CHEP2024, Krakow, Poland

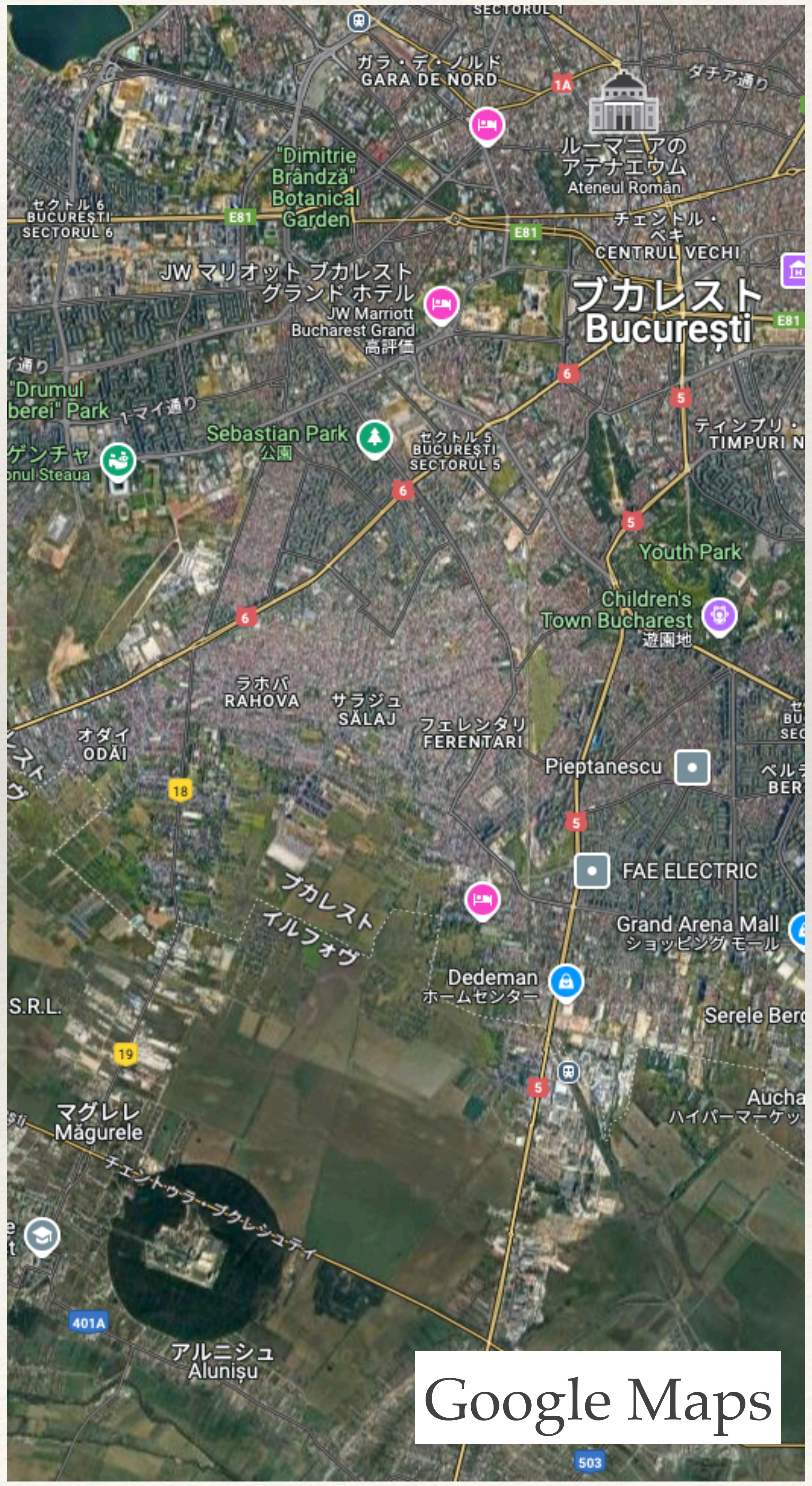
Implementation and development of a DAQ system DELILA at ELI-NP

S. Aogaki
Extreme Light Infrastructure-Nuclear Physics (ELI-NP)

Contents

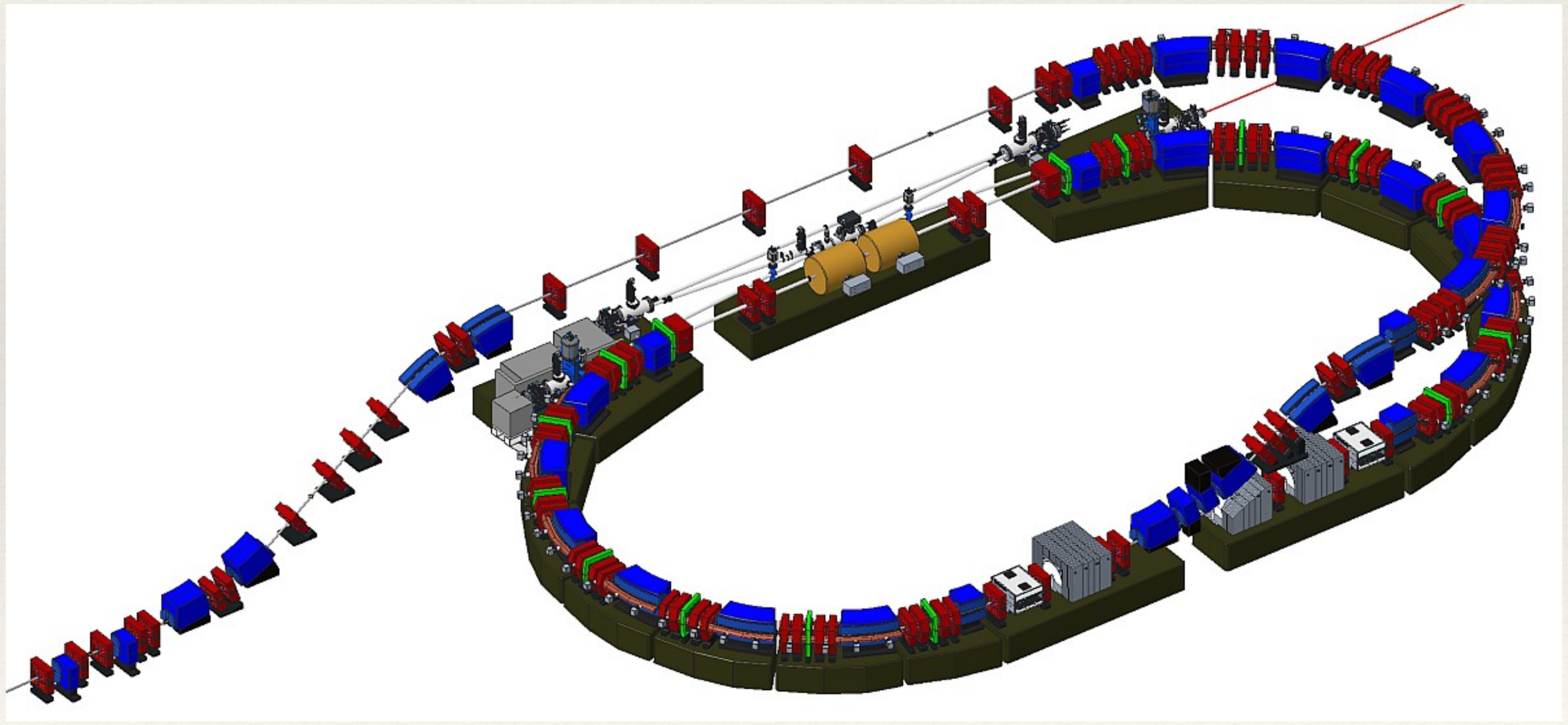
ELI-NP

- **E**xtreme **L**ight **I**nfrastructure - **N**uclear **P**hysics
  - Laser and γ-ray beams facility for Nuclear Physics
    - Laser beamlines are operated
    - γ will be operated soon
  - Magurele (near Bucharest), Romania
- Magurele is a kind of researchers' town, science town
- Also ELI-ALPS (Hungary) and ELI-Beamlines (Czech)



VEGA

- **V**ariable **E**nergy **Ga**mma System at ELI-NP
- Up to 19.5 MeV γ-ray beam
- Total photon flux $1.0 \times 10^{11}/\mathrm{s}$
- Under construction
- Full operation 2026



Requirements for DAQ

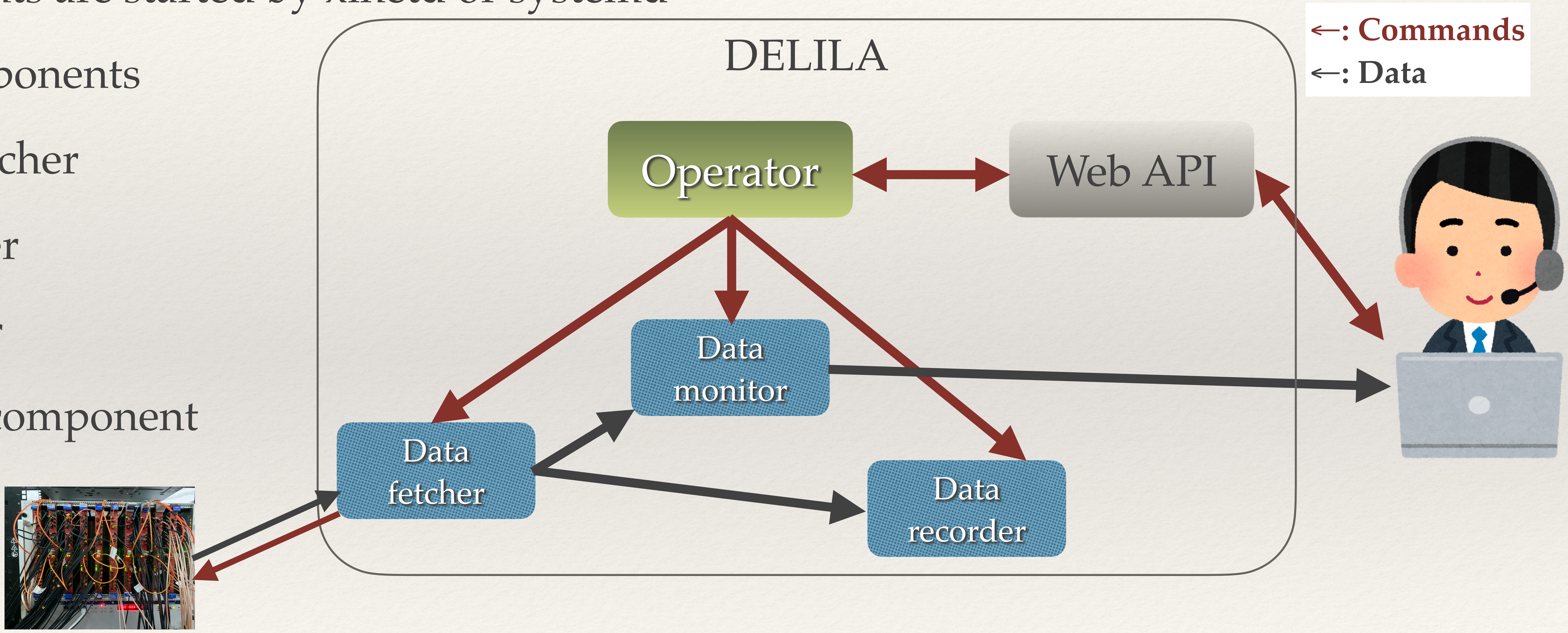
- Support some electronic modules
    - CAEN digitizers (1725, 1730, 1740), Mesytec ADC TDC
    - We can add any electronics if we can fetch data by C/C++
        - e.g. temperature sensor
- Open source
    - We do not want to wait for the update when we need
- Network transparency
    - Controlling some computers from a remote
- Using the same clock source at different electronics

DELILA

- **D**igital **ELI**-NP **L**ist-mode **A**cquisition system
- Using DAQ-Middleware
  - https://daqmw.kek.jp/
  - Component base system
  - Easy porting
  - Based on a robotics technology (RT) !! NOT Real Time
- Using ROOT library
  - https://root.cern/
  - Monitoring and Recording
  - JSROOT is awesome! But no nice fitting functions now
- Almost all parts are written in C++. The web interface is TypeScript with Angular

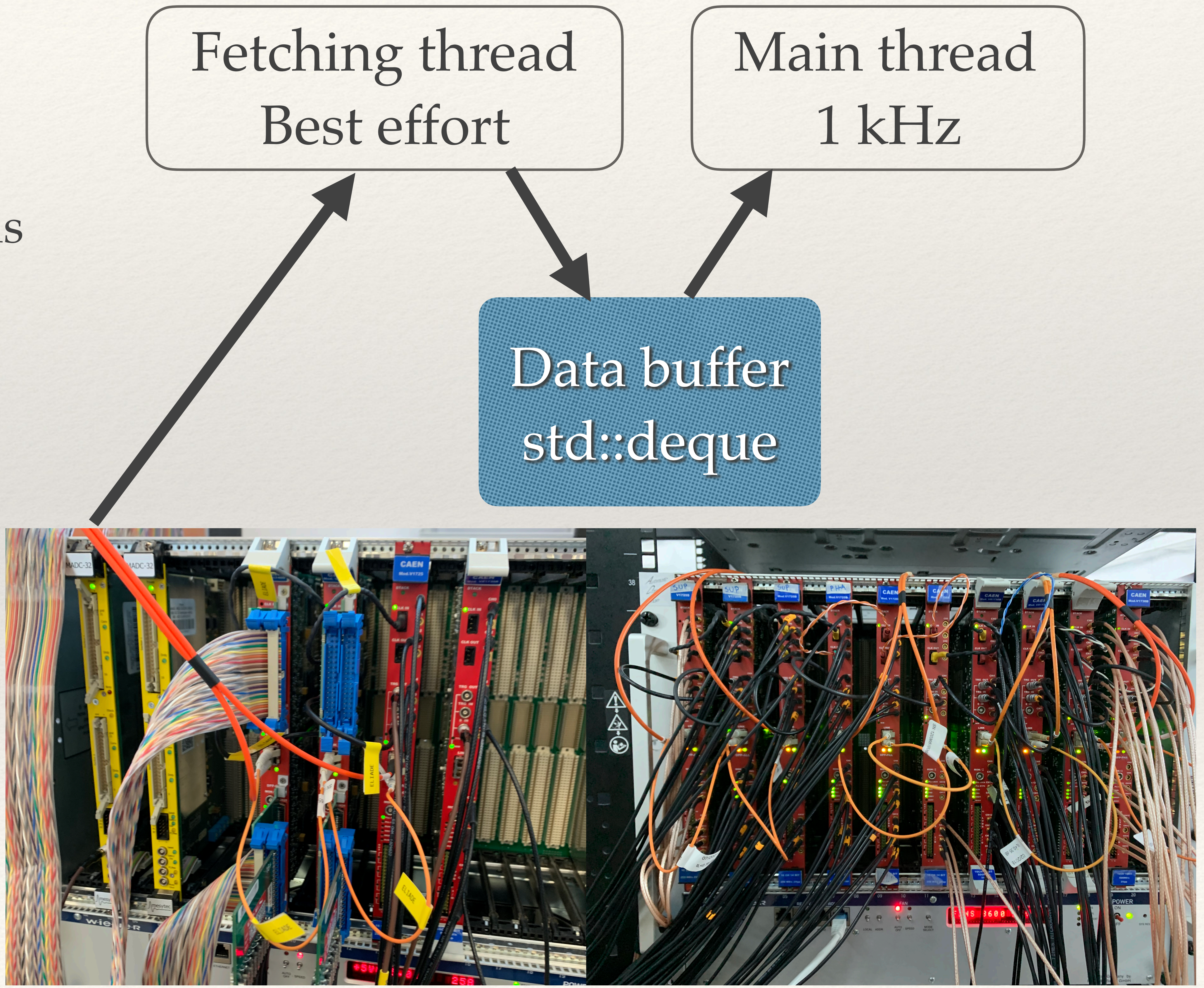
Components

- Components are started by xinetd or systemd
- Basic components
  - Data fetcher
  - Recorder
  - Monitor
- Operator component



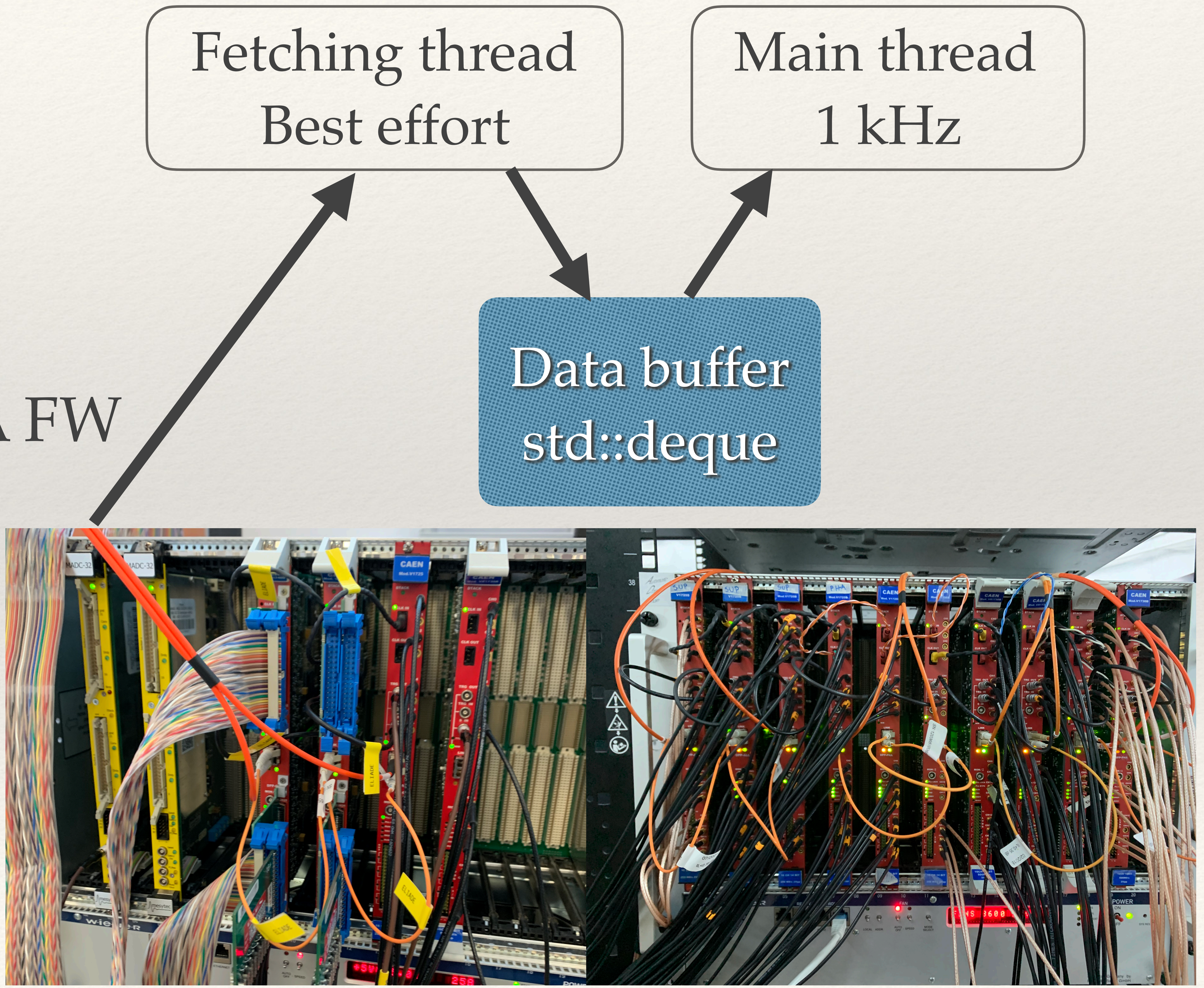
Data fetcher

- Two threads
  - Main thread
    - Called by the operator component every 1ms
    - Check whether data is available or not
      - Sending the data downstream
  - Data fetching thread
    - Communicating to electronics
    - Fetching and packing data
    - 1ms sleep or no sleep



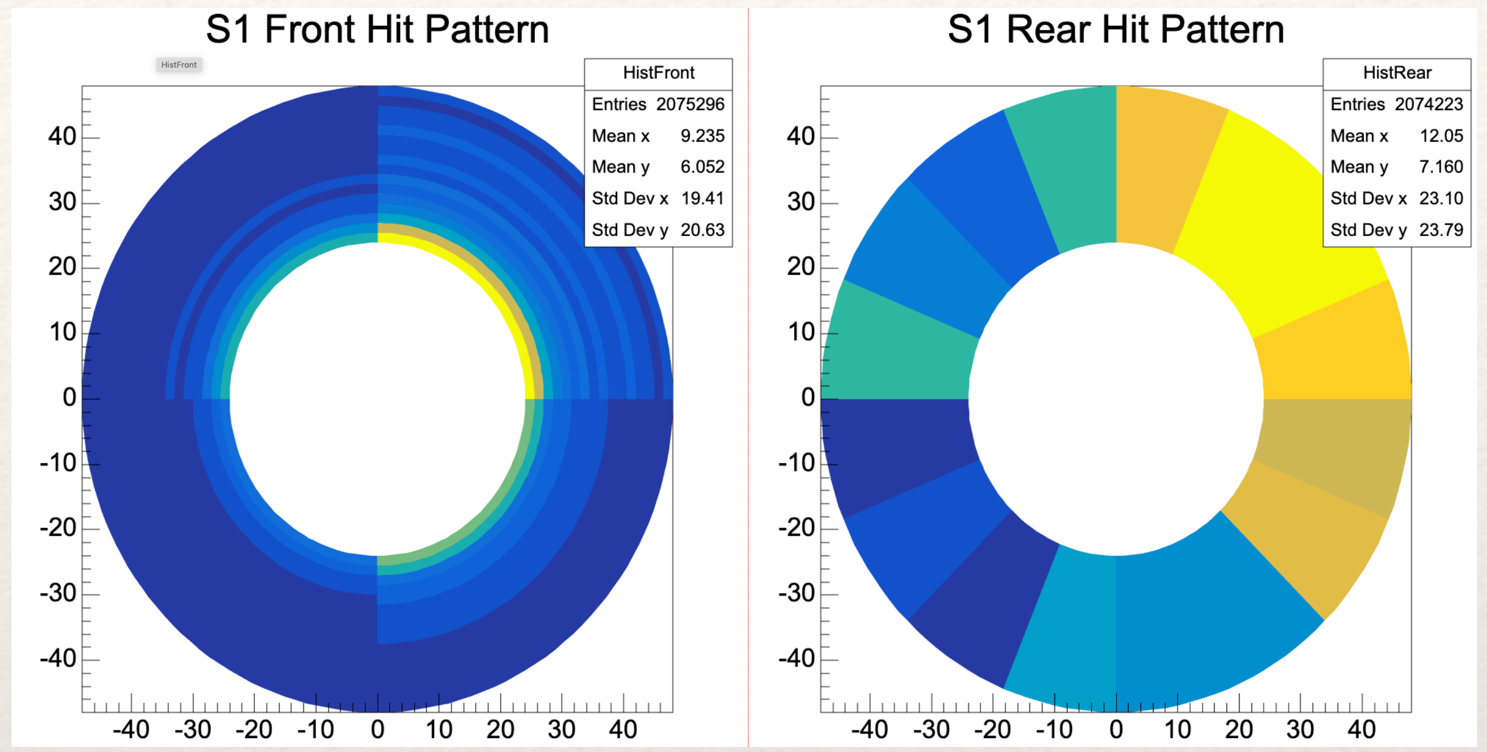
Data fetcher

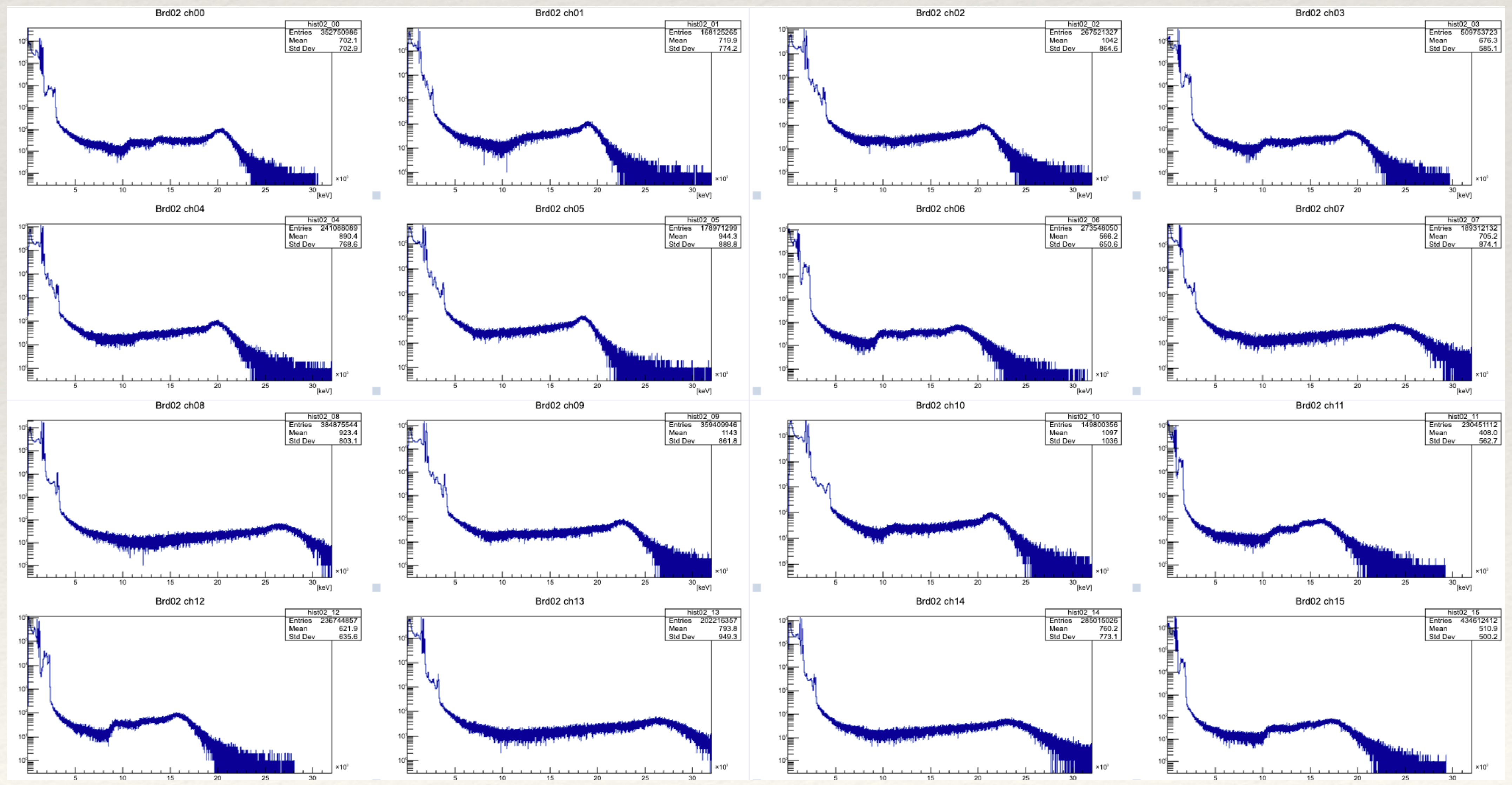
- Support some digitizers
- CAEN digitizers
  - 1740 DPP-QDC FW
  - 1730, 1725 DPP-PSD and DPP-PHA FW
  - Variations of above
- Mesytec
  - MADC-32



Data monitor

- Using ROOT THttpServer
  - Sometimes, using JSROOT for special layout
- Making thread pool and process data
  - TH2Poly::Fill is very heavy
  - A single thread can not process more than 1 M Hz data, need a trick. TH1 is thread-safe
- Calculating the counting rate and uploading to a DB for Grafana





Data recorder

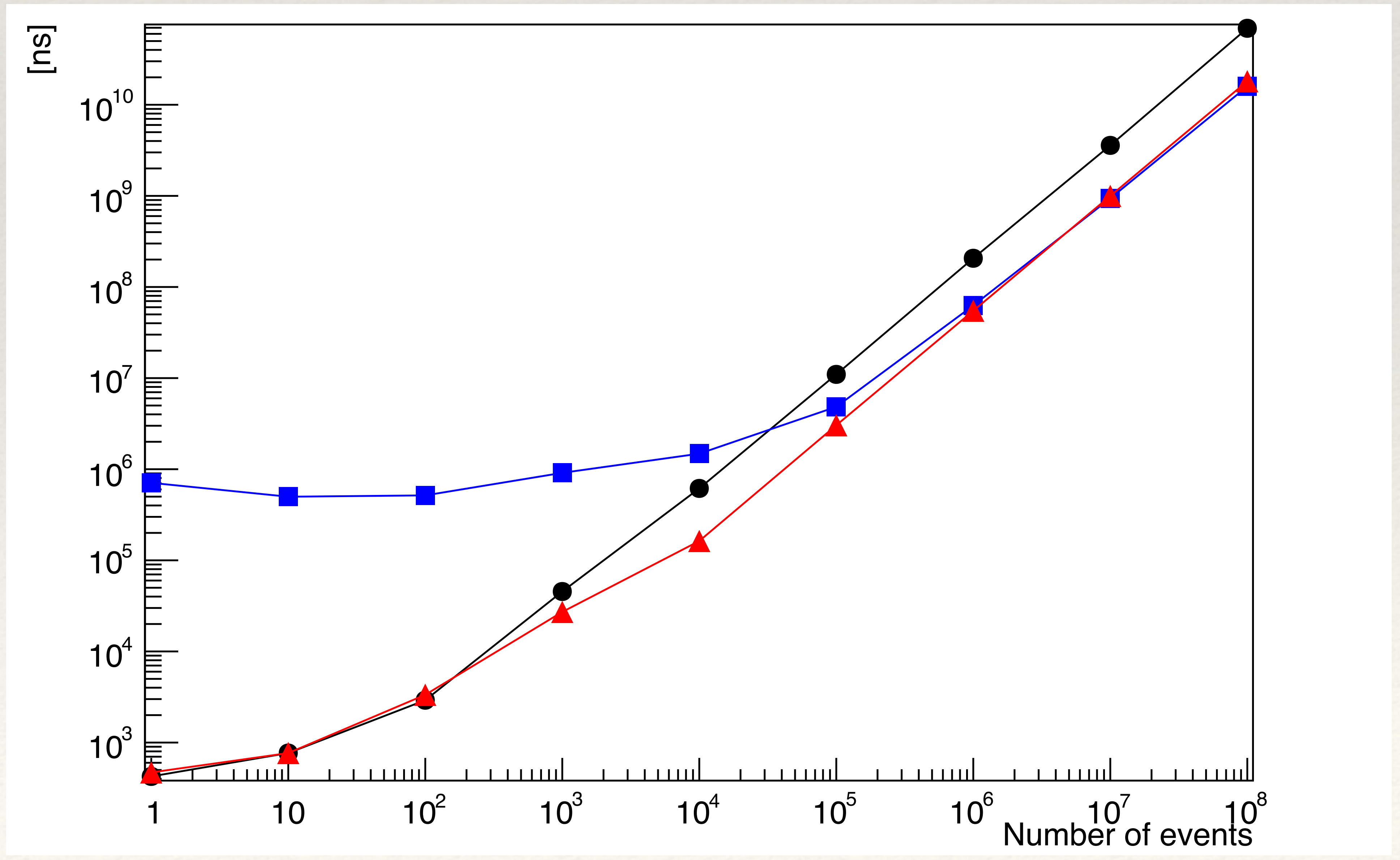
- Writing data into ROOT format
  - Sometimes, this is a bottleneck
    - If using compression
- Sorting by timestamp
  - std::vector
  - TBB is slow with GCC 14.2.1, Core i7-11700

1GB data writing	With Compression	Without Compression	Simple binary
Time duration / Speed SATA SSD	< 12 s < 90 MB / s	< 2 s < 500 MB / s	< 2 s < 500 MB / s
Time duration / Speed MV2 SSD	< 11 s < 90 MB / s	< 0.7 s < 1500 MB / s	< 0.7 s < 1500 MB / s

Single thread

16 threads (OMP): __gnu_parallel::sort

16 threads (TBB): std::sort(std::execution::par



Other software modules

- Web API
- Event Builder

Web API and controller UI

- Web API server
  - Oat++
  - https://oatpp.io/
  - Controlling the data acquisition process
  - Recording run information
- Controller
  - Communicating with the API server
  - Not so rich now

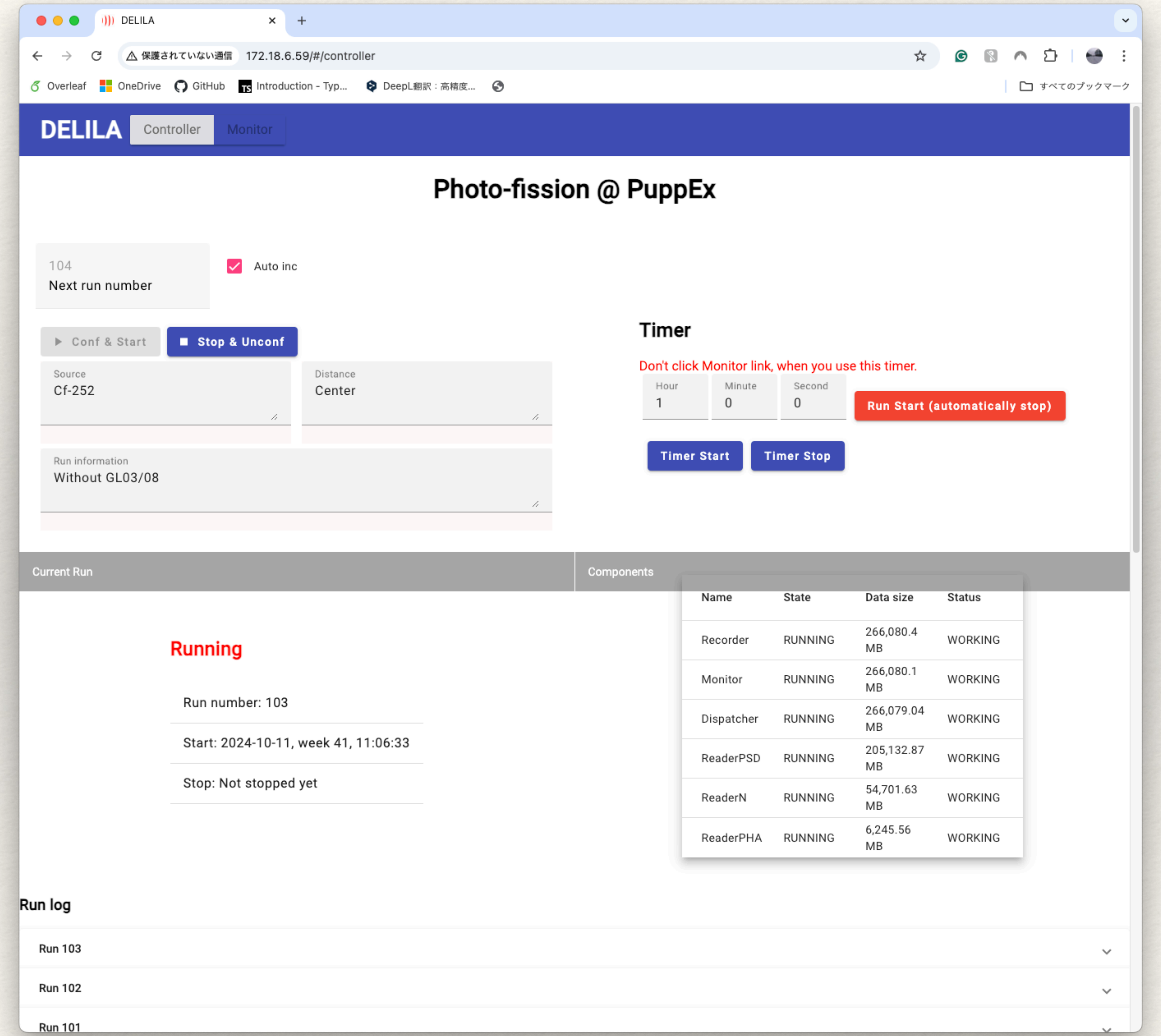


Photo-fission @ PuppEx

104
Next run number

[✓ Auto inc]

Source: Cf-252

Distance: Center

Run information: Without GL03/08

Timer

Don't click Monitor link, when you use this timer.

Hour	Minute	Second
1	0	0

Running

Run number: 103

Start: 2024-10-11, week 41, 11:06:33

Stop: Not stopped yet

Name	State	Data size	Status
Recorder	RUNNING	266,080.4 MB	WORKING
Monitor	RUNNING	266,080.1 MB	WORKING
Dispatcher	RUNNING	266,079.04 MB	WORKING
ReaderPSD	RUNNING	205,132.87 MB	WORKING
ReaderN	RUNNING	54,701.63 MB	WORKING
ReaderPHA	RUNNING	6,245.56 MB	WORKING

Run log

Run 103

Run 102

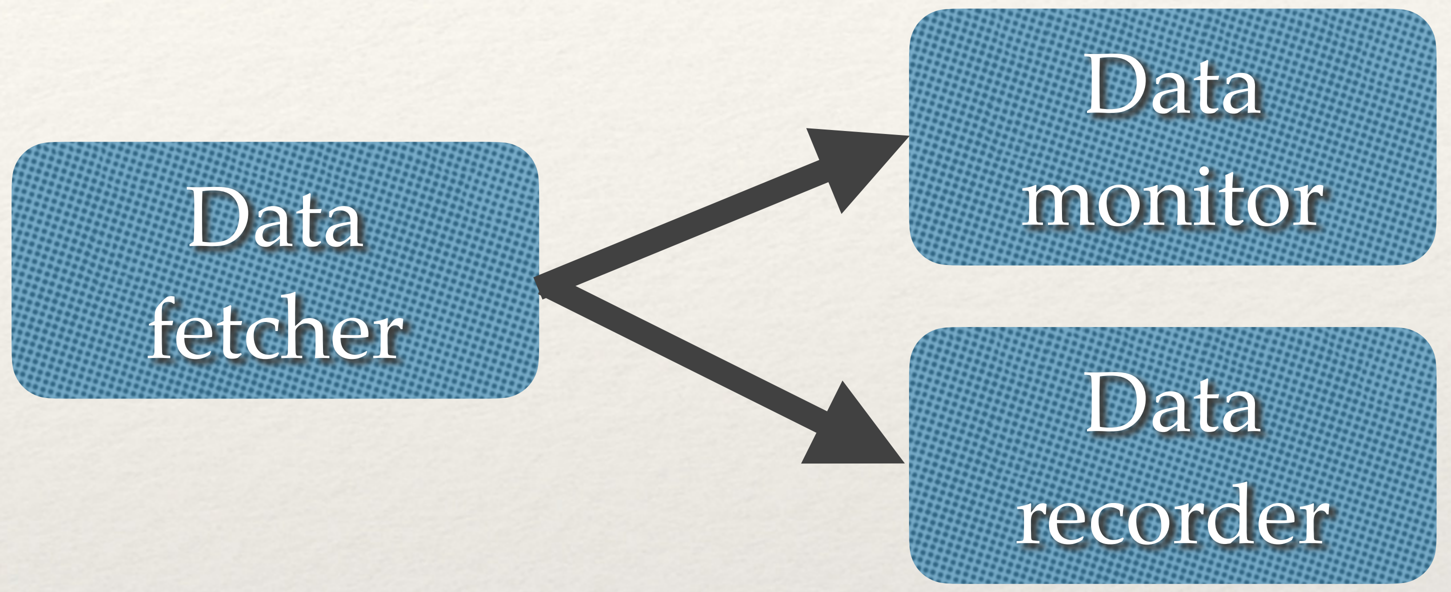
Run 101

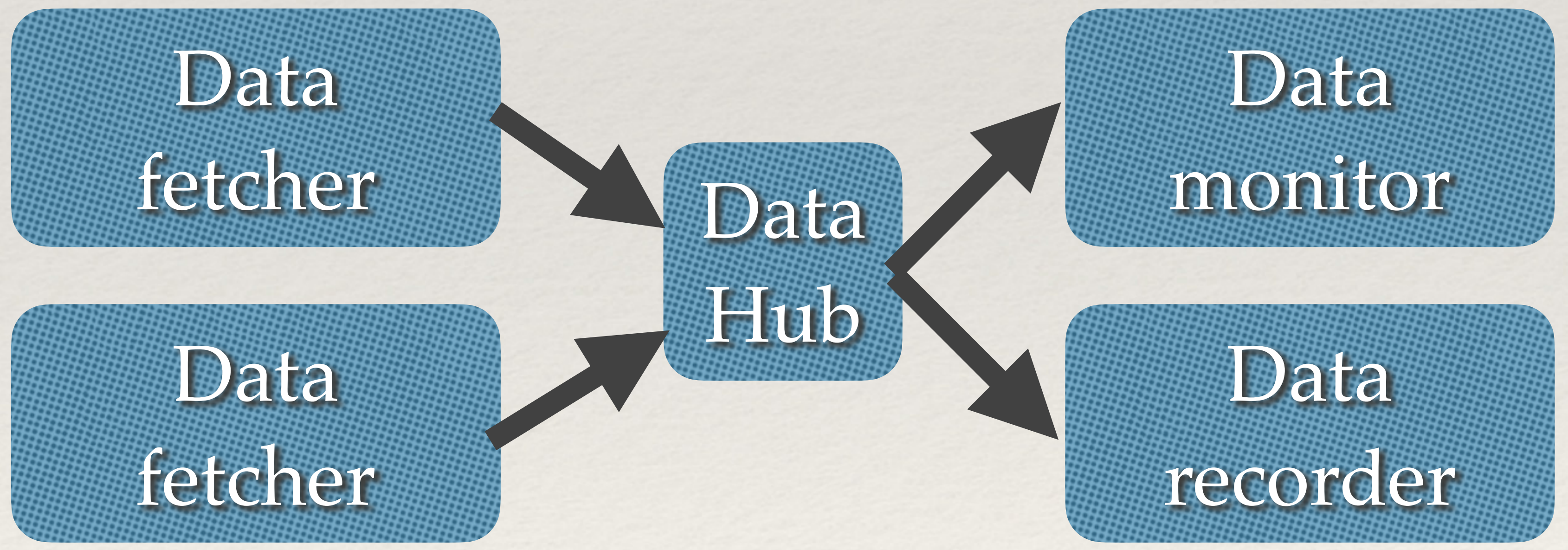
Event Builder

- All events are sorted and stored simple TTree by timestamp
  - Triggerless mode, also with VETO or trigger
- The user specifications
  - Event trigger detectors
    - Flexible condition settings NYI
  - Anti Coincidence setting
  - Time window
- Only offline
  - If we can have a good computational resource, I will implement the online version

Performance

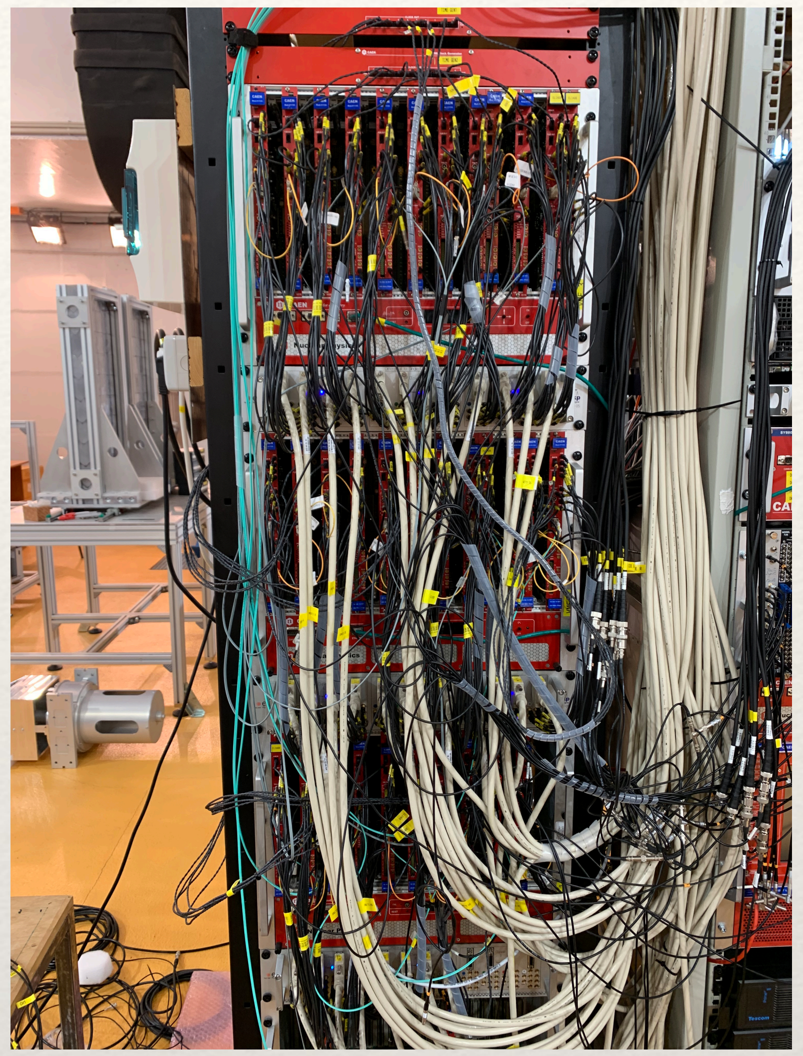
- Trigger rate, data transfer speed (One event 26B, can be 14B)
  - Simple setup: < 18 MHz, 480 MB/s
  - Realistic setup: < 4 MHz, 100 MB/s
- Now sending data by DAQ-Middleware (CORBA)
  - Synchronization of all components by middleware
  - Sometimes, one component waits for others
- Direct connecting (WebSocket): > 40 MHz, 1 GB/s/client
  - Disk speed and network speed can be a bottleneck

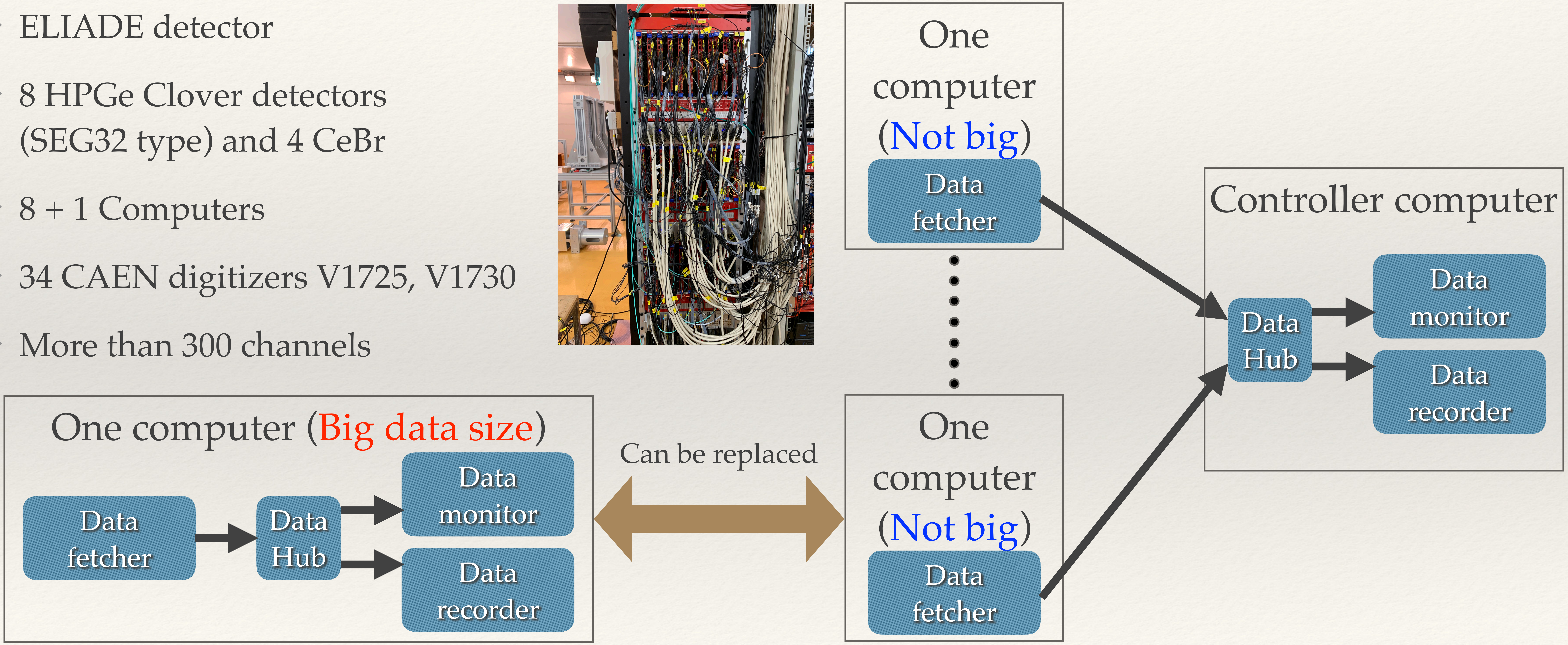




User cases

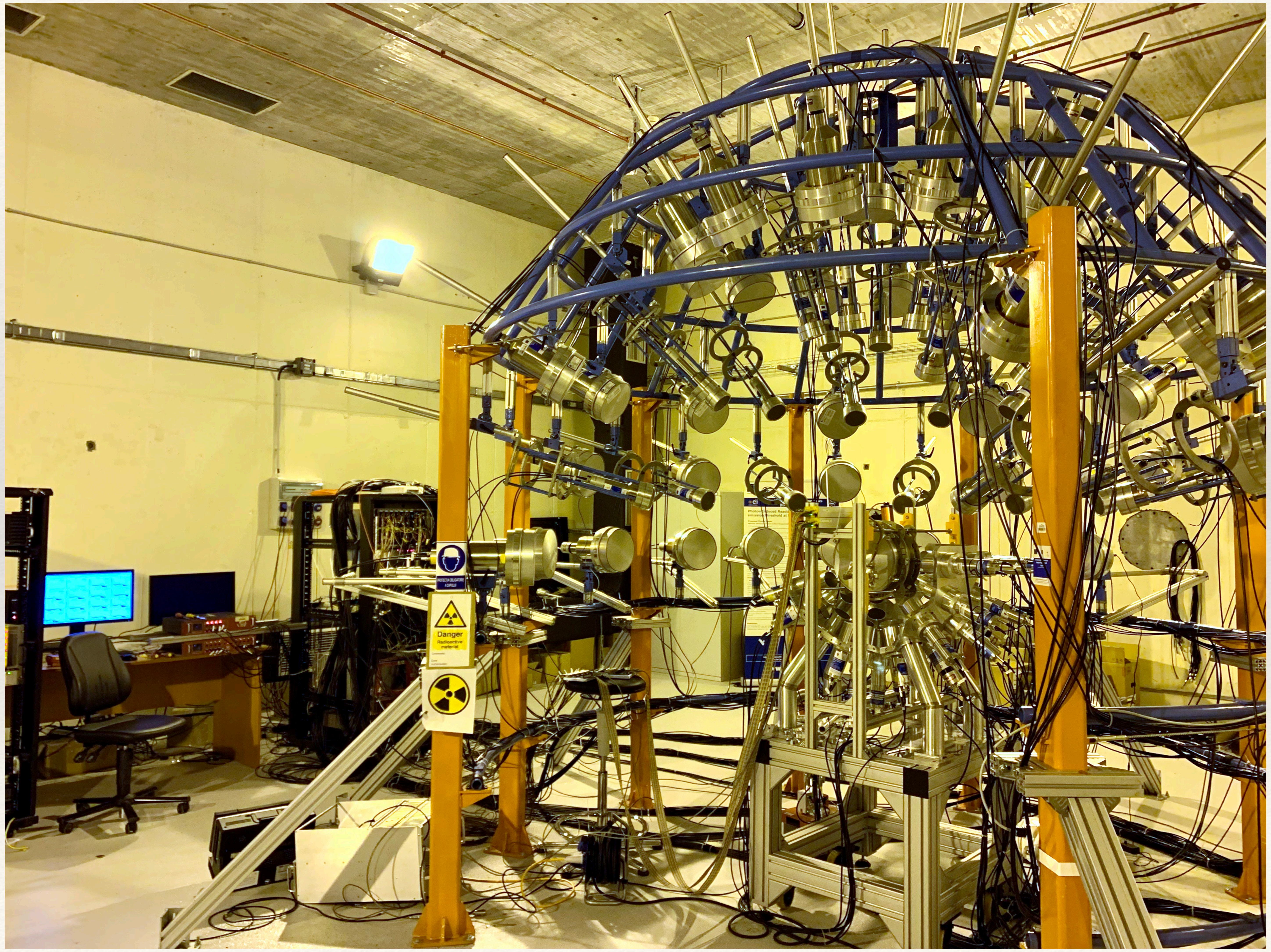
- ELIADE detector
- 8 HPGe Clover detectors (SEG32 type) and 4 CeBr
- 8 + 1 Computers
- 34 CAEN digitizers V1725, V1730
- More than 300 channels





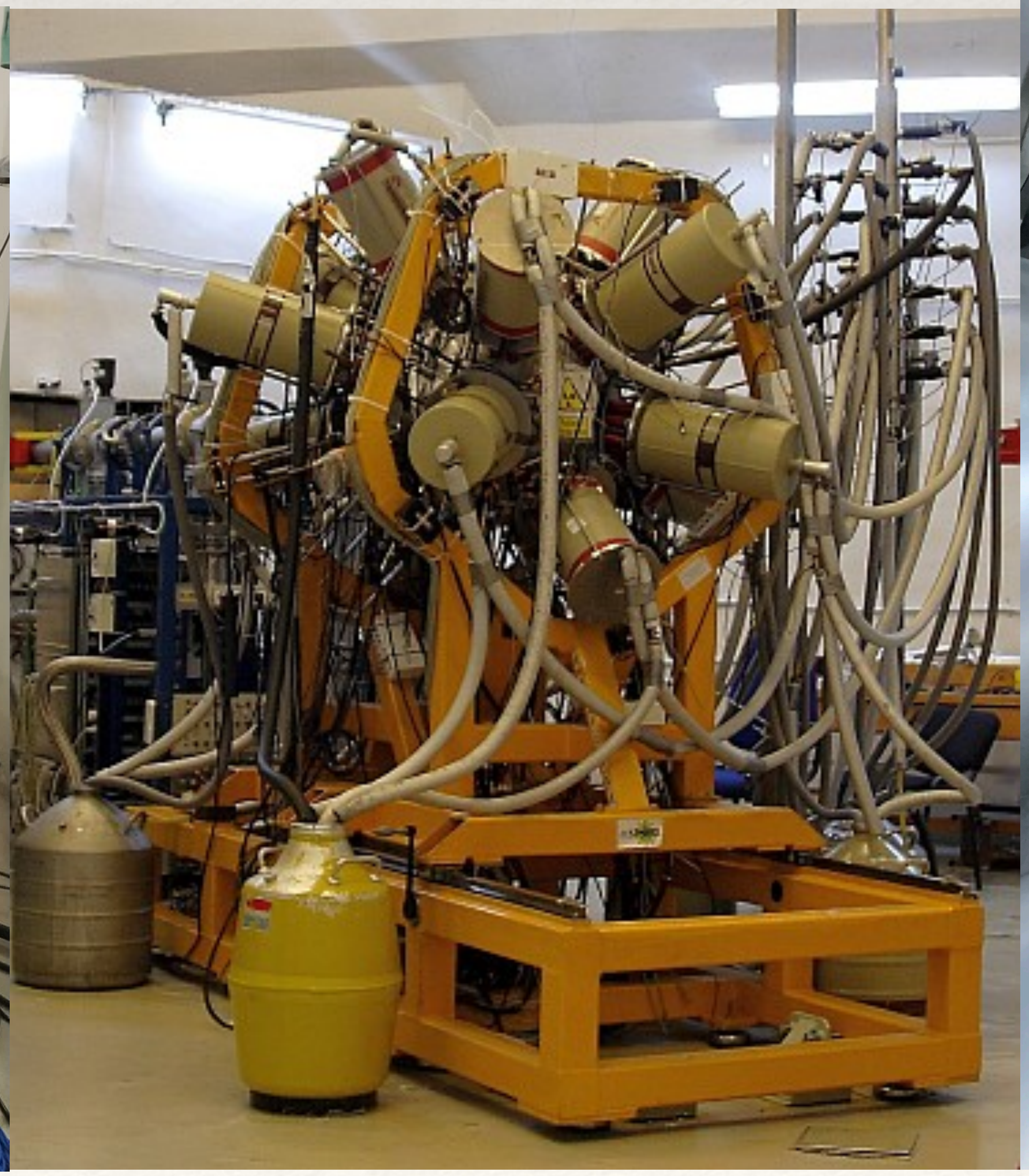
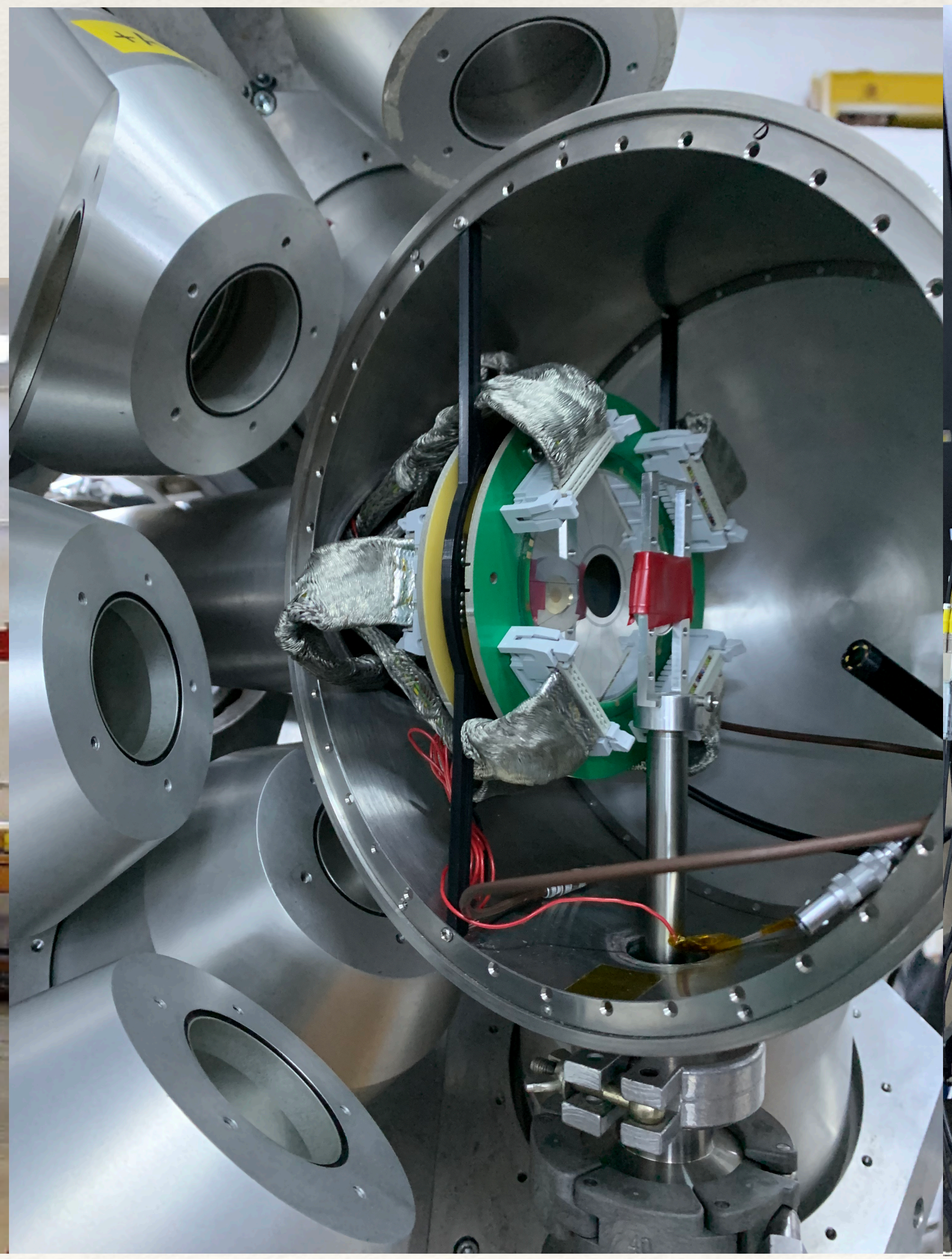
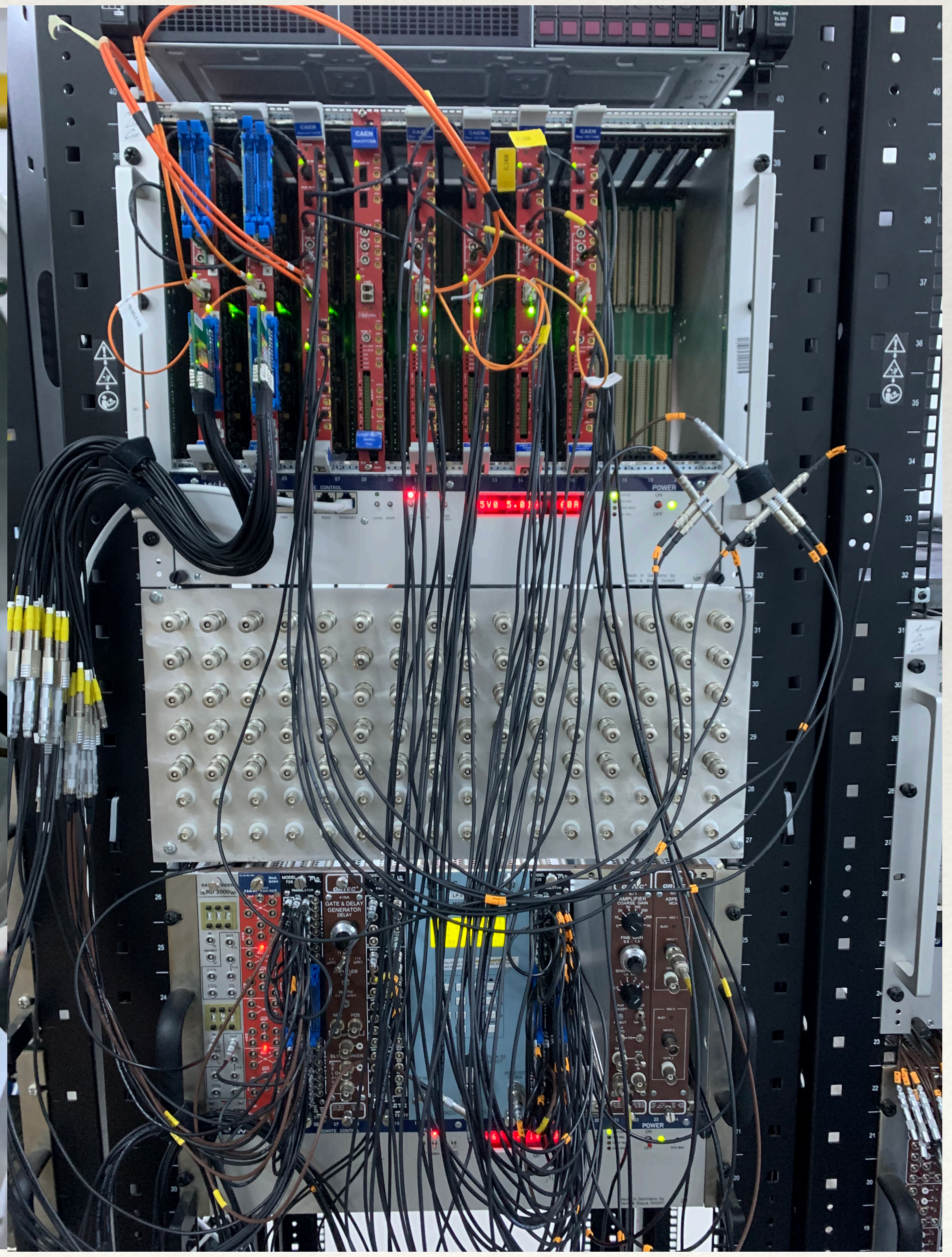
User cases

- ELIGANT detector
- 34 LaBr and CeBr detectors for γ
- 62 Liquid scintillator and Li glass detectors
- 2 DSSD Si detectors
- 1 computer
- Relatively simple



User cases

- 3MV and 9MV Tandem beamlines at IFIN-HH, Romania
- Publications
  - A. Kuşoğlu et al. Phys. Rev. Lett. 133, 072502
  - P.A. Soderstrom et al. IL NUOVO CIMENTO 47 C (2024) 58
  - S. Aogaki et al. NIM A 1056 (2023) 168628
  - R. Roy et al. EPJ Web of Conferences 297, 02007 (2024)
  - And more

Conclusion

- Pros
  - Fitting for many experiments
  - Well running with not-so-big data size or not necessary to merge data during an experiment
  - Controlling several electronics
  - ROOT format is good for researchers, most probably…

Conclusion

- Cons
  - DAQ-Middleware documents are mainly written in Japanese
    - My colleagues are mainly Romanian, European

Future plans

- Rewrite the CAEN digitizer controller class of the data fetcher
  - CAEN provides better libraries now
  - Test application running well with ^{60}Co source
- Implementing better GUI
- Replace DAQ-Middleware with our own framework
  - The designing and planning stage now

Acknowledgments

- Supported by
  - The Romanian Ministry of Research, Innovation and Digitalization under contract 10N/PN 23 21 01 06
  - The ELI-RO program funded by the Institute of Atomic Physics, Măgurele, Romania, contract number ELI-RO/RDI/2024_002 and ELI-RO/RDI/2024_004
  - And users

Codes on GitHub

- https://github.com/ELI-NP/DELILA
- https://github.com/aogaki/DELILA-WebAPI
- https://github.com/aogaki/DELILA-Controller
- https://github.com/aogaki/DELILA-Event
- Oat++ and DAQ-Middleware are also there

Thank you for your attention

DAQ group, GDED, ELI-NP