Implementations of streaming DAQ on actual detector systems

Y. Igarashi, M. Dozono[A], R. Honda, N. Kobayashi[A], C. S. Lin,
S. Ota[A], S. Y. Ryu[A], K. Shirotori[A], H. Sendai,
T. N. Takahashi[A]
KEK IPNS, [A]Osaka University RCNP

2024/4/23

IEEE Realtime 2024

Streaming capable DAQ software : Concept

- Overall management
- Overall control

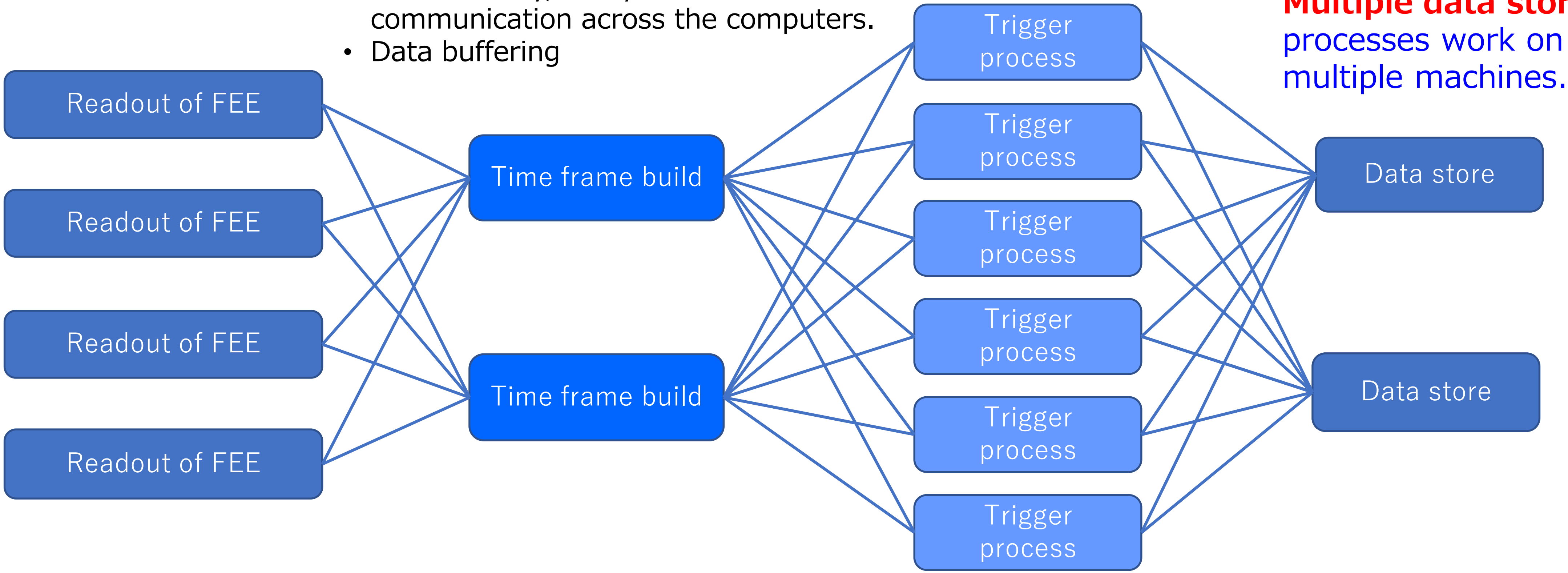
Bottleneck-less mesh connection

- One-to-many, many-to-one data communication across the computers.
- Data buffering

Load distribution

Multiple data storage

processes work on multiple machines.



How to achieve these?

- More advanced communication method than TCP/IP, socket
- Universal database

A streaming capable DAQ software

To accomplish to develop the streaming DAQ, we employed the followings

- The processes communicate to other many processes
  → ZeroMQ
    - One to many, many to one communication
    - Message queue works as a data buffer.
- Process and state management of a large number of processes
  → Key-Value database: Redis
    - Memory-oriented and fast response
    - Key-space notification
    - Pub/Sub → It can be used for control.
- State machine and control of it
  → FairMQ
    - FairMQ is a experiment framework developed for GSI/FAIR experiments.

→ We combined FairMQ and Redis.

FairMQ(core part) + Redis
→ NestDAQ (Network based streaming DAQ)

NestDAQ Process structure

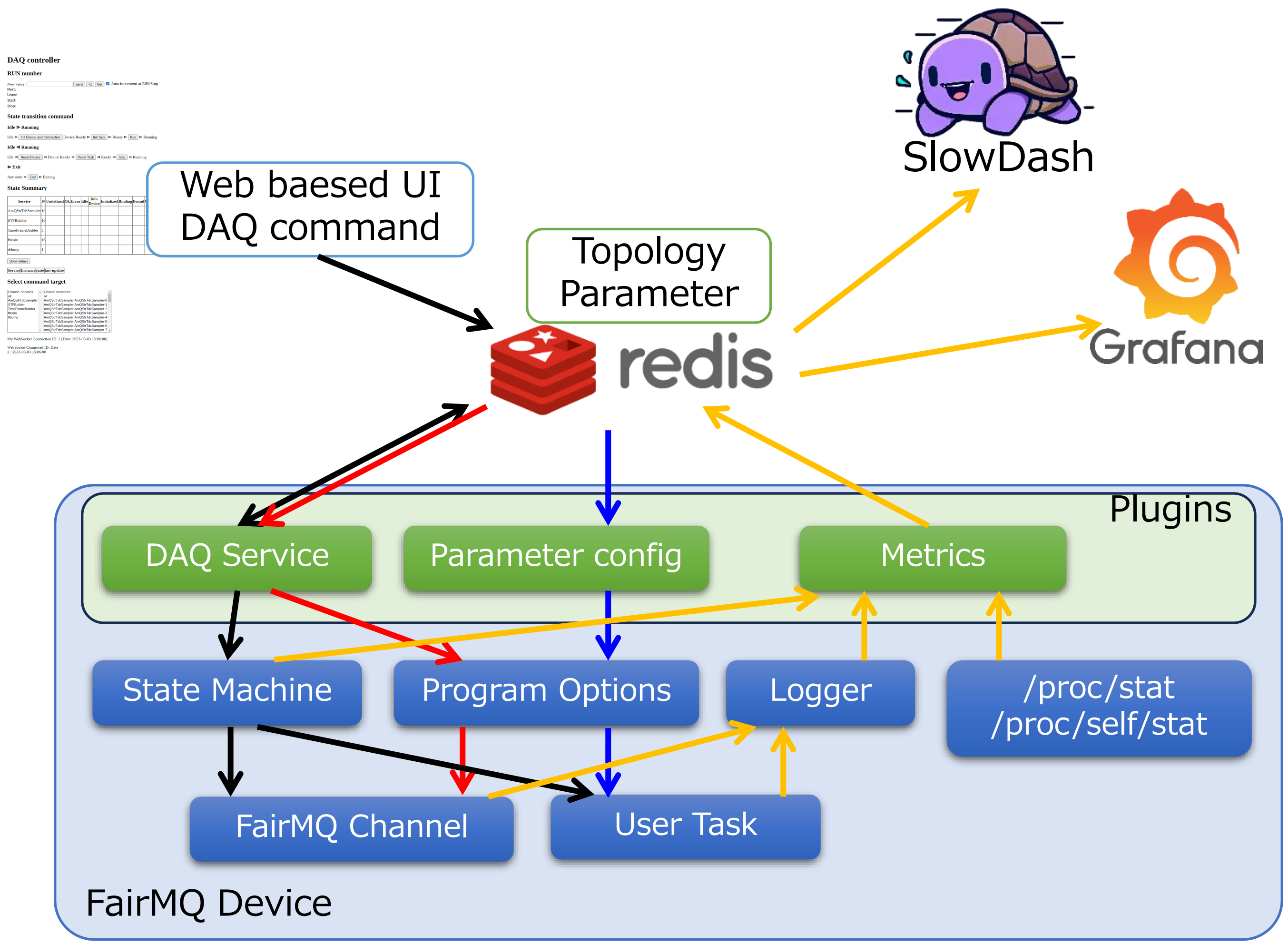
FairMQ can extend their functions by plugins.

FairMQ Plugins

- DAQ Service Plugin
  - Run control
    - Control the state machine
    - Set the run number
  - Service discovery
    - Semi-automatic connection configuration

- Metrics Plugin
  - Grasping the processes statuses

- Parameter config Plugin
  - Read program option from the command line or the database.
  - Read device initialization parameters from the database.



Configure the huge number of connections

DAQ Service : Service discovery

- It’s not realistic to hand write a connection table for over the 1000 connections.
  → Semi-automatic connection configuration

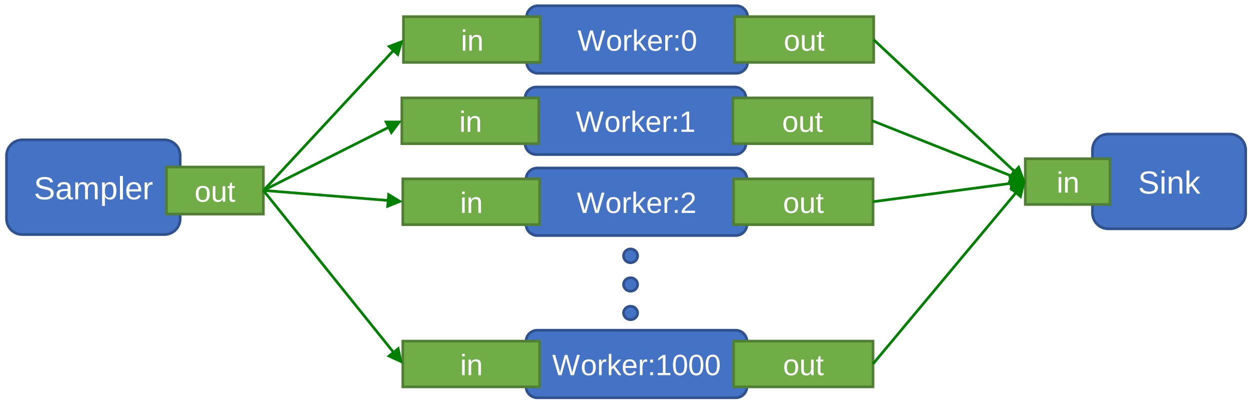
  - The database provides information about each process names, its data channel-ports and their connections
  - Service discovery configures connections regardless of the number of processes.

Example: An arbitrary number of worker processes

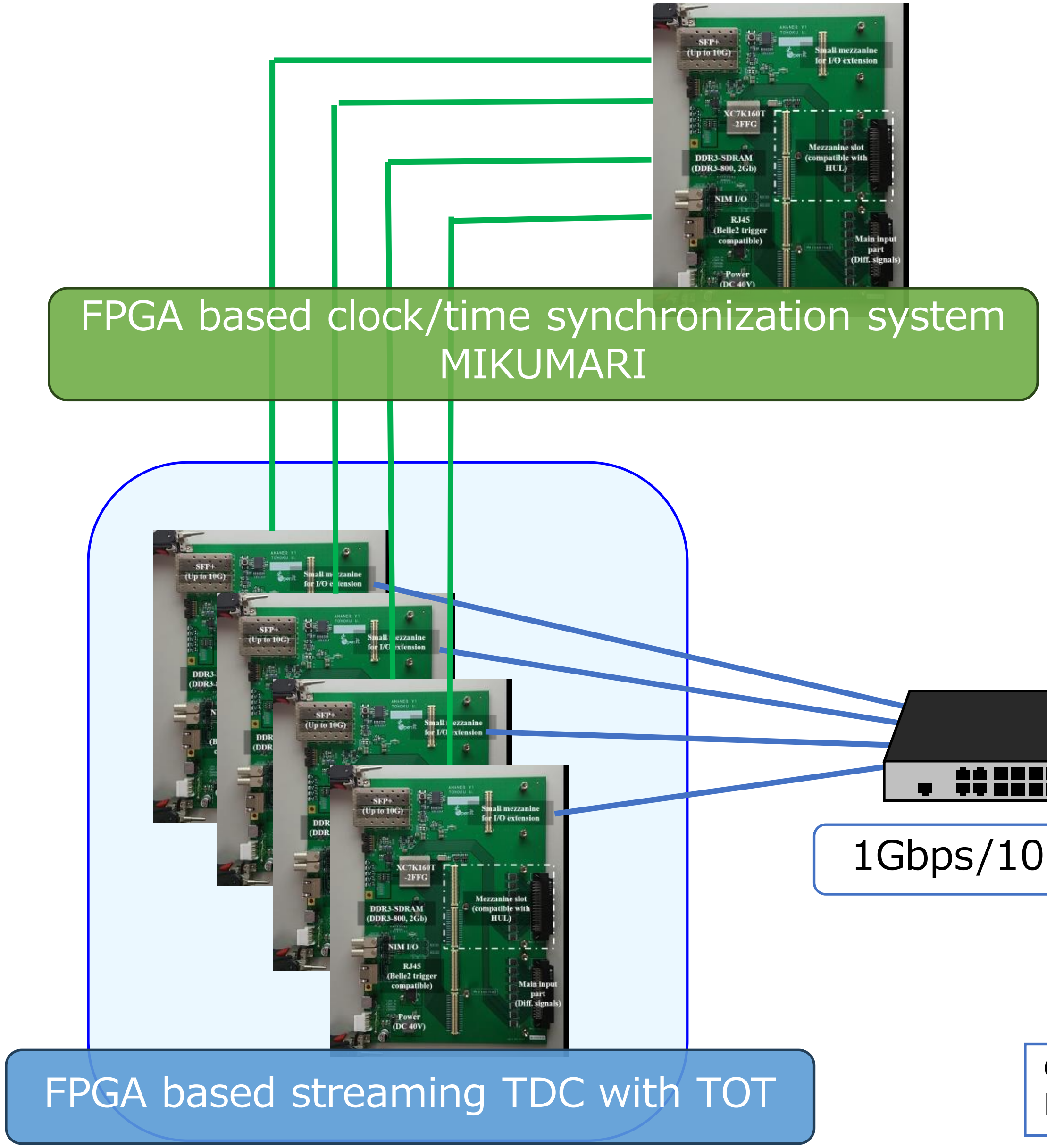
Topology data on the database

```
#------------------------------------------------------------------------
#             service      channel       options
#------------------------------------------------------------------------
endpoint      Sampler      out           type push   method bind
endpoint      Sink         in            type pull   method bind
endpoint      Worker       in            type pull   method connect
endpoint      Worker       out           type push   method connect

#------------------------------------------------------------------------
#        service1          channel1         service2      channel2
#------------------------------------------------------------------------
link     Sampler           out              Worker        in
link     Worker            out              Sink          in
```

Configured topology structure



Readout devices and clock/time synchronization

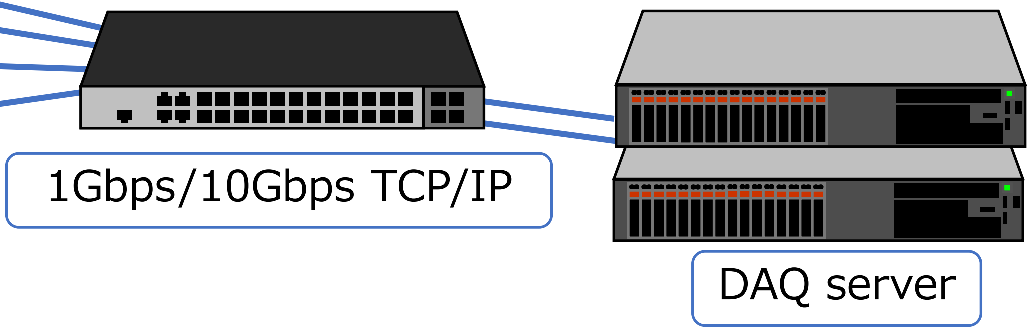


Front-end electronics (FEE)

- Data transfer from FEE was used FPGA implemented TCP/IP (SiTCP).
- They synchronize by a clock/time system(MIKUMARI).
- FEE handles continuous data by dividing it into a time intervals named hart-beat frame(HBF).
  - Current HBF is 524.288 us (Clock 125MHz, 16bit)

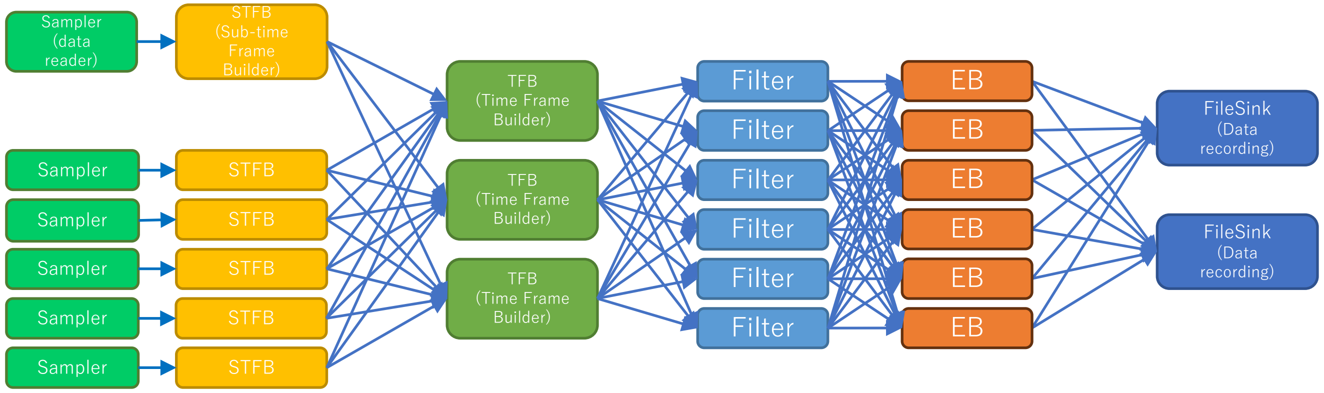
Continuous TDC data stream

TDC data of HBF | HB delimiter | TDC data of HBF | HB delimiter | TDC data of HBF | …

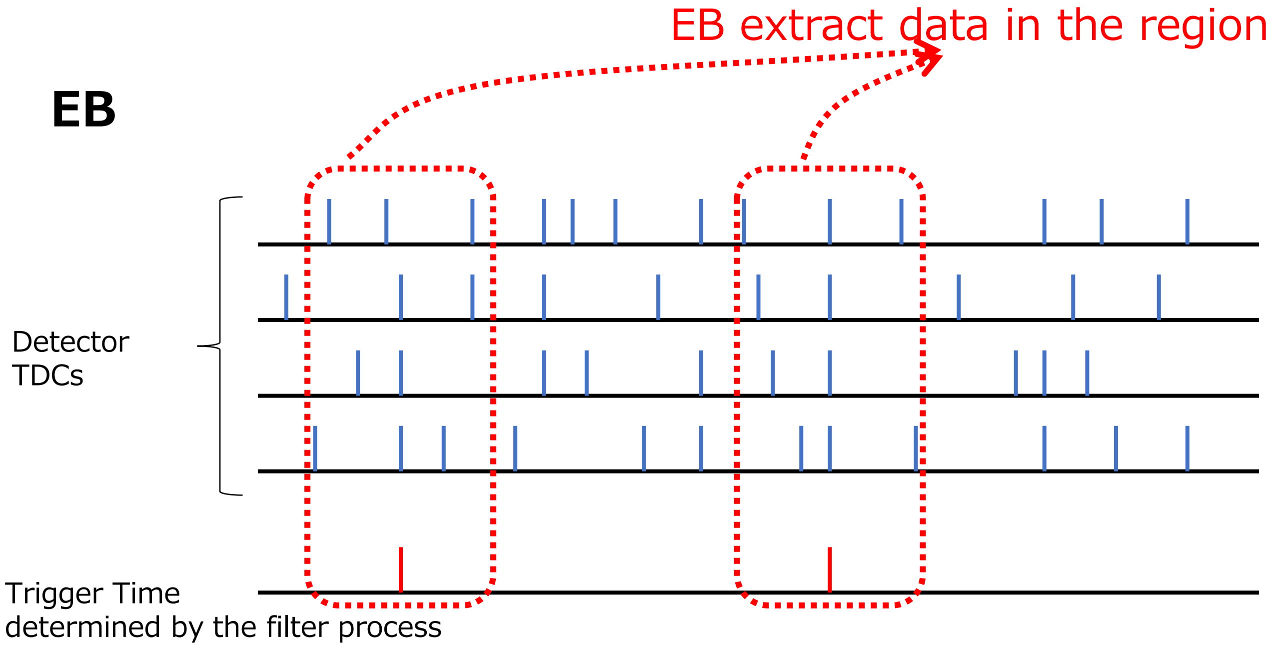


DAQ processes and behavior

Common DAQ configuration



EB



- Sampler
  - Reading data from front-end electronics
- Sub-Time Frame Builder (STFB)
  - Chopping the data from the sampler for each HBF,and several HBF are put together to make a Sub-Time Frame.
- Time Frame Builder (TFB)
  - Making Time Frame combined from Sub-Time Frame data from each Sub-Time Frame Builder
- Filter/Online software trigger
  - Finding the good event in the time frames.
- Event builder (EB) (for Streaming Read Out)
  - Extracting the data in the time near the found trigger time.
  - EB reduces the size of data.
- FileSink
  - Writing received data to the file.

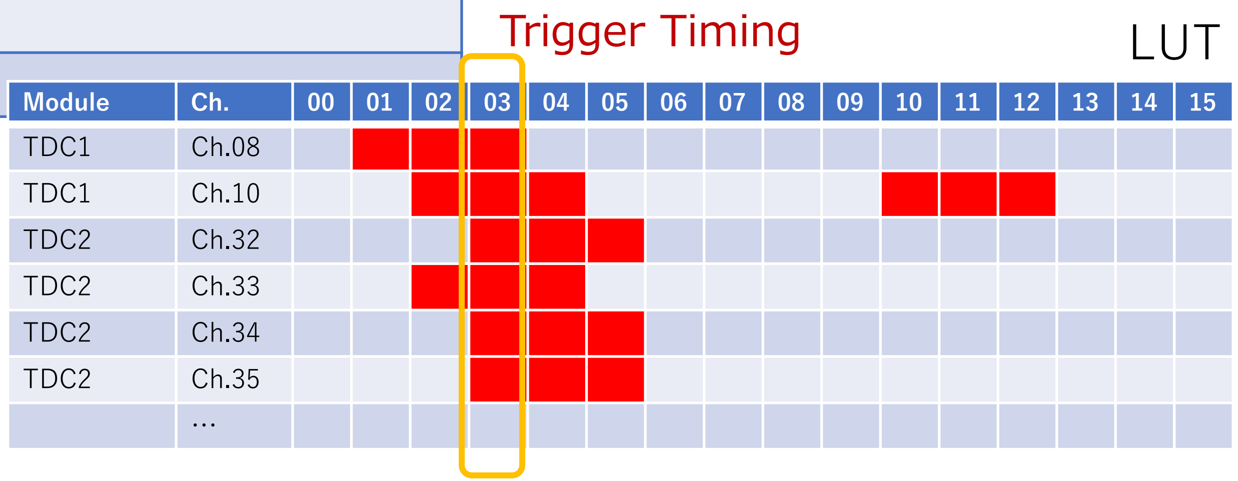
General logic online triggering filter by LUT

- Behavior of a general combinational logic filter
  - Create a logic table by calculation from the expression before the RUNNIG state.

1. Check marks to an array of HBF length.
2. Scan the array and picking up the array index where the LUT returns true.
3. Store the index where the value changes to a vector. (edge detection)

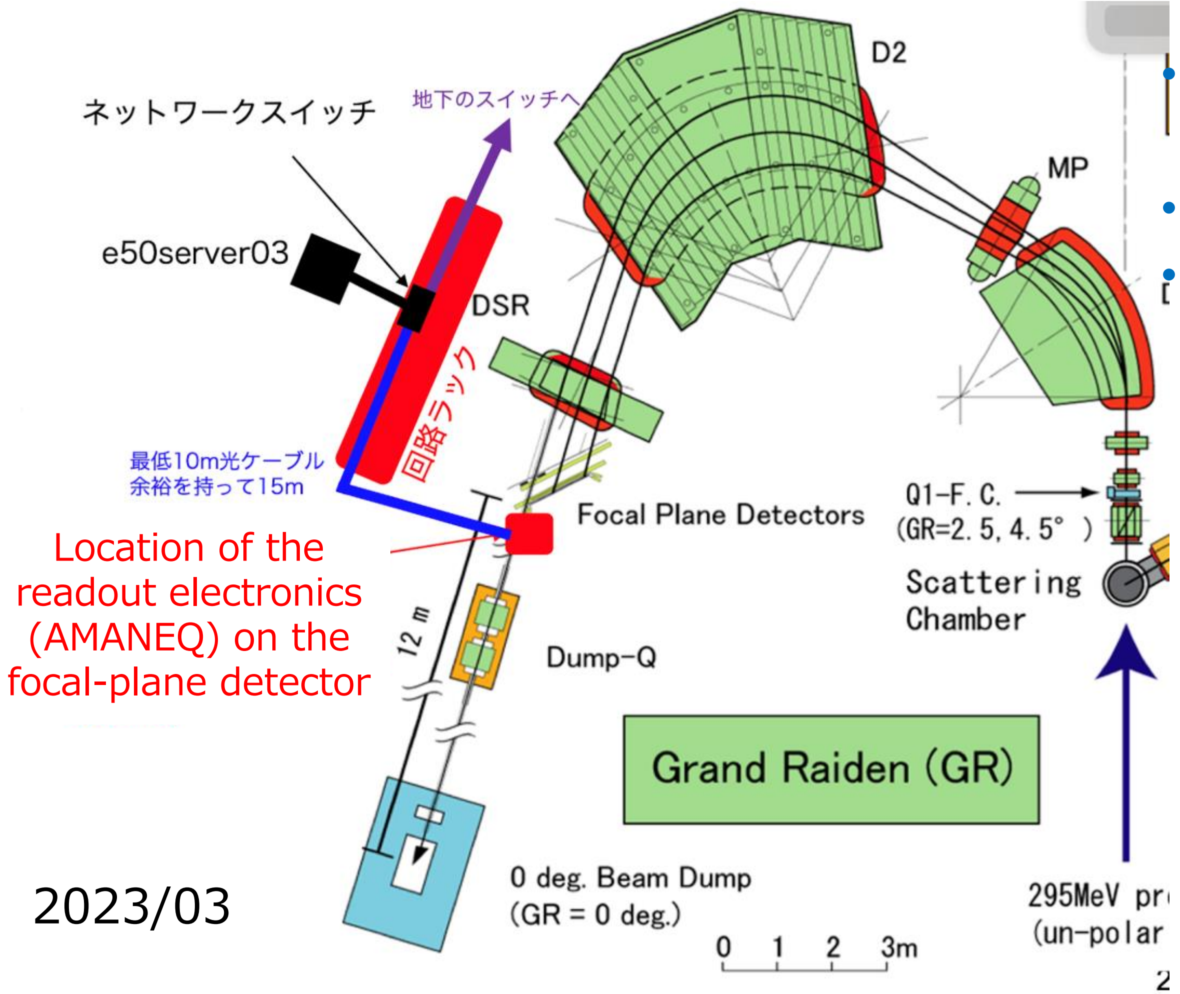
KEY: parameters:LogicFilter (Hash of Redis)

Field	Value
Trigger-signals	(0xc0a802a9 8 0) (0xc0a802a9 10 0) (0xc0a802aa 32 0) (0xc0a802aa 33 0) (0xc0a802aa 34 0) (0xc0a802aa 35 0)
Trigger-expression	(0 & 1) & (2 & 3) & (4 & 5)
Trigger-width	3



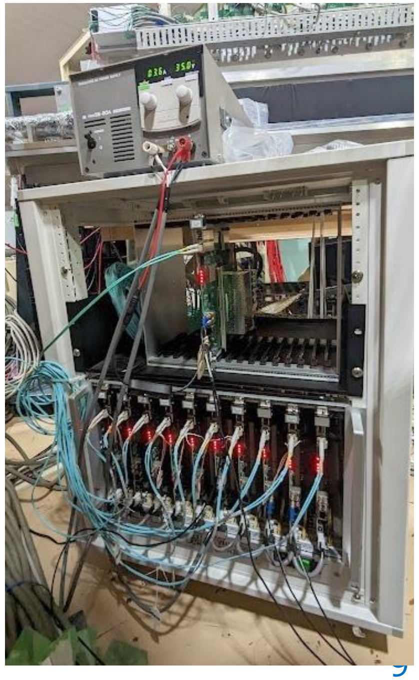
First trial to apply a streaming DAQ to a actual detector

RCNP Grand Raiden spectrometer



2023/03

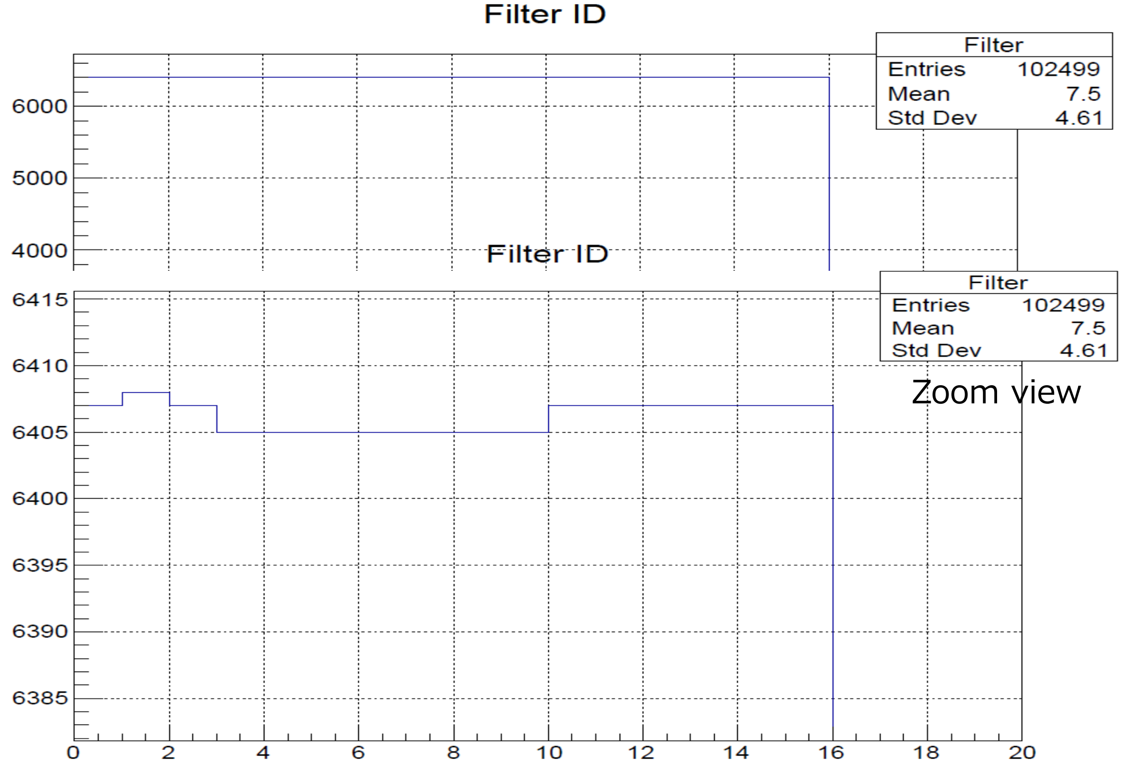
- Plastic scintillation counters
  → FPGA base streaming HR-TDC with TOT x2
- Drift chambers
  → FPGA base streaming TDC with TOT x8
- Clock distribution system “MIKUMARI”
- Software trigger process (coincidence) “NestDAQ”
  → Confirmation of the streaming DAQ



First application of a streaming-readout data-acquisition system, products of SPADI Alliance, to physics experiments at RCNP towards the standardization
Shinsuke Ota (RCNP, Osaka University), April 23 morning

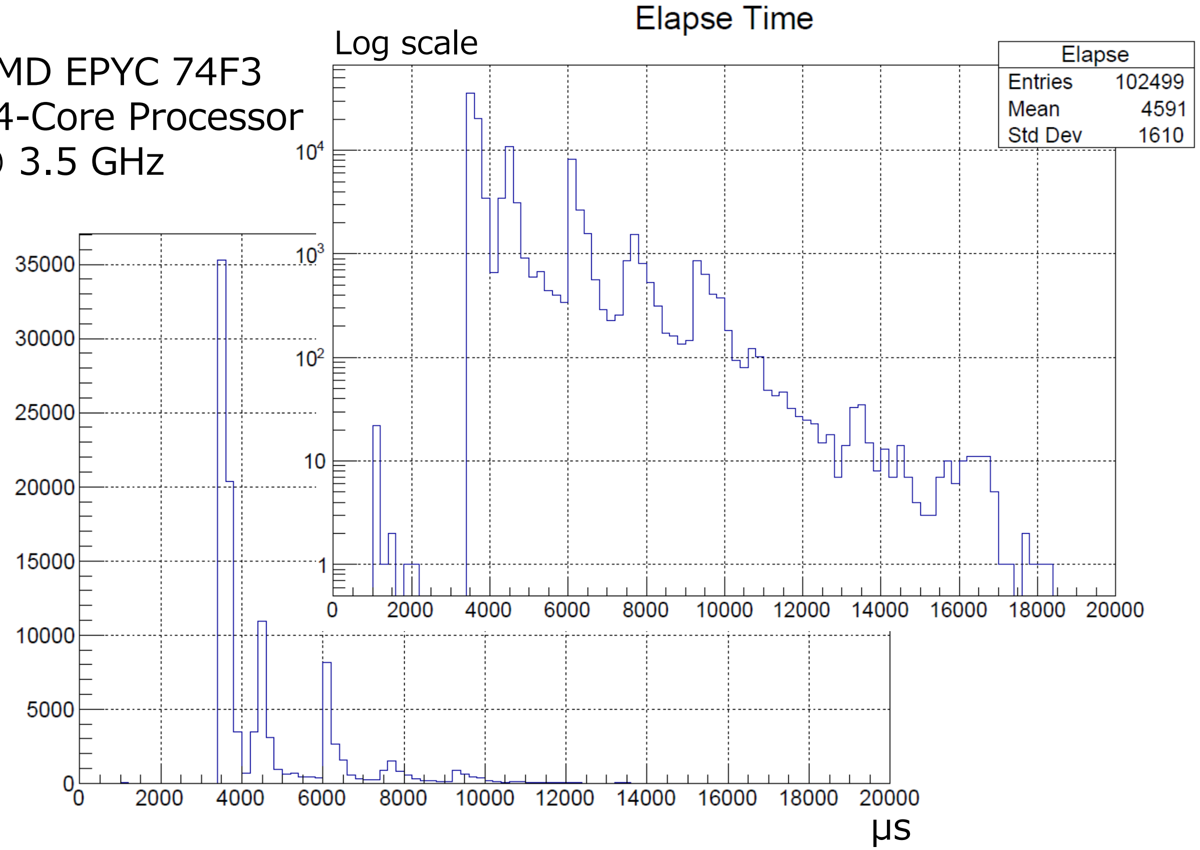
Behavior of the filter process

Load distribution



5HBF consumption duration (w/o data transfer)

AMD EPYC 74F3
24-Core Processor
@ 3.5 GHz



Filter 16 process

The load is distributed moderately by Round-Robin + Skip at Queue-Full algorism without a global task dispatcher.

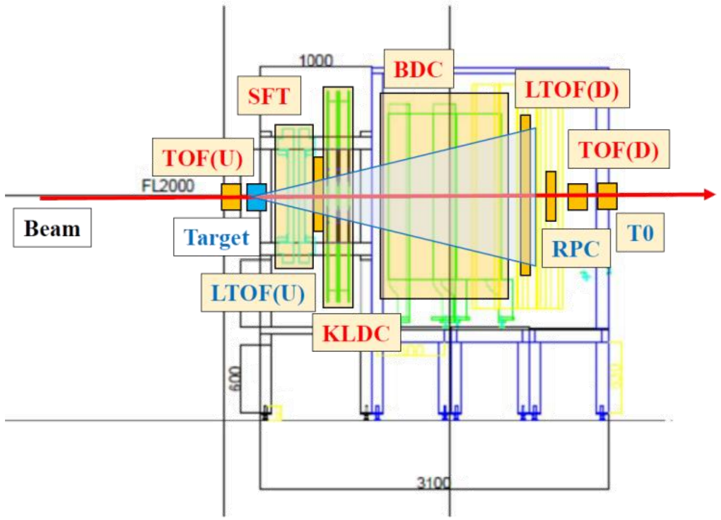
Trigger: Ch1 * Ch2 * Ch3 * Ch4

It should be processed during 5HB 0.524*5ms = 2.62ms

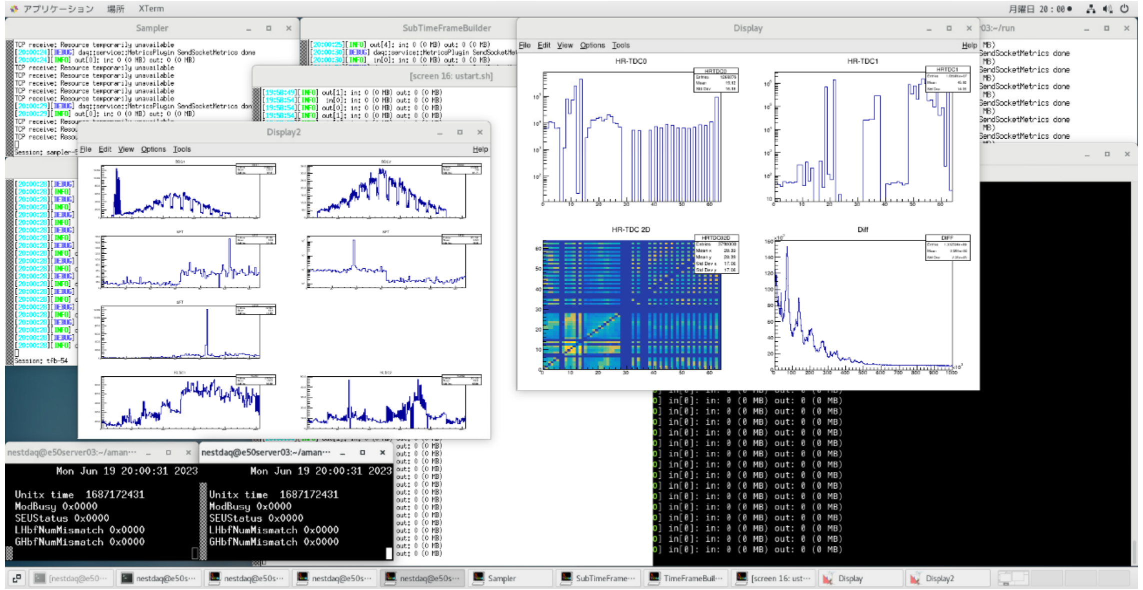
→ It is possible to process to use more than 2 processes because the average consumption time is 4.6 ms.

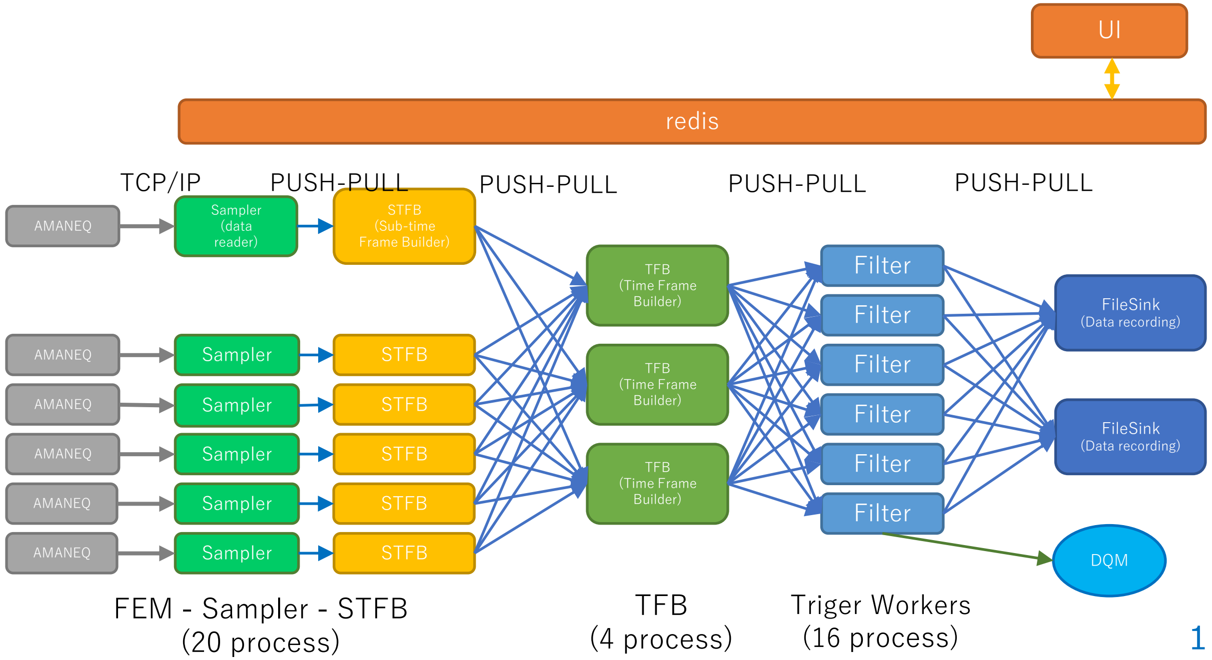
E50 detector/DAQ test in J-PARC HD K1.8BR

- Front-end electronics, number of channels
  - HRTDC x2 : 128 channel
  - TDC x15 : 1920 channel
  - MIKUMARI x3 : 96 channel
- Combinational logic trigger process by LUT
- Data Quality Monitor using PUB/SUB communication by the Probe port
- Data flow (at recording):
  - ～180MB/s (average of Flat Top ON/OFF)
  - ～240MB/s (at Flat Top ON)



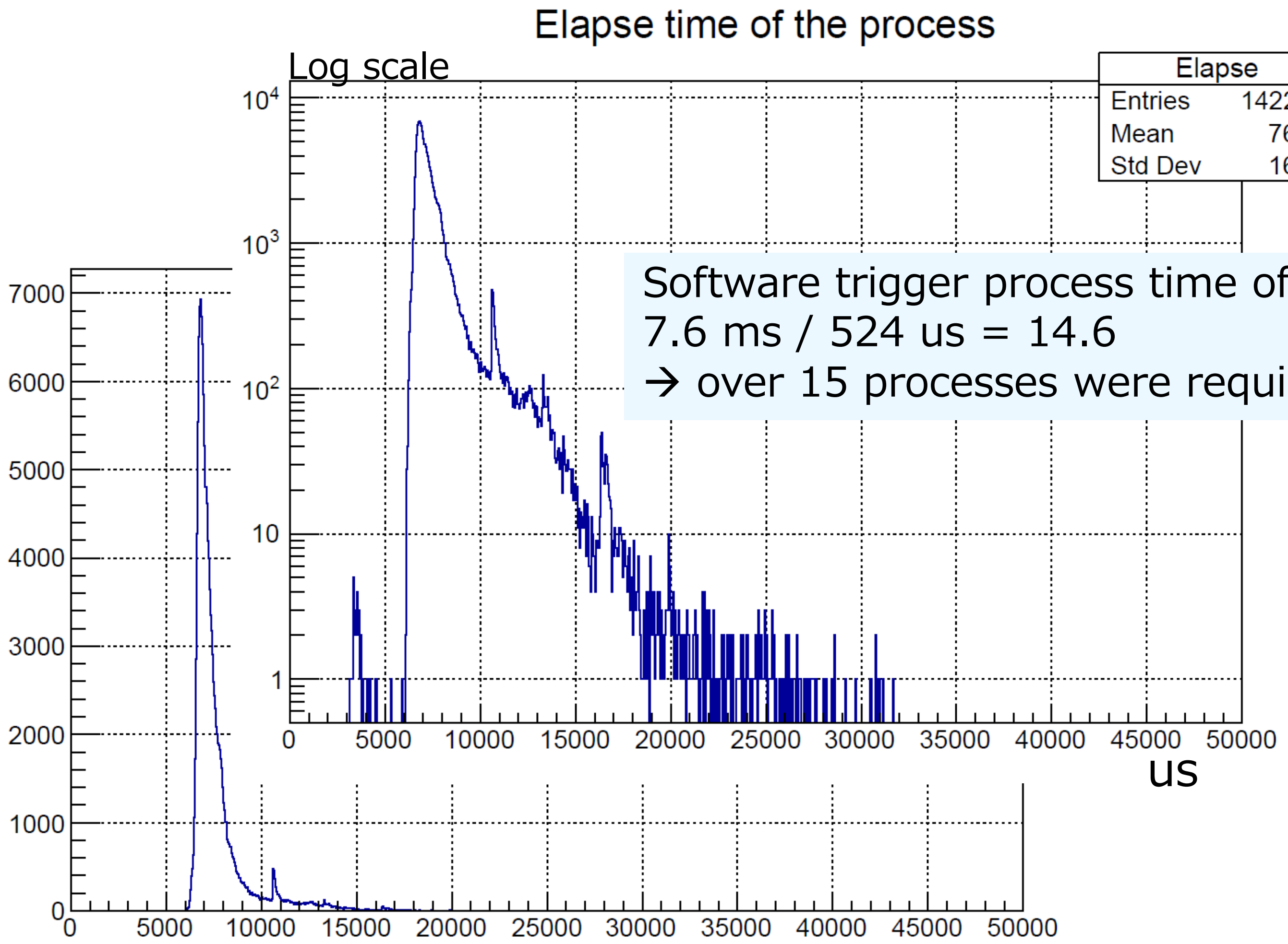
2023/06



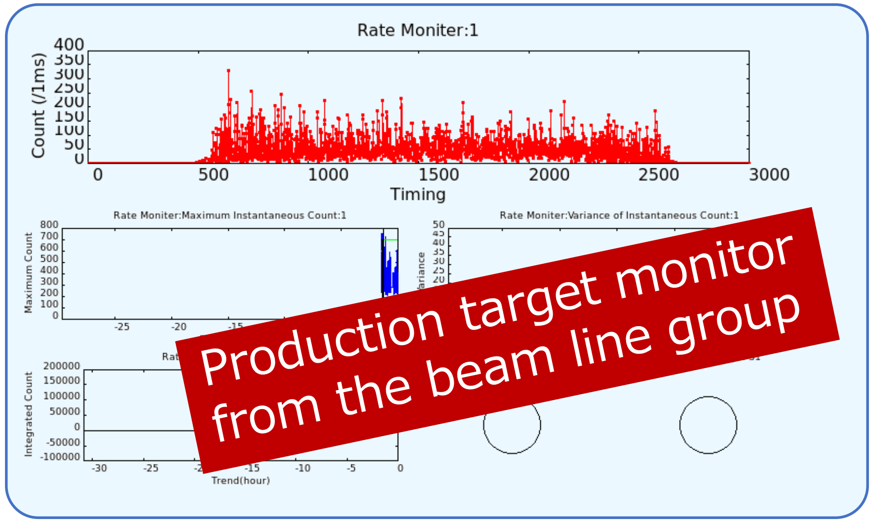


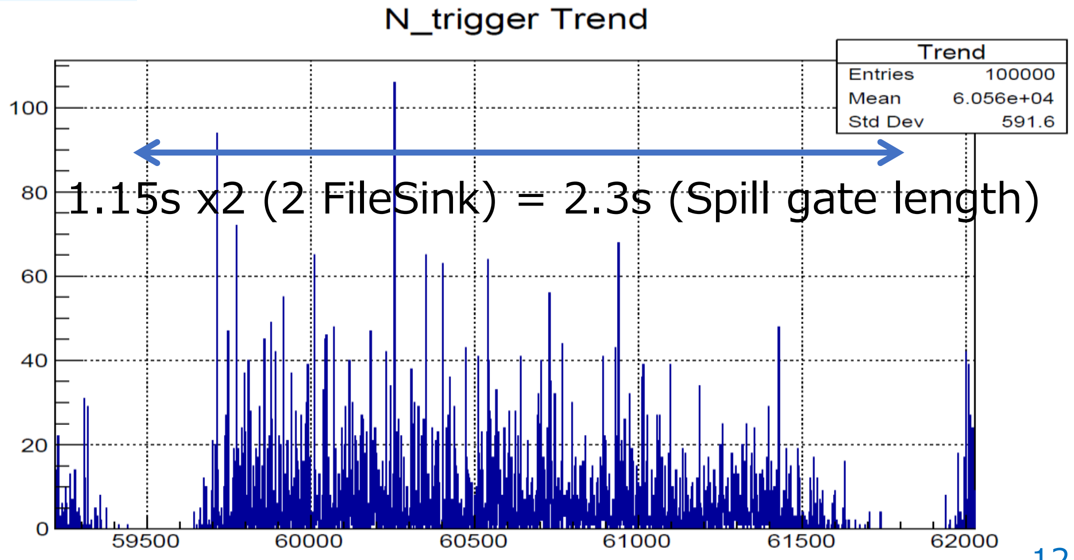
Elapsed process time and counting trends

Trigger logic: ((D1L*D1R)+(D2L*D2R)+(D3L*D3R))*((U1L*U1R)+(U2L*U2R))



Software trigger process time of 1 HBF
7.6 ms / 524 us = 14.6
→ over 15 processes were required





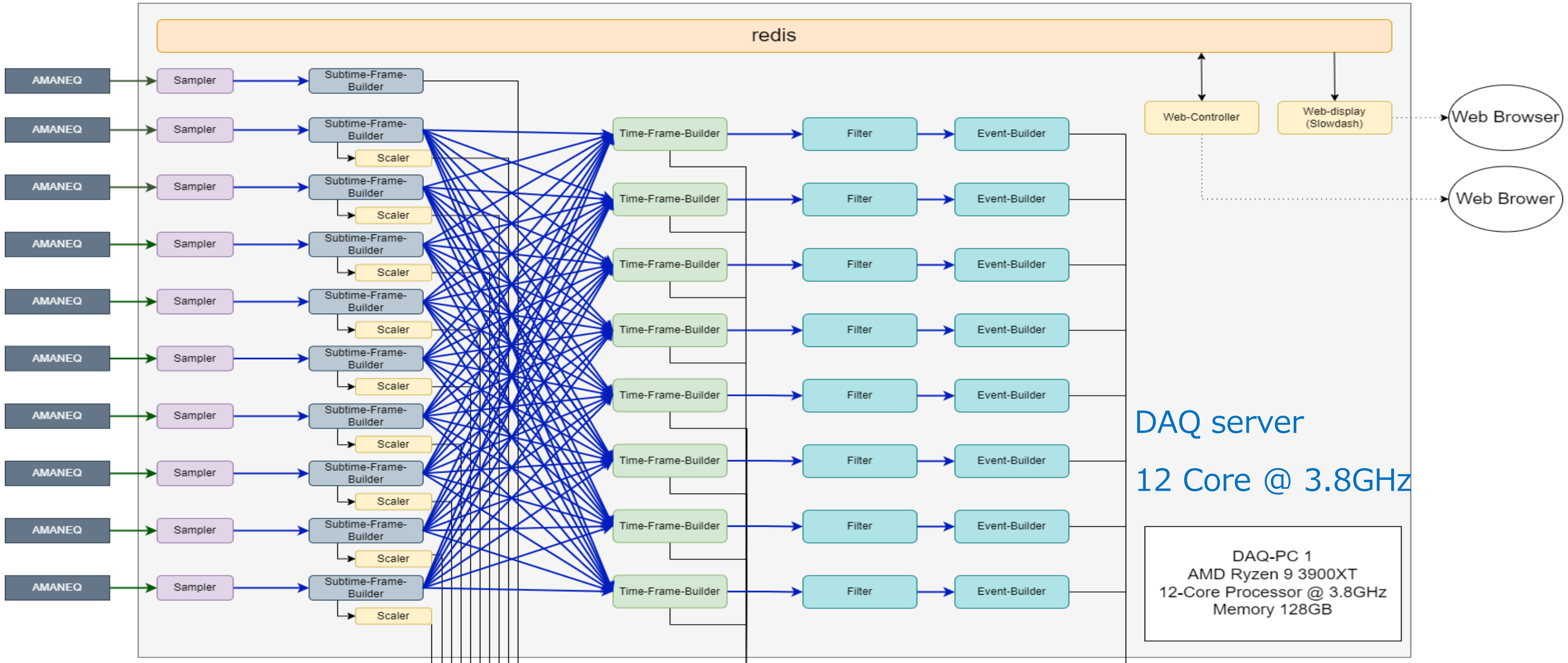
RCNP GR/WS E585

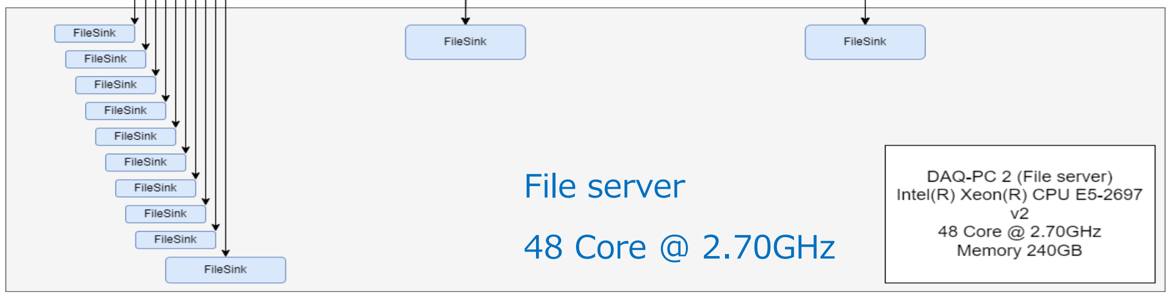
Appling a real physics data taking on Grand Raiden

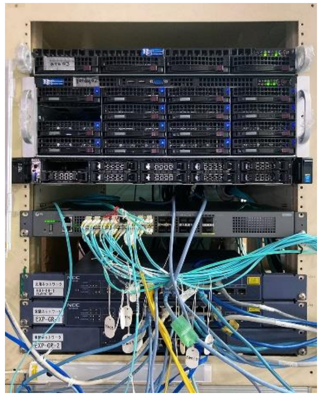
- General logic trigger filter
- Event builder (for SRO)
- Recording pre-scaled unbiased data
- Software scaler
- Web UI
  - DAQ controller
  - Online monitor by SlowDash*
    - Data flow visualization
    - Software scaler
    - Issue flag display

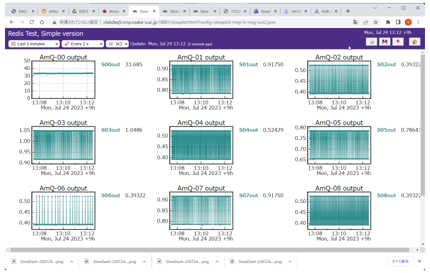
2023/07

Two server PC configuration









*SlowDash: A web tool for control and monitoring of concurrent systems
Developed by Sanshiro Enomoto, University of Washington
https://github.com/slowproj/slowdash

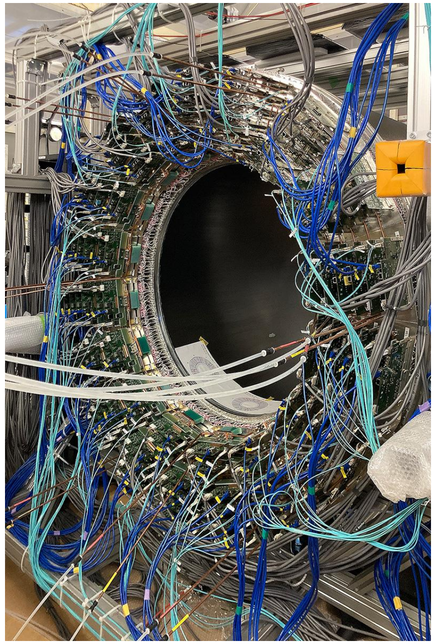
Topology configuration data of RCNP E585

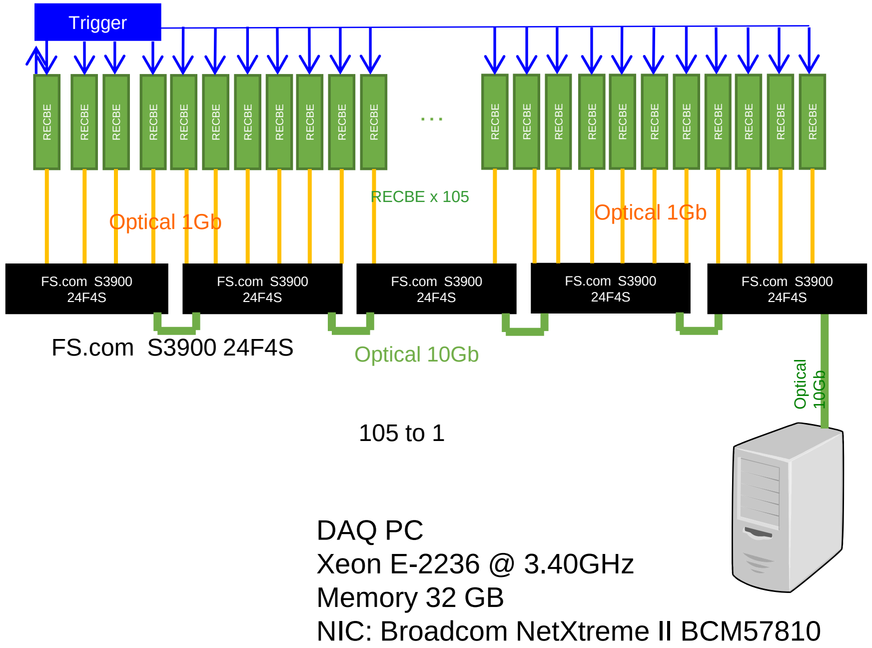
```
#--------------------------------------------------------------------------------
#          service              channel     options
#--------------------------------------------------------------------------------

# Sampler
endpoint   AmQStrTdcSampler    out         type push  method bind
endpoint   MikuTdcSampler      out         type push  method bind

# STF
endpoint   STFBuilder          in          type pull  method connect
endpoint   STFBuilder          out         type push  method connect autoSubChannel true
endpoint   STFBuilder          dqm         type push  method connect

endpoint   MikuSTFBuilder      in          type pull  method connect
endpoint   MikuSTFBuilder      out         type push  method connect

# DQM
endpoint   Scaler              in          type pull  method bind
endpoint   Scaler              out         type push  method connect

# TF
endpoint   TimeFrameBuilder    in          type pull  method bind
endpoint   TimeFrameBuilder    out         type push  method connect autoSubChannel true
endpoint   TimeFrameBuilder    decimator   type push  method connect

#fltcoin
endpoint   fltcoin             in          type pull  method bind
endpoint   fltcoin             out         type push  method connect autoSubChannel true

# EB
endpoint   EventBuilder        in          type pull  method bind
endpoint   EventBuilder        out         type push  method connect

# Sink
endpoint   FileSink            in          type pull  method bind portRangeMin 22001 portRangeMax 22100
endpoint   MikuSink            in          type pull  method bind portRangeMin 22201 portRangeMax 22300
endpoint   DecSink             in          type pull  method bind portRangeMin 22401 portRangeMax 22500
endpoint   ScrSink             in          type pull  method bind portRangeMin 22601 portRangeMax 22700
```

```
#--------------------------------------------------------------------------------
#        service1            channel1     service2            channel2
#--------------------------------------------------------------------------------
link     AmQStrTdcSampler    out          STFBuilder          in
link     MikuTdcSampler      out          MikuSTFBuilder      in
link     STFBuilder          out          TimeFrameBuilder    in
link     STFBuilder          dqm          Scaler              in
link     MikuSTFBuilder      out          MikuSink            in
link     TimeFrameBuilder    out          fltcoin             in
link     TimeFrameBuilder    decimator    DecSink             in
link     fltcoin             out          EventBuilder        in
link     EventBuilder        out          FileSink            in
link     Scaler              out          ScrSink             in
```

The option to use strict round-robin distribution with peer-to-peer connection for time frame building

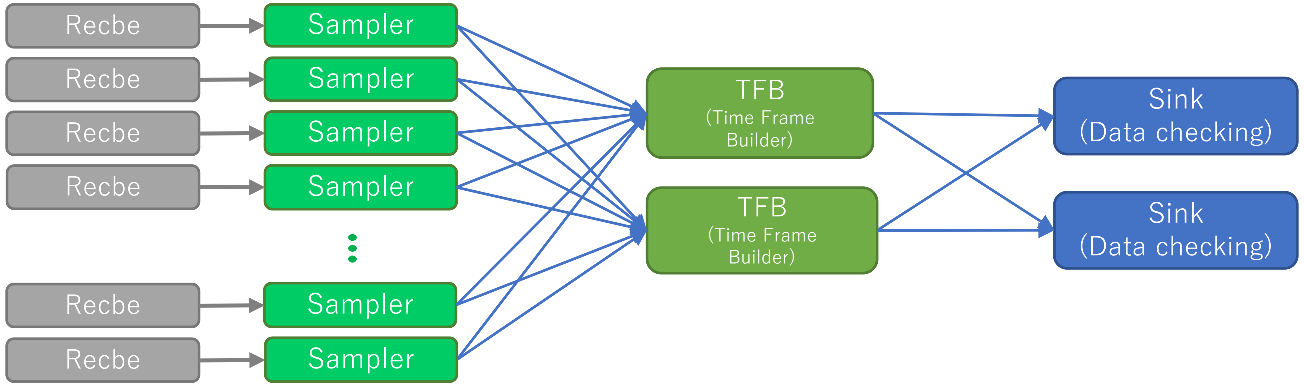
Trial to read a triggered DAQ (COMET CDC)

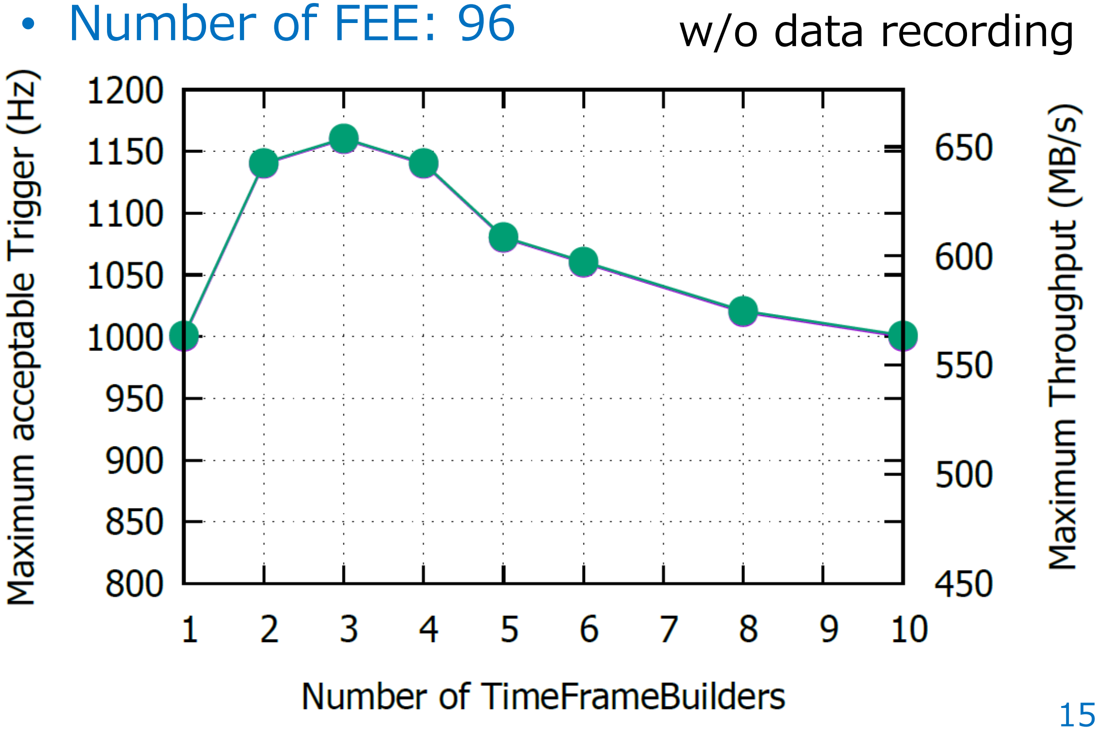




It works fine with a Sampler that reads data as "event by event" with the event ID instead of the time-frame ID.

- Event size: 6156B
- Number of FEE: 96



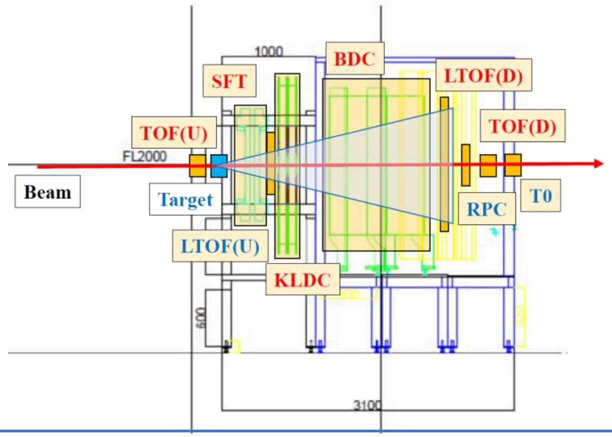


E50 detector/DAQ test 2nd trial at 2024

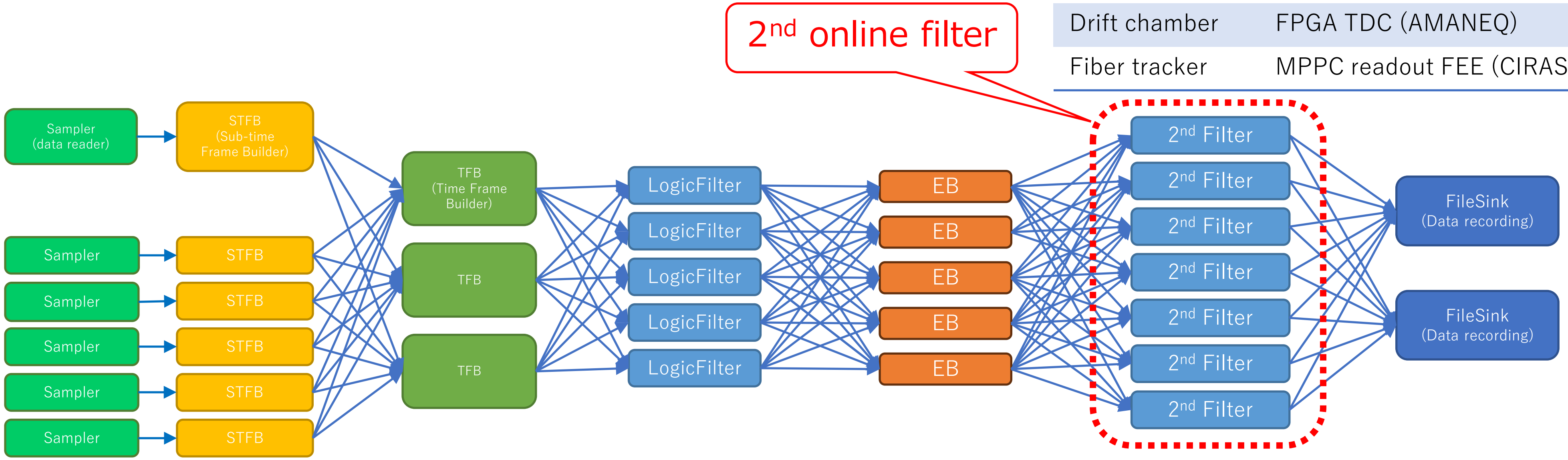
We are taking beams in the J-PARC Hadron Hall from April 12th to May 27th.

We are trying to improve the streaming DAQ.

- Improved clock/timing system
- Improved communication format
- 2nd online filtering and data taking for 2nd online filtering study
  - TOF
  - Tracking
  - Introduction of GPUs



Detector	Front-end electronics	Number of modules	Number of channels
	Clock/time system (MIKUMARI)	4	128
TOF	FPGA HR-TDC (AMANEQ)	2	128
Drift chamber	FPGA TDC (AMANEQ)	15	1920
Fiber tracker	MPPC readout FEE (CIRASAME)	18	2304



Ongoing now!

Summary

- The development of a streaming capable DAQ software framework "NestDAQ" based on FairMQ and Redis is in progress.

- We had some experience with DAQ using this framework.
  - We were able to acquire data in several detector systems, including the acquisition of physics experiment data.
  - The software part of the DAQ can be used not only for streaming DAQ but also for triggered DAQ.
  - It also works with DQMs or online monitors.

- To do or in progress
  - Log Collector
    - How do we manage many logs from many processes.
    - We are evaluating popular log collectors such as Fluentbit.
  - Organized data handling for inter-process communication.
  - More effective and advanced online trigger filters
    - Tracking, GPU processing, …
  - Good UI…

Thank you for your attention!