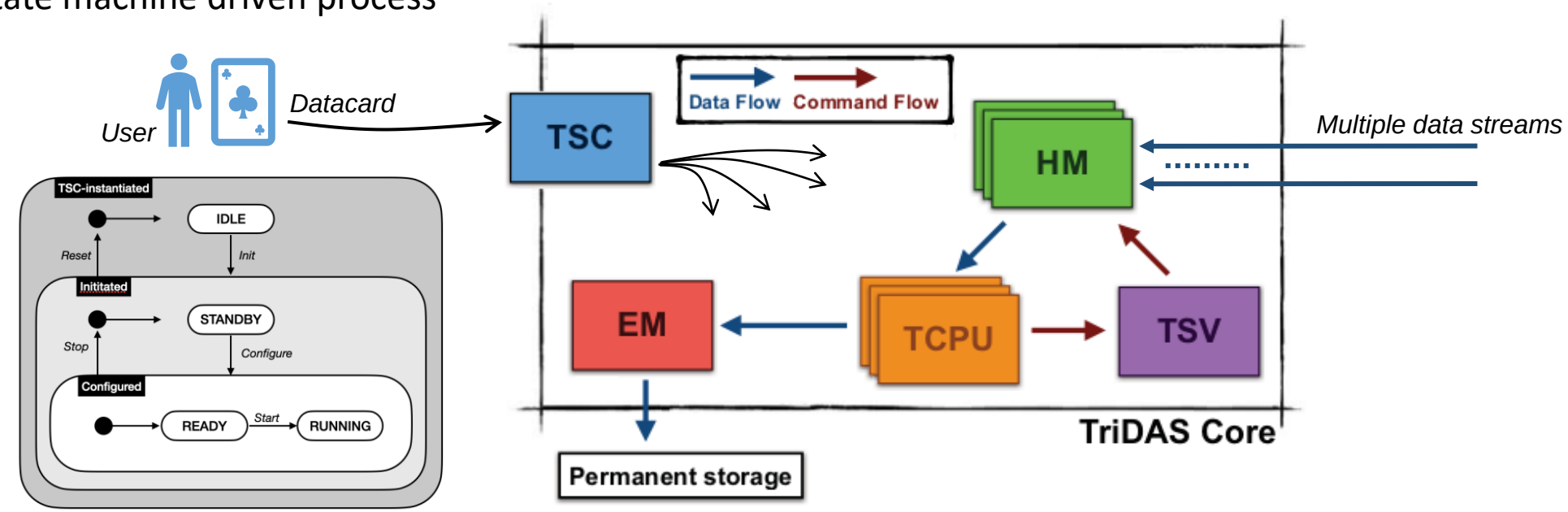




The TriDAS framework

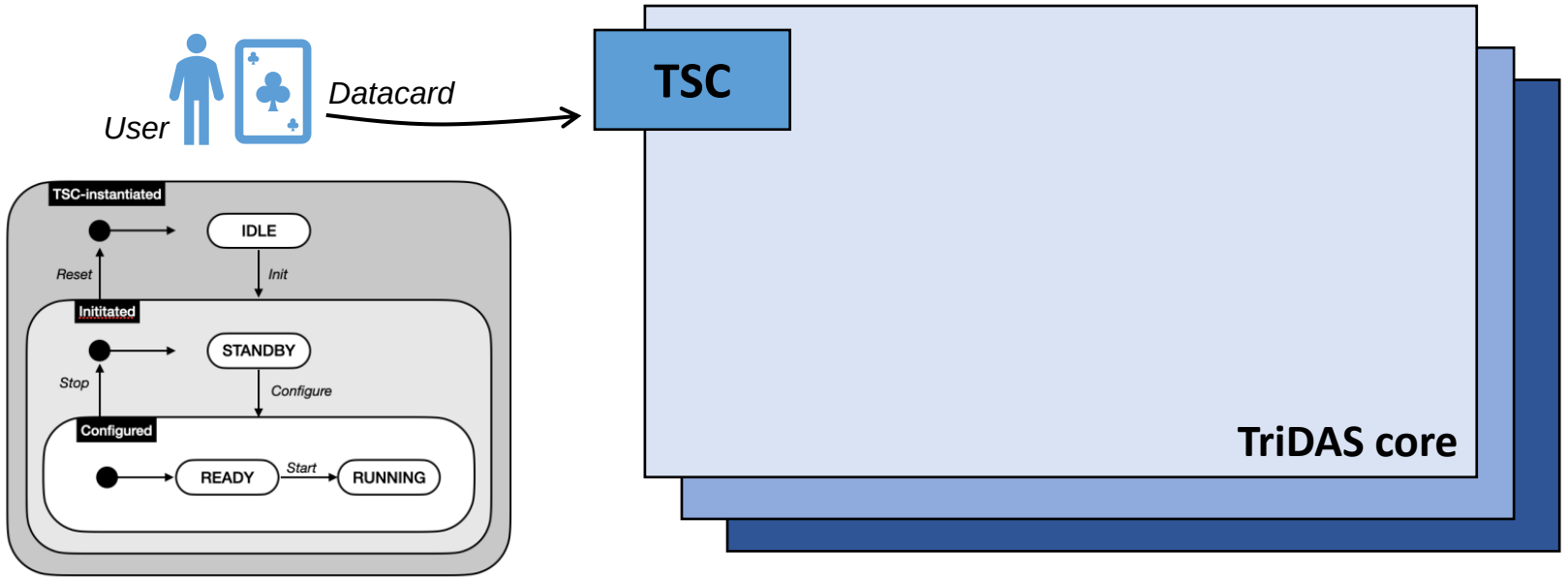
- TriDAS characteristics:
  - C++17 multithreaded software framework
  - Dependencies: CMake, ZeroMQ, Boost
  - Flexible design:
    - Configurable via datacard (e.g. trigger algorithms)
    - Data format
  - State machine driven process

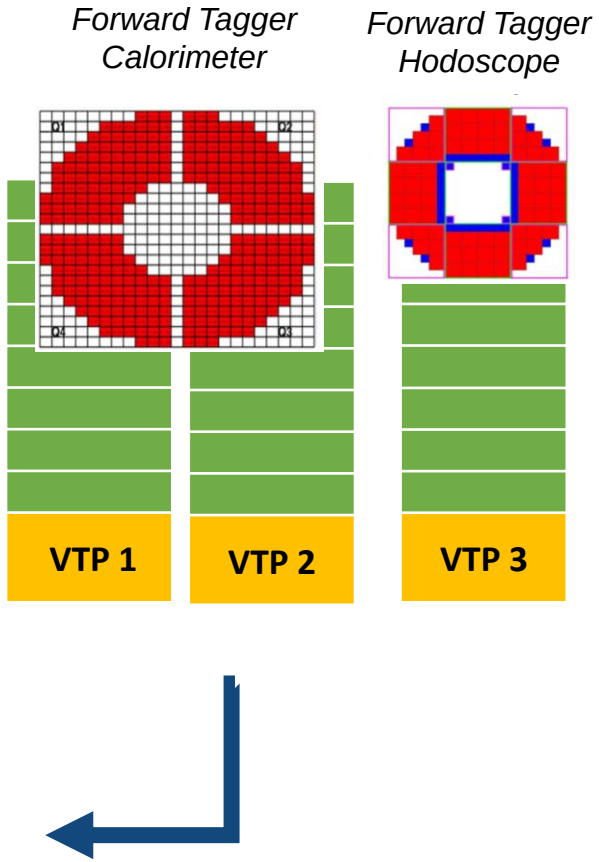
- Composed by 5 modules:
  - HM (*Hit Manager*)
  - TCPU (*Trigger CPU*)
  - TSV (*TriDAS SuperVisor*)
  - EM (*Event Manager*)
  - TSC (*TriDAS System Controller*)



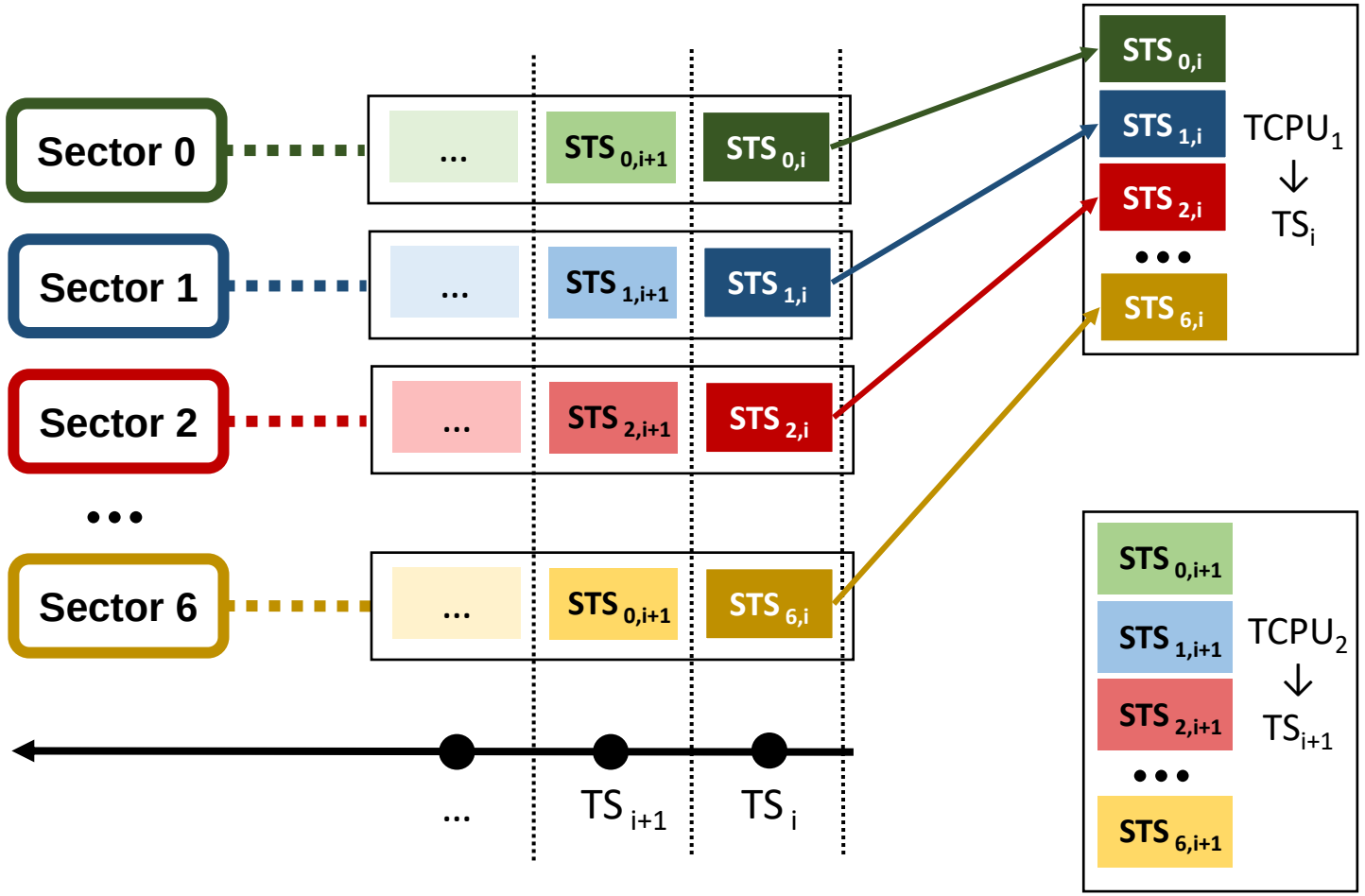
Input streams

- TriDAS input:
  - 2 streams from 3 VXS Trigger Processors (VTPs) = 6 streams
    - Throughput: **4 GBps** per stream (total: 24 GBps)
  - 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)



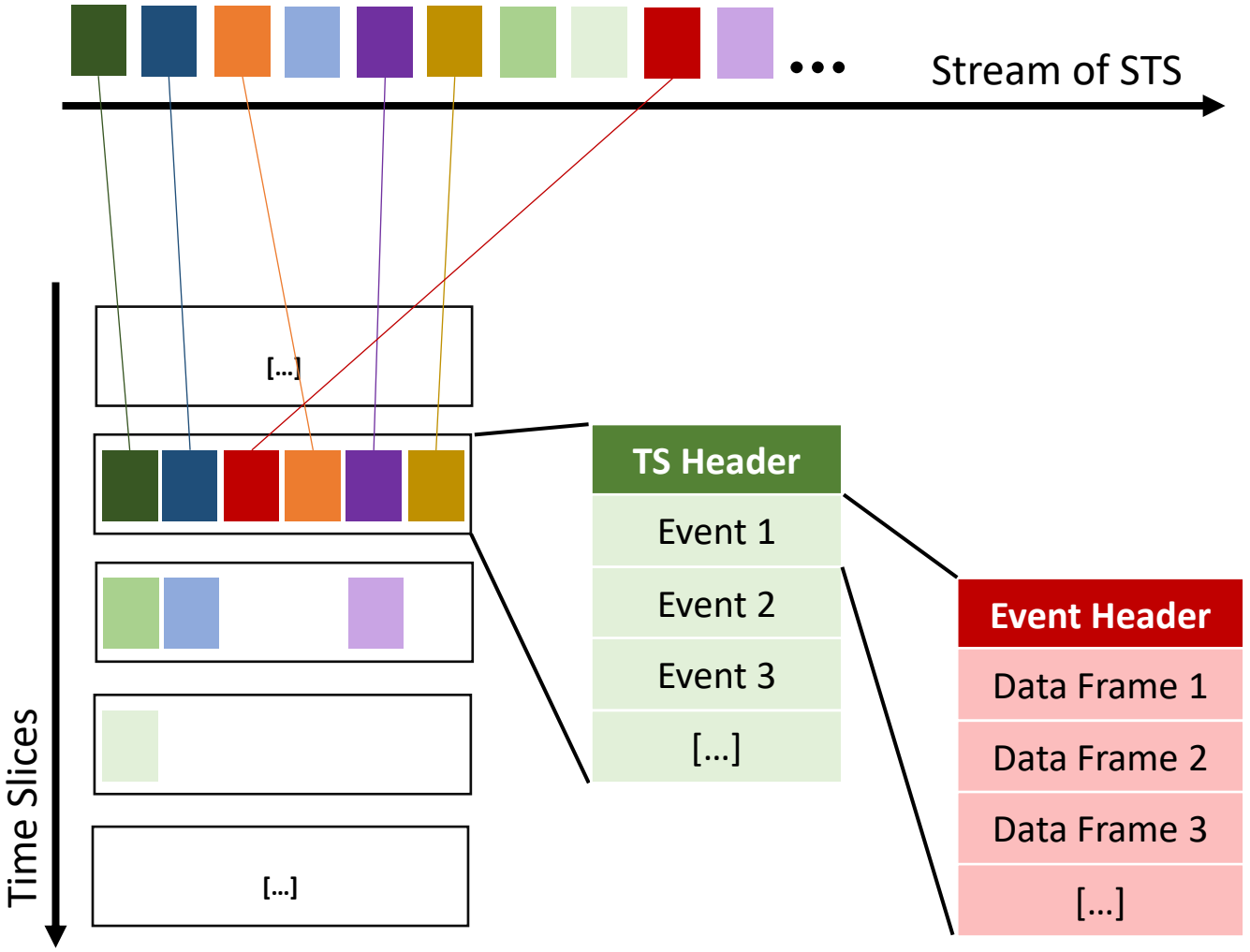


Data Flow: the HM aggregation



- 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 (50 ms) called **Time Slice (TS)**
  - Sends the STSs to a TCPU according to the token received from the TSV
- A TCPU receive all the STSs of a TS

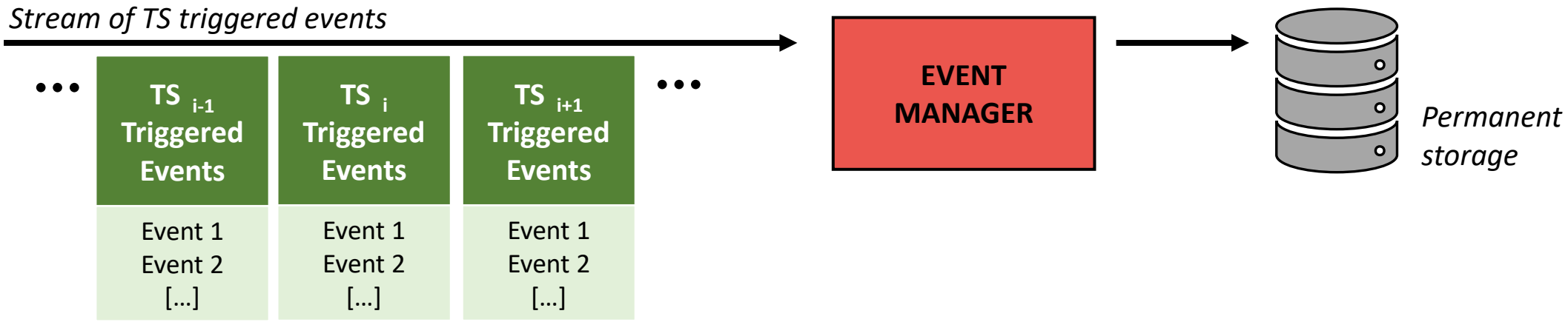
Data Flow: event building & trigger



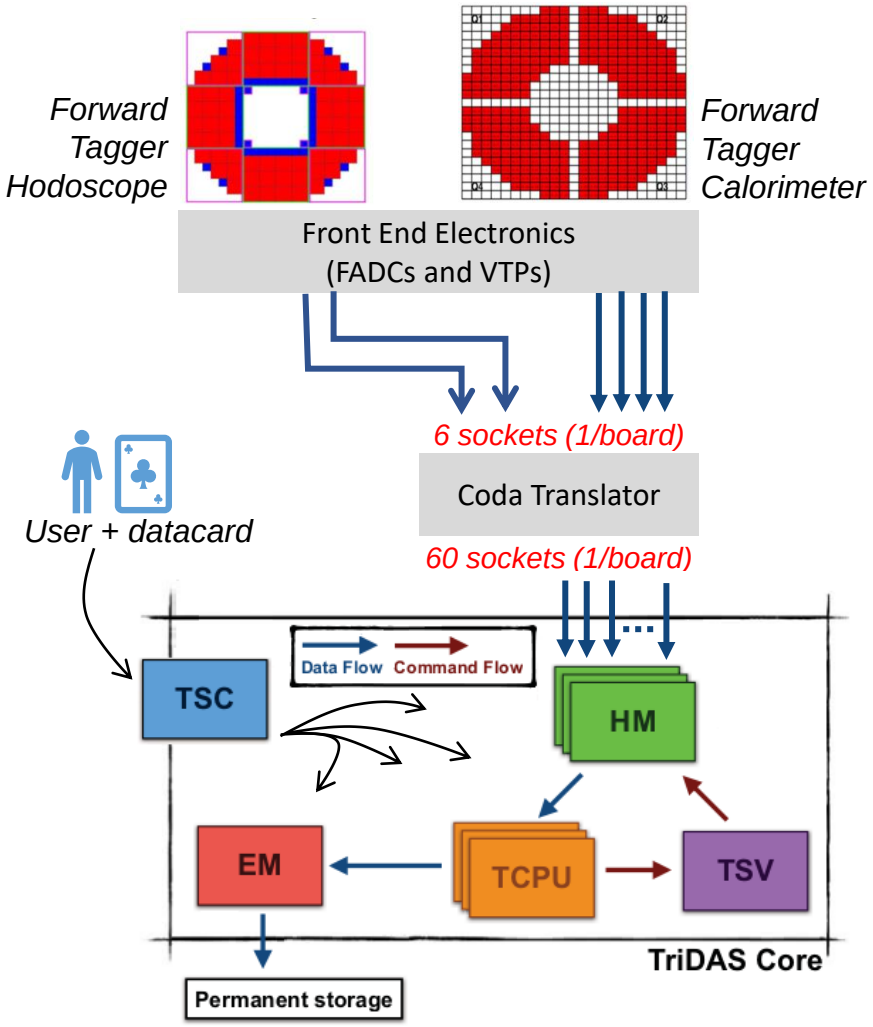
- The TCPU:
  - Receives a stream of STSs
    - The STSs temporal order isn't respected
    - There are many threads, each one arrange the data of a TS
  - Reconstructs events per time window
  - Applies one or more trigger algorithms
    - **External plugin** selected in the datacard
    - In the test at Jlab in summer 2020 the Jana2 framework was used
- 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:
  - Orders by time the TSs
  - Writes the TSs into some post trigger files
- All the useful event information are stored for further analysis



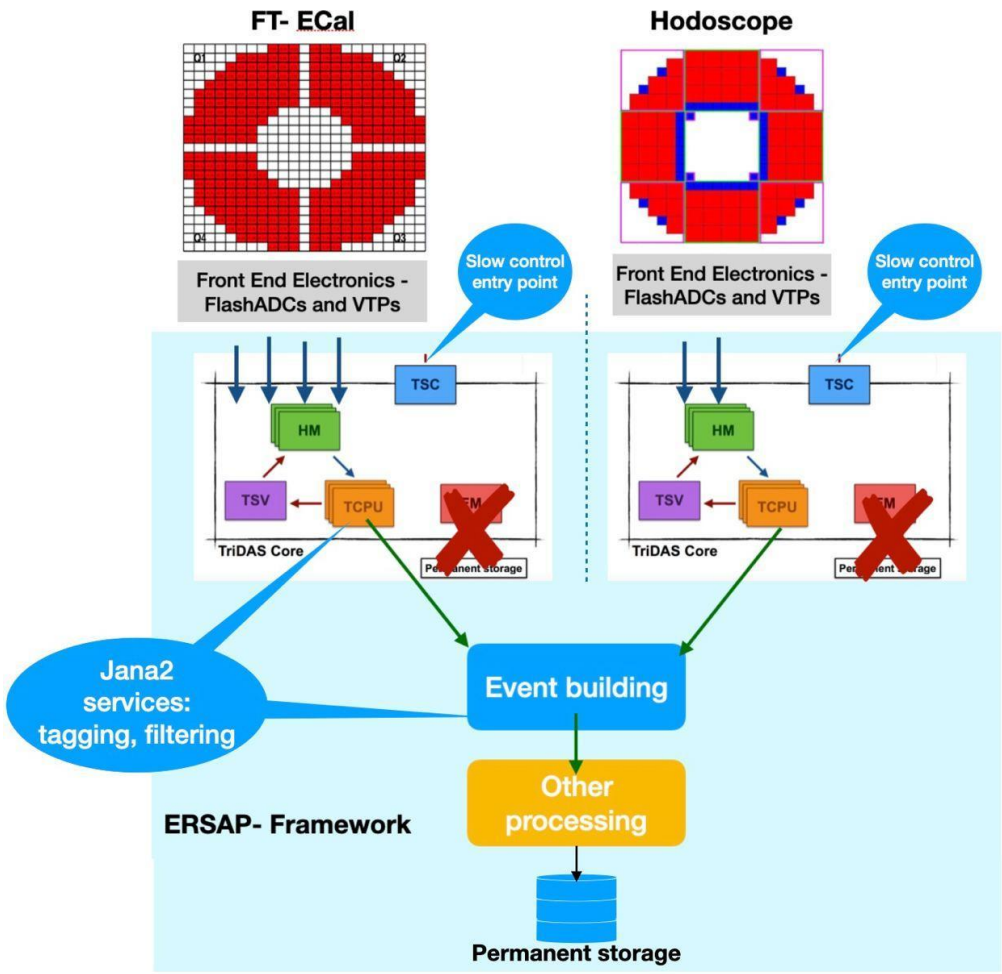
Test in summer 2020



- Test setup:
  - System tested on the Forward Tagger sub-detector (768 ch)
  - Between the front-end electronics and TriDAS there was the CODA translator layer
  - Frontend electronics calibration: throughput set from few tens of MBps to 100 MBps
  - 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

TriDAS in 2021 and future work

- An intensive review is ongoing:
  - The TriDAS prototype tested in 2020 is under upgrade
    - Update C++ version and the code dependencies
    - General review of each component
    - Remove CODA translator and manage a new data format
  - Purpose: use an improved TriDAS version for the next CLAS12 runs
  - **Test ongoing since fall 2021**
- Next objective: TriDAS – ERSAP integration
  - ERSAP (*Environment for Realtime Streaming Acquisition and Processing*): JLab micro-services architecture for data-stream acquisition and processing
  - see Vardan Gyurjyan's talk
- TriDAS code available at https://baltig.infn.it/tridas/tridas-core



QUESTIONS?

✉ laura.cappelli@cnaf.infn.it

BACKUP SLIDES

Command Flow: the TSV

- The TSV tokens mechanism:
  - A TCPU has as a set of tokens
  - When a TCPU thread is free, it sends to the TSV its token
  - The TSV associates that token to a new TS and forwards the information to the HMs
  - The HMs learn the association TS – TCPU
  - When an STS of that TS arrives to a HM, it sends that STS to the corresponding TCPU