A DAQ software framework for SRO

IGARASHI Youichi
KEK

2023/11/28

Streaming readout Workshop SRO-XI

Streaming capable DAQ software : Concept

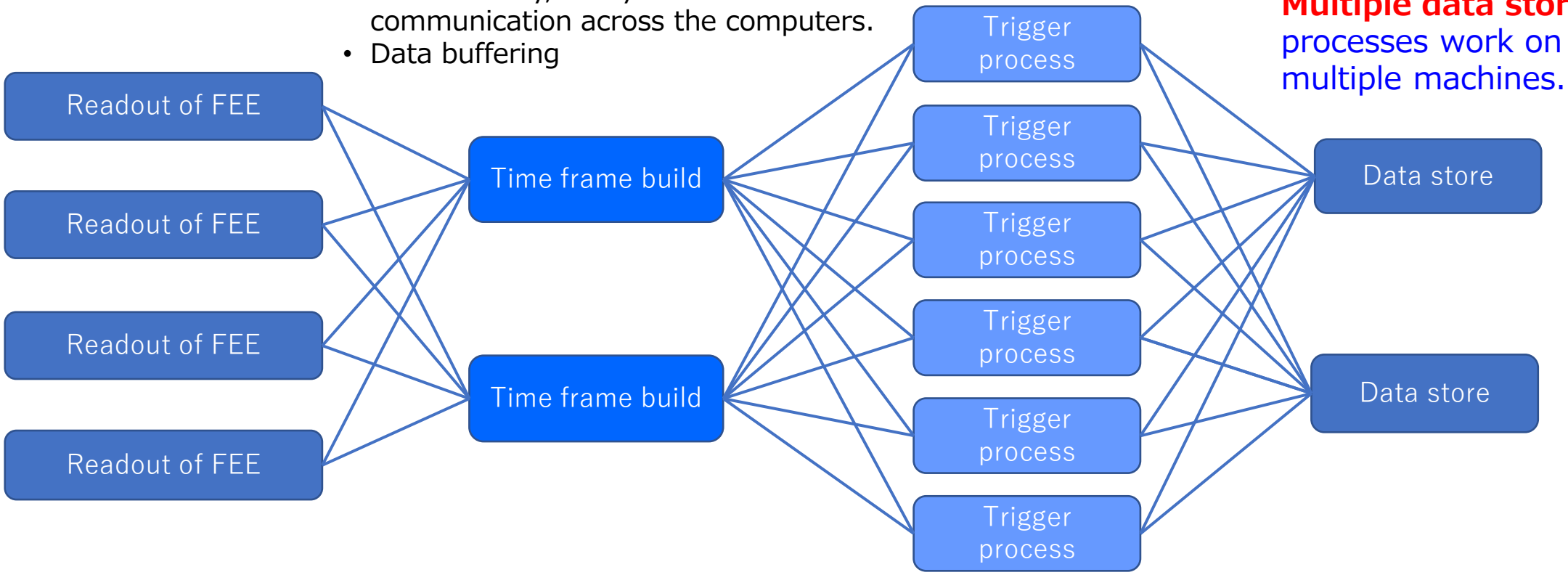
- Overall management
- Overall control

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

- How do the processes communicate to other many processes
  → ZeroMQ
- How do we configure the state machine and how do we control state?
  → FairMQ
- How do we manage a large number of processes
  → Key-Value database: redis

→ What if we combine FairMQ and redis ?

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

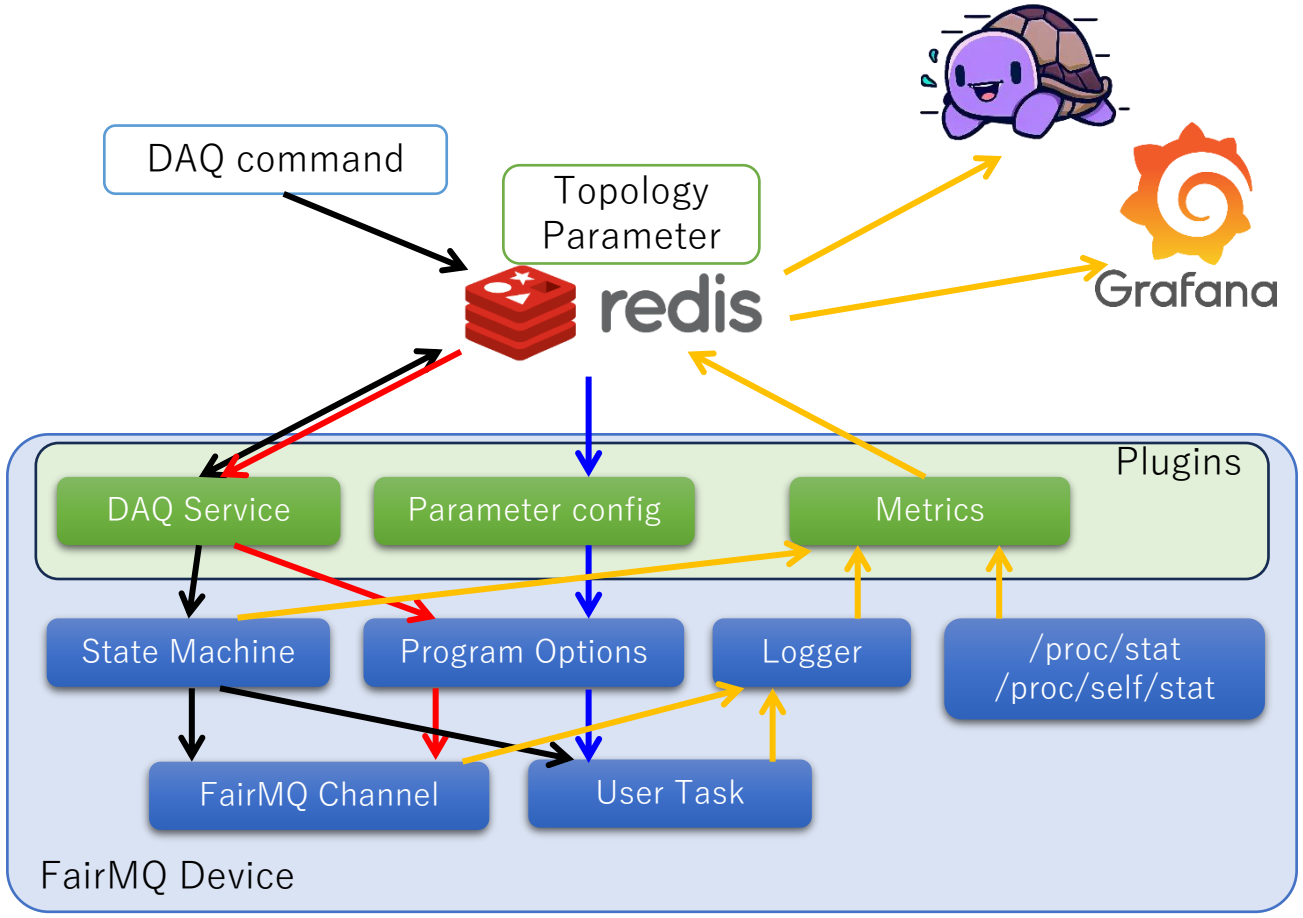
- We employed the state machine and controlled system from FairMQ.
- We used “redis” for the overall management and control.
  - NoSQL database / Key-Vale type
  - Memory-oriented and fast response
  - Key-space notification → It can be used for control.

NestDAQ Process structure

- Redis NoSQL/Key-Value database is used to manage and control the status of all DAQ processes.
- 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

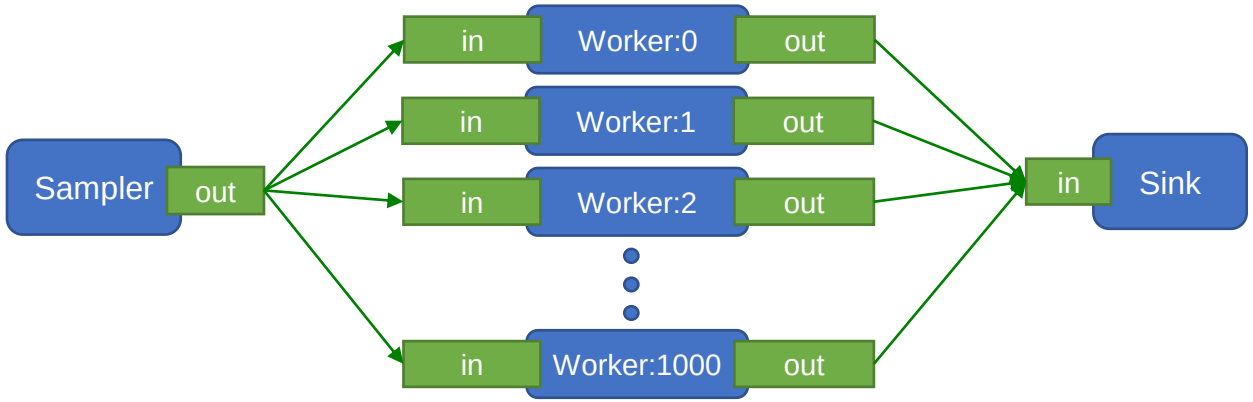
- It’s not realistic to hand write a connection table for over the 1000 connections.
  - → Semi-automatic connection configuration
    - The database provides information about each process grouped as a function (service), its data channel-ports and their connections

Example: An arbitrary number of worker processes

Topology data on the database

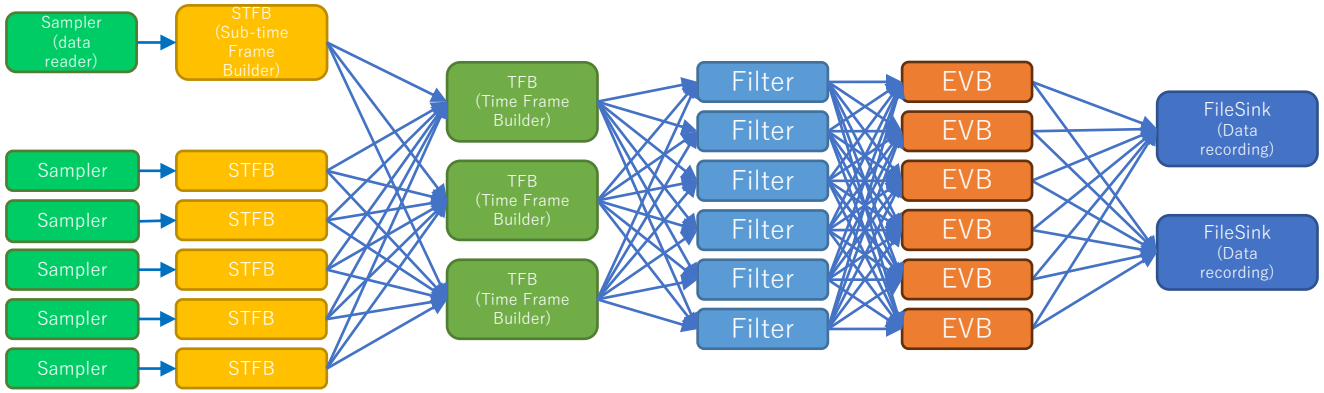
```
#-------------------------------------------------------------------
#           service      channel       options
#-------------------------------------------------------------------
endpoint     Sampler      out           type push  method bind
endpoint     Sink         in            type pull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



DAQ process configuration and connection

Common DAQ configuration



3 stage N column parallel configuration

- Sampler
  - Reading data from streaming front-end electronics
  - The data have hart-beat Frames (HBF) to separate time period. The current period of HBF is 524 us.
- Sub-Time Frame Builder
  - The data from the sampler is cut out for each HBF, and several of them are put together to make a Sub-Time Frame.
- Time Frame Builder
  - Making Time Frame combined from Sub-Time Frame data from each Sub-Time Frame Builder
- Filter/Online Trigger
  - Finding the good event in the Time Frames.
- Event builder (for Streaming Read Out)
  - Extracting the data in the time near the found event time.
- FileSink
  - Writing received data to the file.

General logic online triggering filter by LUT

- Behavior of a general combination logic filter
  - Create a logic table by calculation before the RUNNIG state.
  1. Low-resolution TDC / High-resolution TDC → 4ns TDC
  2. Check and add marks to an array of HBF (524us/4ns) length.
  3. Scan the array and picking up the array index where the LUT returns true.
  4. Store the index where the value changes to a vector. (edge detection)

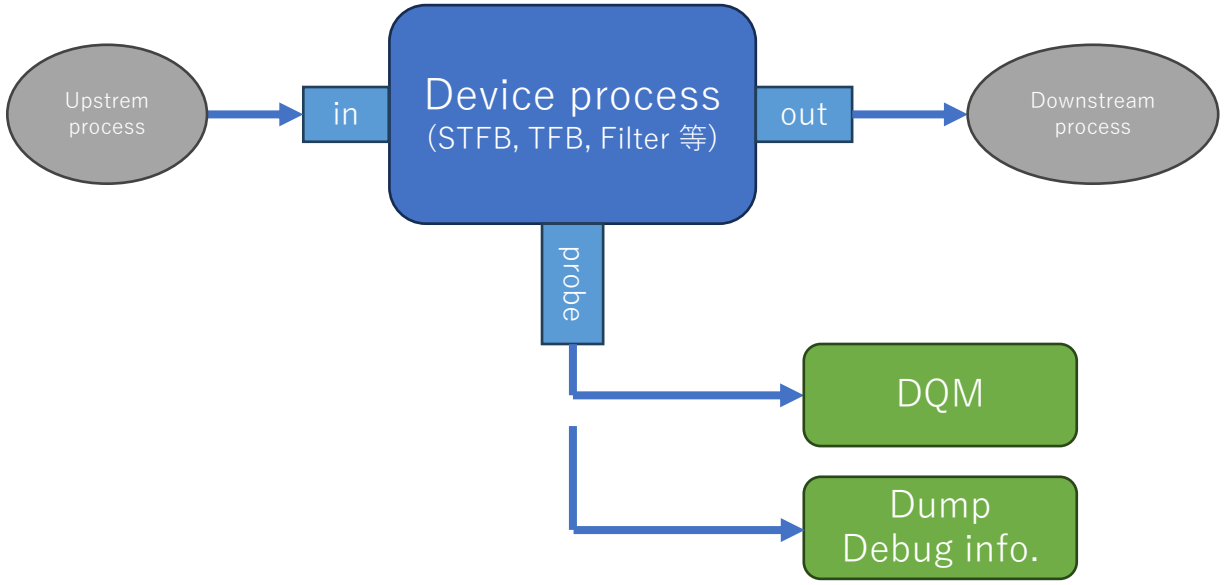
```
fTrig->SetTimeRegion(1024 * 128); ← 1 HBF
fTrig->ClearEntry();

fTrig->SetMarkLen(10); ← 4ns * 10 (coincidence time width)
fTrig->Entry(0xc0a802a9,  0, 0); //DR
fTrig->Entry(0xc0a802a9,  1, 0); //DL
fTrig->Entry(0xc0a802a9,  2, 0); //DR
fTrig->Entry(0xc0a802a9,  3, 0); //DL
fTrig->Entry(0xc0a802a9,  4, 0); //DR
fTrig->Entry(0xc0a802a9,  5, 0); //DL

fTrig->Entry(0xc0a802aa, 32, 0); //UR
fTrig->Entry(0xc0a802aa, 33, 0); //UL
fTrig->Entry(0xc0a802aa, 34, 0); //UR
fTrig->Entry(0xc0a802aa, 35, 0); //UL

fTrig->MakeTable("0 1 & 2 3 & | 4 5 & | 6 7 & 8 9 & | &");
```

Module	Ch.	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
AMANEQ1	Ch.00		■	■	■												
AMANEQ1	Ch.01			■	■	■						■	■	■			
AMANEQ1	Ch.02				■	■	■										
AMANEQ1	Ch.03			■	■	■											
	…																
	Ch.35				■	■	■										

Just a little ingenuities

Probe port

- All DAQ processes (Device process) have a probe port that provides the same data as the out port.
  - Grab the data from any process while the debugging and understanding the situation.
  - The data can be streamed separately.
    - It can be used for DQM/Online monitor.



File Replayer

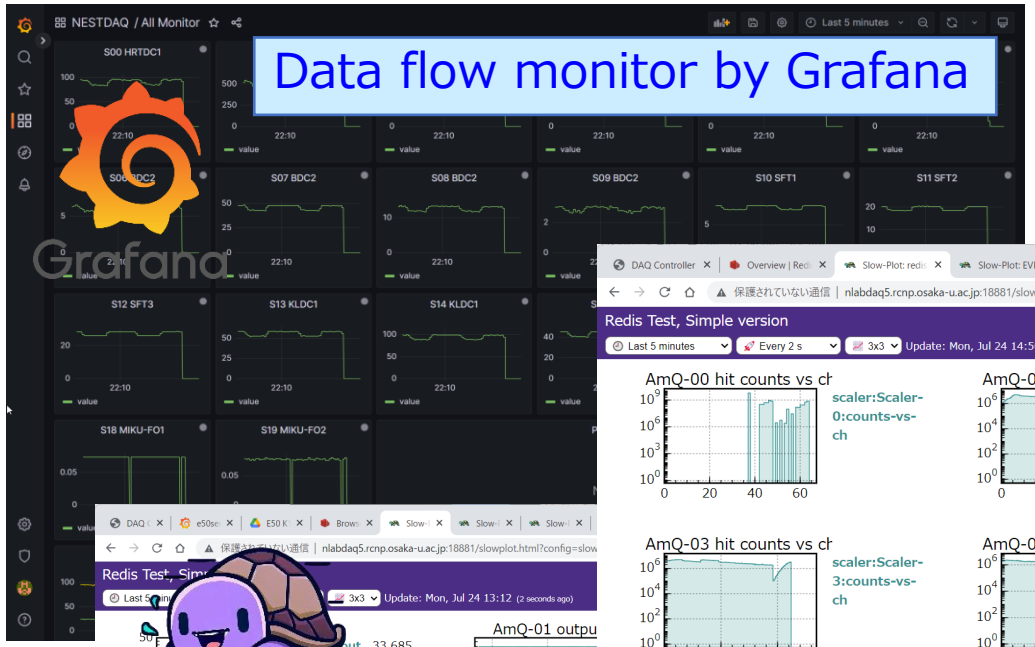
- File Replayer can read data files and process the data again in the same sequence as it was taken.

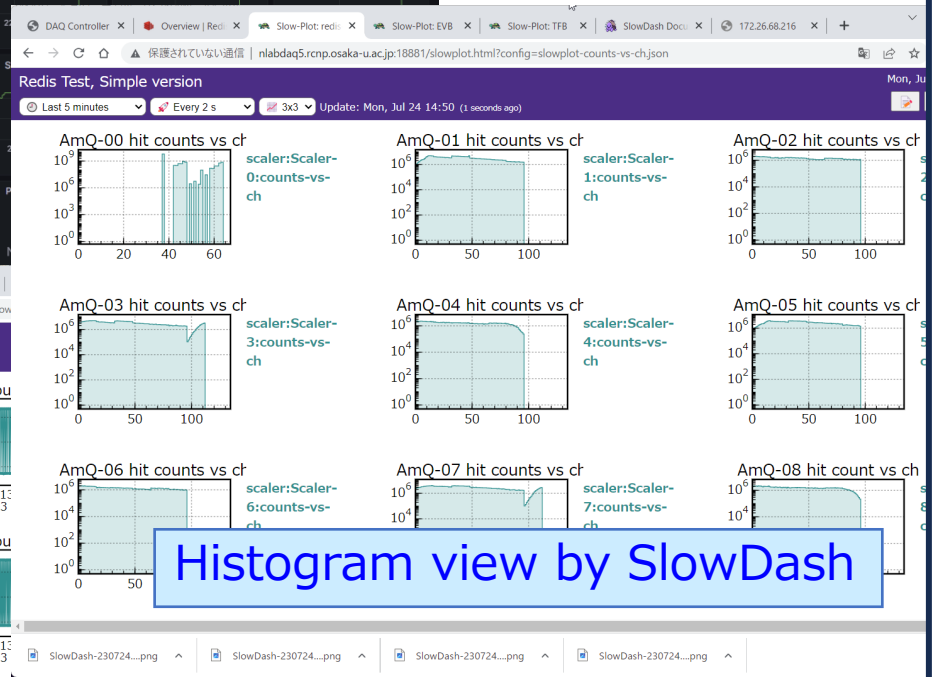
Development using TFB File player and STFB File player

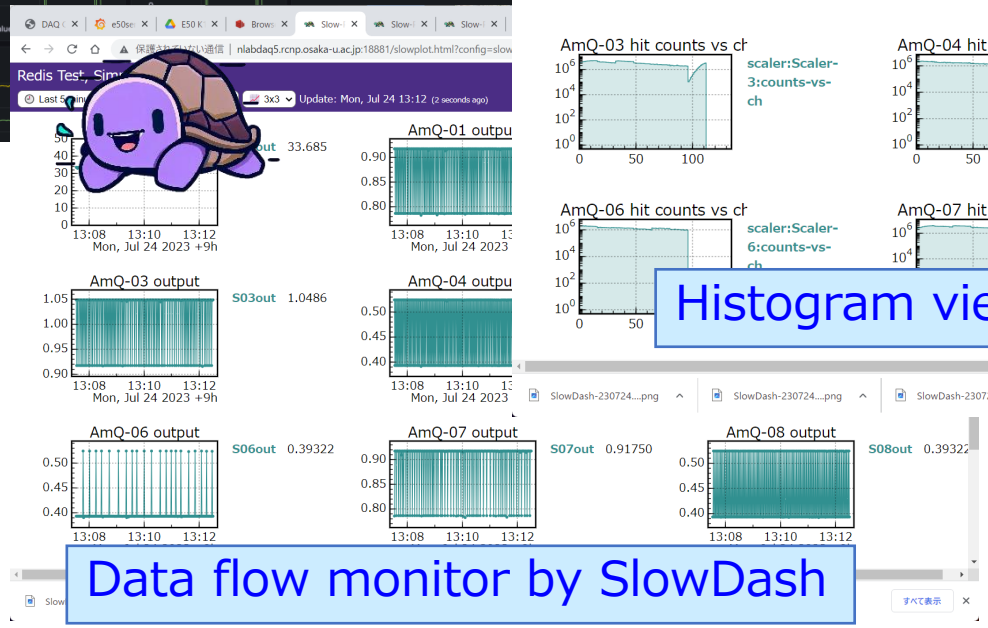
Development of a filter program

Development of a Time-Frame-Builder and a filter program

Web UI







DAQ control

DAQ controller

RUN number

New value: Send +1 Get Auto increment at RUN Stop
Next:
Least:
Start:
Stop:

State transition command

Idle ⊳ Running

Idle ⊳ Init Device and Connection Device Ready ⊳ Init Task ⊳ Ready ⊳ Run ⊳ Running

Idle ⊲ Running

Idle ⊲ Reset Device ⊲ Device Ready ⊲ Reset Task ⊲ Ready ⊲ Stop ⊲ Running

⊳ Exit

Any state ⊳ End ⊳ Exiting

State Summary

Service	N	Undefined	Ok	Error	Idle	Init-Device	Initialized	Binding	Bound	Connecting	Device-Ready	Init-Task	Ready	Running	Reset-Task	Reset-Device	Exiting	last-update
AmQStrTdcSampler	10												10					2023-03-03T15:16:29
STFBuilder	10												10					2023-03-03T15:16:27
TimeFrameBuilder	3												3					2023-03-03T15:16:28
fltcoin	16												16					2023-03-03T15:16:28
tfdump	1												1					2023-03-03T15:16:27

Show details

Service	Instance	state	last-update

Select command target

Choose Services: all, AmQStrTdcSampler, STFBuilder, TimeFrameBuilder, fltcoin, tfdump

Choose Instances: all, AmQStrTdcSampler:AmQStrTdcSampler-0, AmQStrTdcSampler:AmQStrTdcSampler-1, AmQStrTdcSampler:AmQStrTdcSampler-2, AmQStrTdcSampler:AmQStrTdcSampler-3, AmQStrTdcSampler:AmQStrTdcSampler-4, AmQStrTdcSampler:AmQStrTdcSampler-5, AmQStrTdcSampler:AmQStrTdcSampler-6, AmQStrTdcSampler:AmQStrTdcSampler-7

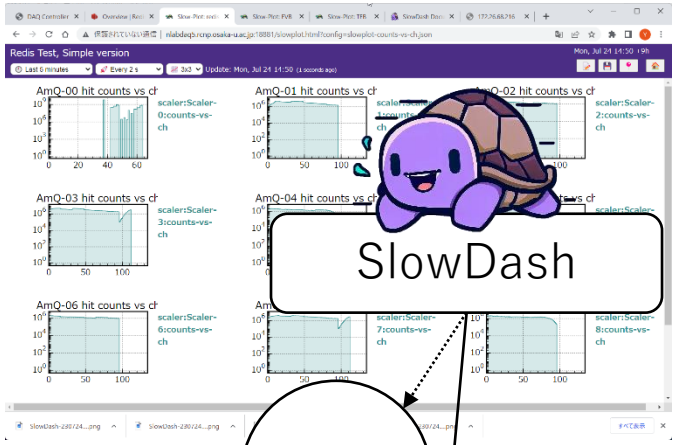
My WebSocket Connection ID: 2 (Date: 2023-03-03 15:06:08)

WebSocket Connected ID: Date
2 : 2023-03-03 15:06:08

* SlowDash is a web based visualization tool developed by S. Enomoto (Washington U.)

Mini booking tool for online display

- SlowDash is a visualizer on the web browser.
  - It is developed by S. Enomoto, Wasinton U.
  - It can display trends and histograms from the values on the database.
- Display histogram to Slowdash from DAQ processes
  - Similar interface of TH1, TH2 of ROOT.
    - Mini booking tool: uhbook
    - Converting function to JSON: Slowdashify()
    - Data store function to the redis DB: RedisDataStore




```
Title: Hello
Entry:      100
Over flow:  0
Under flow: 0
0.0e+00:0.0e+00|
6.7e+00:0.0e+00|
1.3e+01:0.0e+00|
2.0e+01:0.0e+00|
2.7e+01:0.0e+00|
3.3e+01:0.0e+00|
4.0e+01:1.0e+00|#
4.7e+01:1.0e+00|#
5.3e+01:1.0e+00|#
6.0e+01:3.0e+00|###
6.7e+01:0.0e+00|
7.3e+01:9.0e+00|#########
8.0e+01:8.0e+00|########
8.7e+01:1.8e+01|##################
9.3e+01:1.3e+01|#############
1.0e+02:1.0e+01|##########
1.1e+02:1.5e+01|###############
1.1e+02:8.0e+00|########
1.2e+02:4.0e+00|####
1.3e+02:6.0e+00|######
1.3e+02:1.0e+00|#
1.4e+02:2.0e+00|##
1.5e+02:0.0e+00|
1.5e+02:0.0e+00|
1.6e+02:0.0e+00|
1.7e+02:0.0e+00|
1.7e+02:0.0e+00|
1.8e+02:0.0e+00|
1.9e+02:0.0e+00|
1.9e+02:0.0e+00|
```

```
Title: Hello 2D
Entry: 10000
Over/Under flow:       0      0      0
                :       0  10000      0
                :       0      0      0

5.0e+01|
5.5e+01|                  . .
6.0e+01|                .... .. .
6.5e+01|              . ......-...
7.0e+01|              ...-+++----.
7.5e+01|              .---xxx+++--.
8.0e+01|            ..--+x*x@x*+--..
8.5e+01|            ..-++x@@@@x++-..
9.0e+01|           ..-+x*@@##*@xx+-.
9.5e+01|            .--x**/####*++...
1.0e+02|            ..-+*@####@*+-..
1.0e+02|            .--xx@@##@xxx+-
1.1e+02|            .--+x*@@@@*+++.
1.2e+02|             .--+++x*x++x+-..
1.2e+02|            ..--+-+xxx+++..
1.2e+02|              ...--++--...
1.3e+02|               .----.....
1.4e+02|               .  ...
1.4e+02|
1.4e+02|
1.5e+02|
```

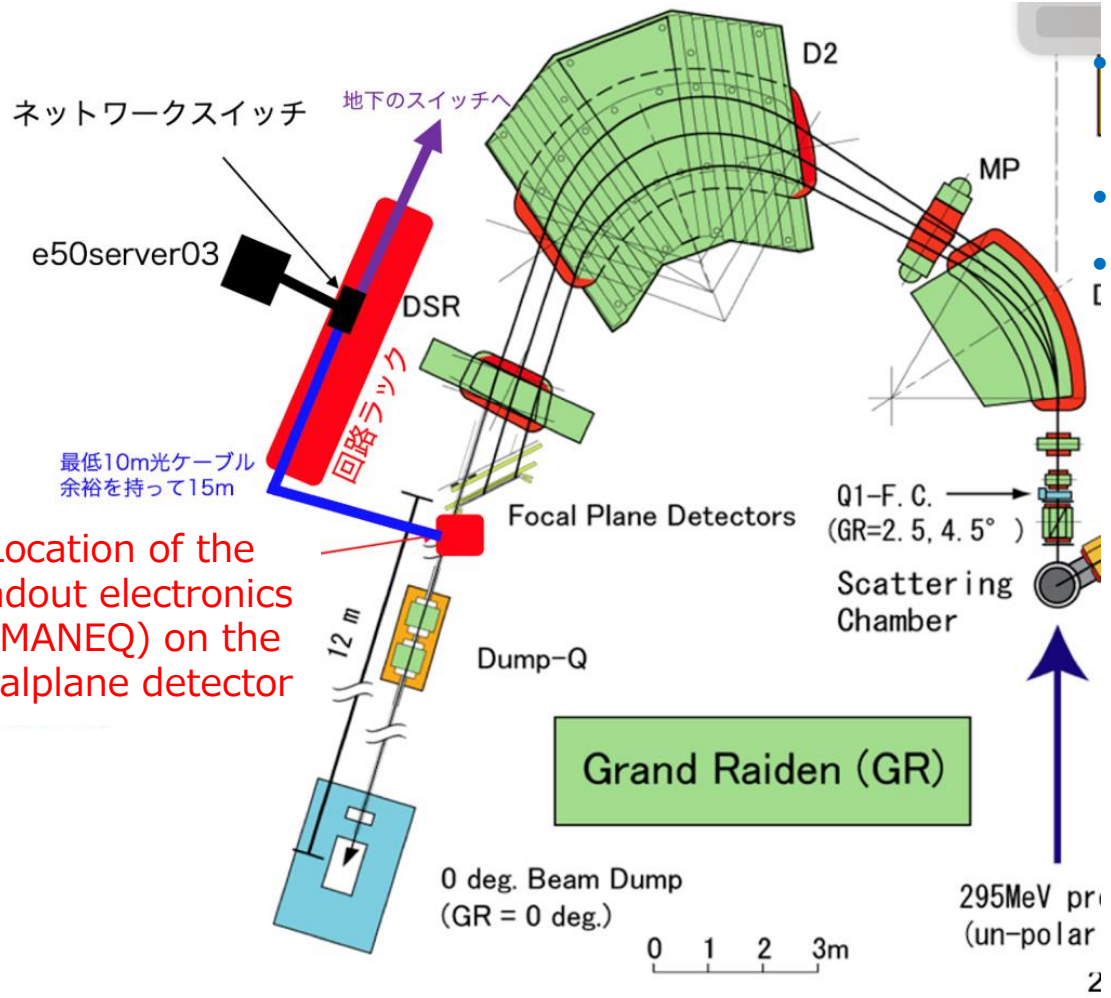
Histogram object

```
{
        "bins": { "min": 0, "max": 200 },
        "counts": [
           0, 0, 0, 0, 0, 20, 0, 1, 2, 2,
           3, 5, 10, 13, 12, 13, 15, 11, 8, 2,
           1, 0, 0, 2, 0, 0, 0, 0, 0, 0 ]
}
```

JSON

First trial to apply a streaming DAQ to a actual detector

RCNP Grand Raiden spectrometer



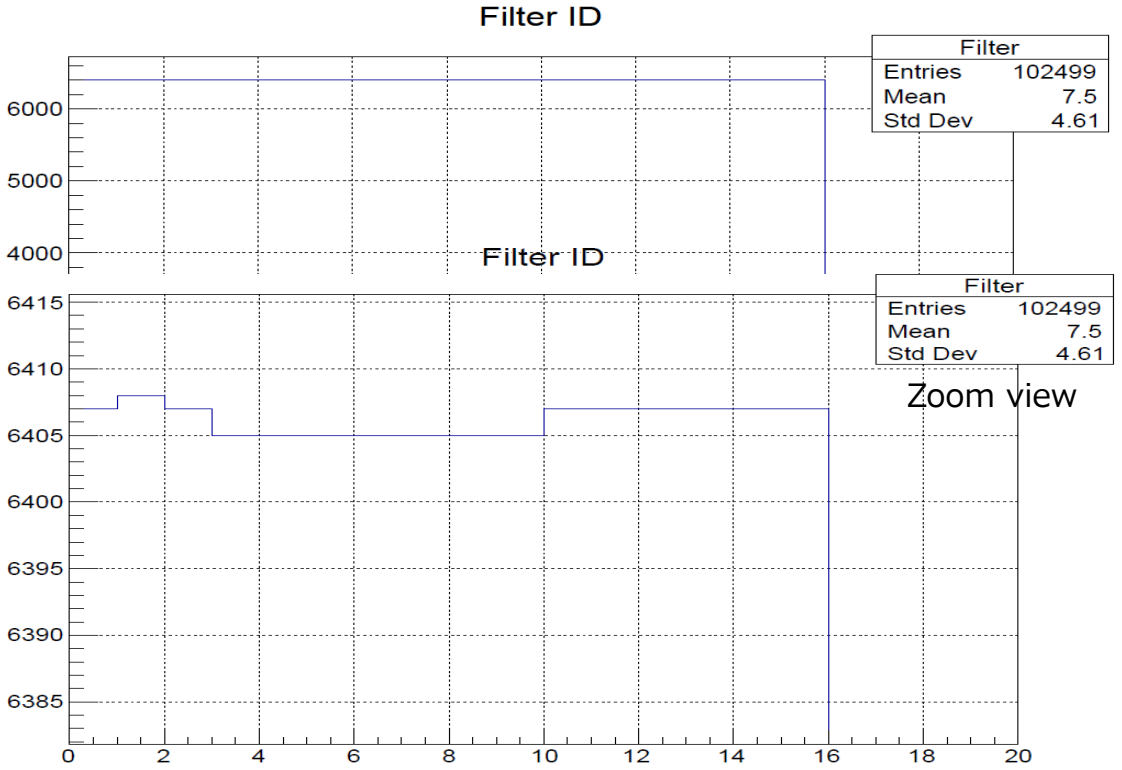
Location of the readout electronics (AMANEQ) on the focalplane detector

- 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 + α) “NestDAQ”
  → Confirmation of the streaming DAQ

2023/03

Behavior of the filter process

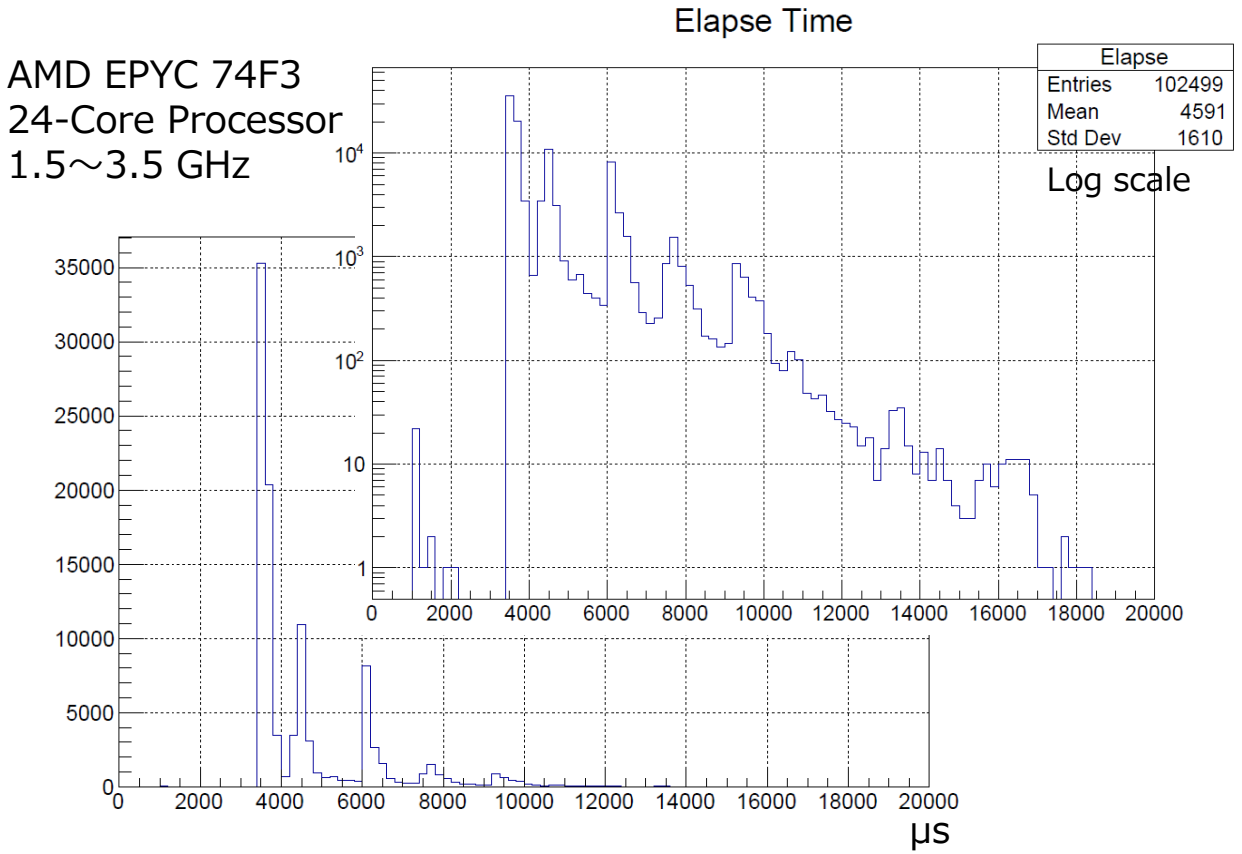
Load distribution



Filter 16 process

The load is distributed moderately by Round-Robin + Skip at Queue-Full algorism.

5HBF consumption duration (w/o data transfer)



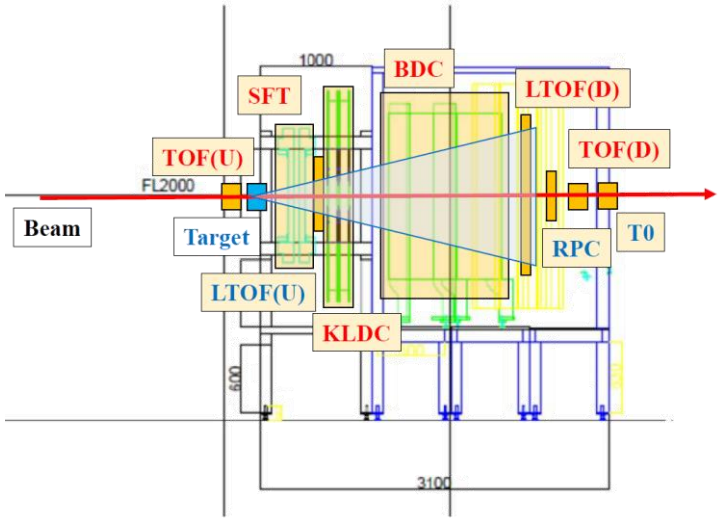
Trigger: Ch1 * Ch2 * Ch3 * Ch4

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

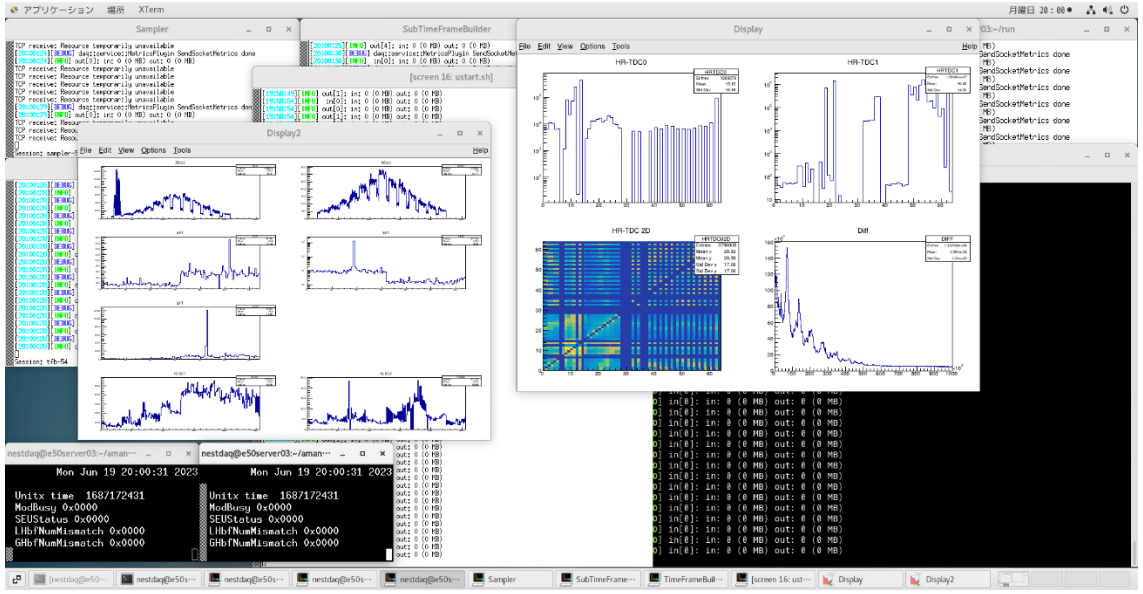
→ It is possible to process to use more than 2 processes because the average consumption time is 4.5 ms.

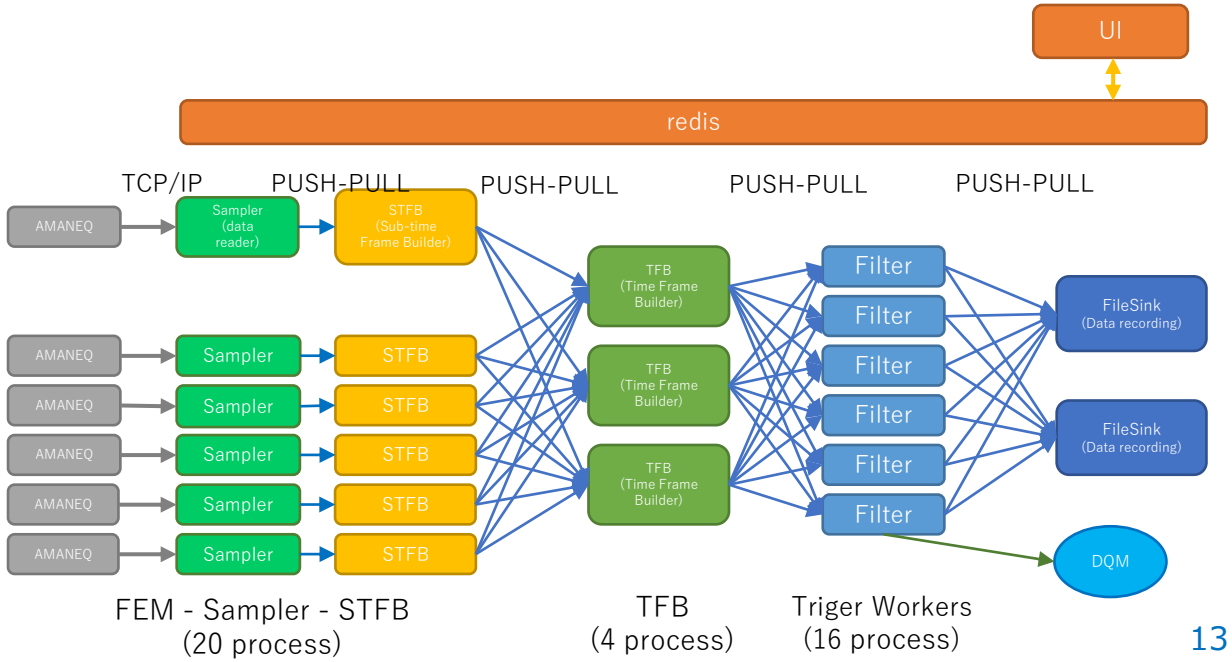
E50 detector test in J-PARC HD K1.8BR

- Front-end electronics, number of channels
  - HRTDC x2 : 128 channel
  - LRTDC x15 : 1920 channel
  - MIKUMARI x3 : 64 channel
- Combinational logic trigger process by LUT
- Data Quality Monitor using PUB/SUB communication by the Probe port
- Data flow (recording):
  - ～180MB/s (average of Flat Top ON/OFF)
  - ～240MB/s (at Flat Top ON)



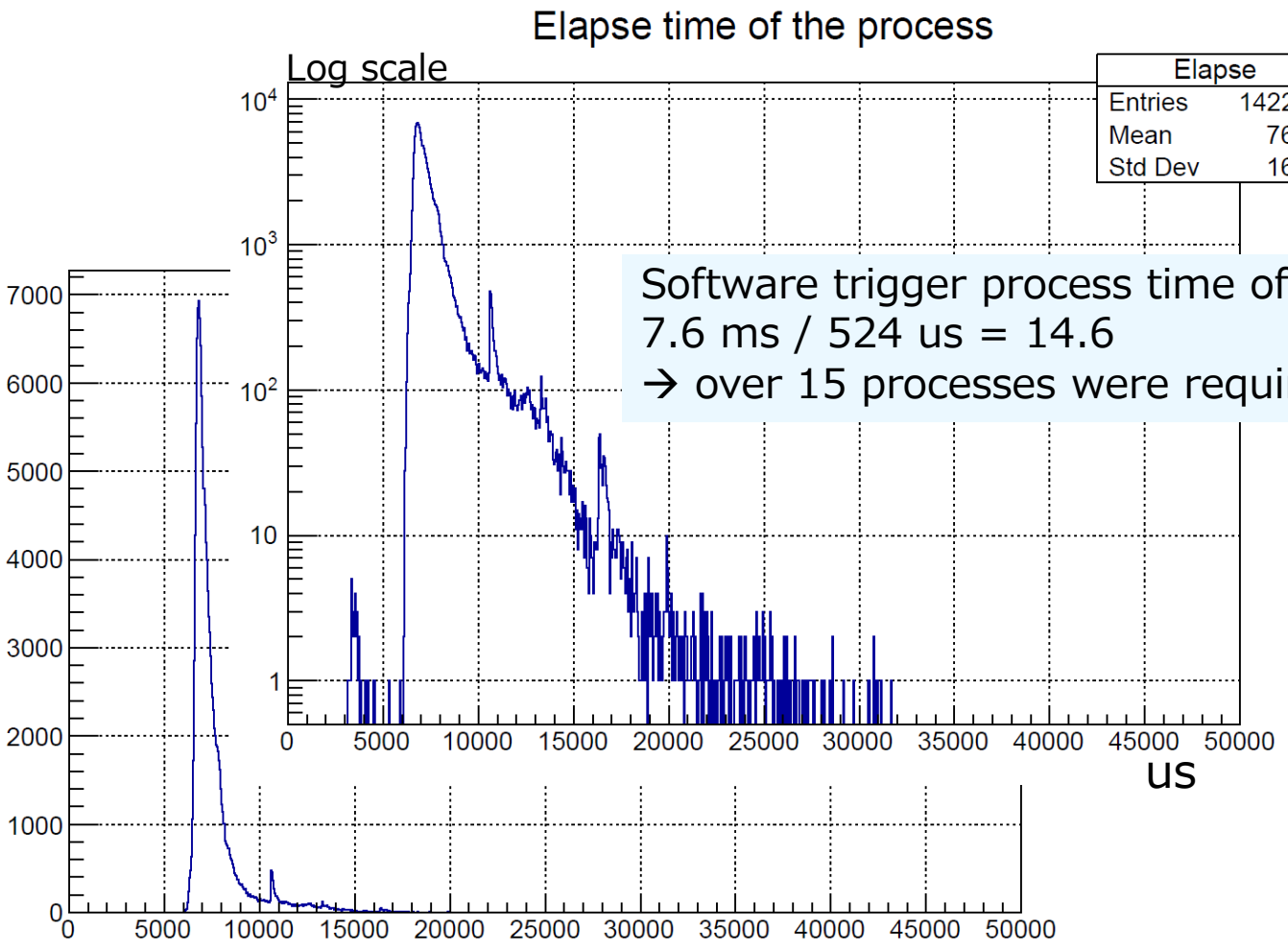
2023/06



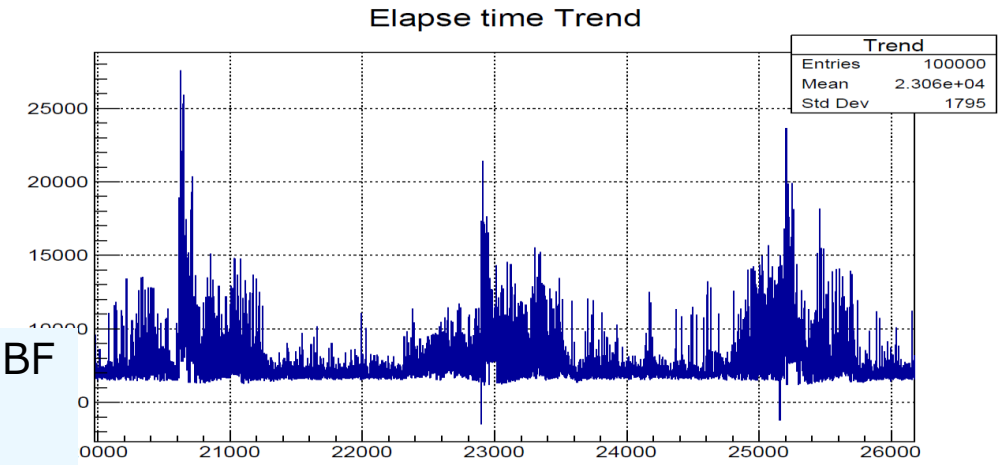


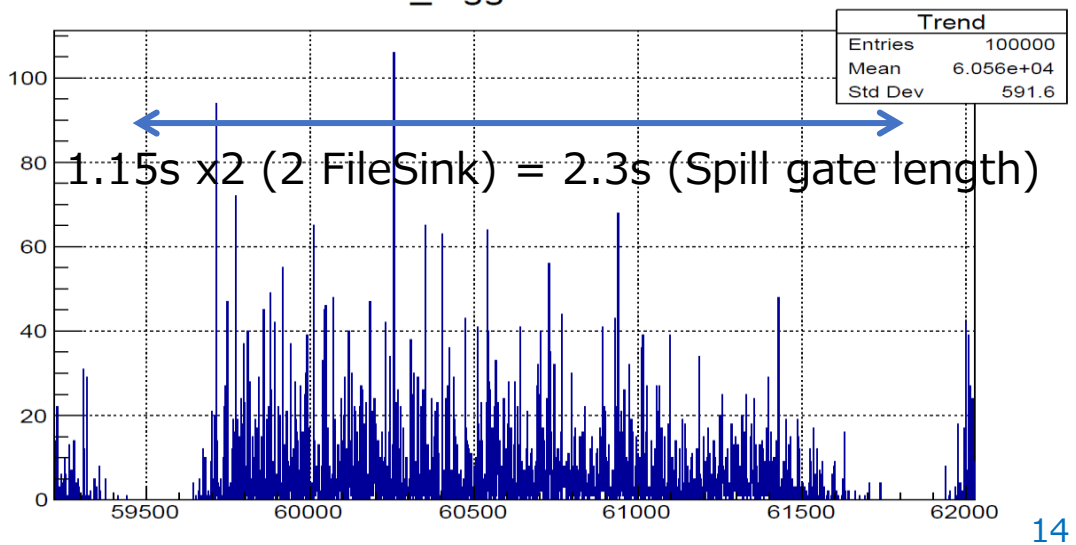
Elapsed process time and counting trends

- 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



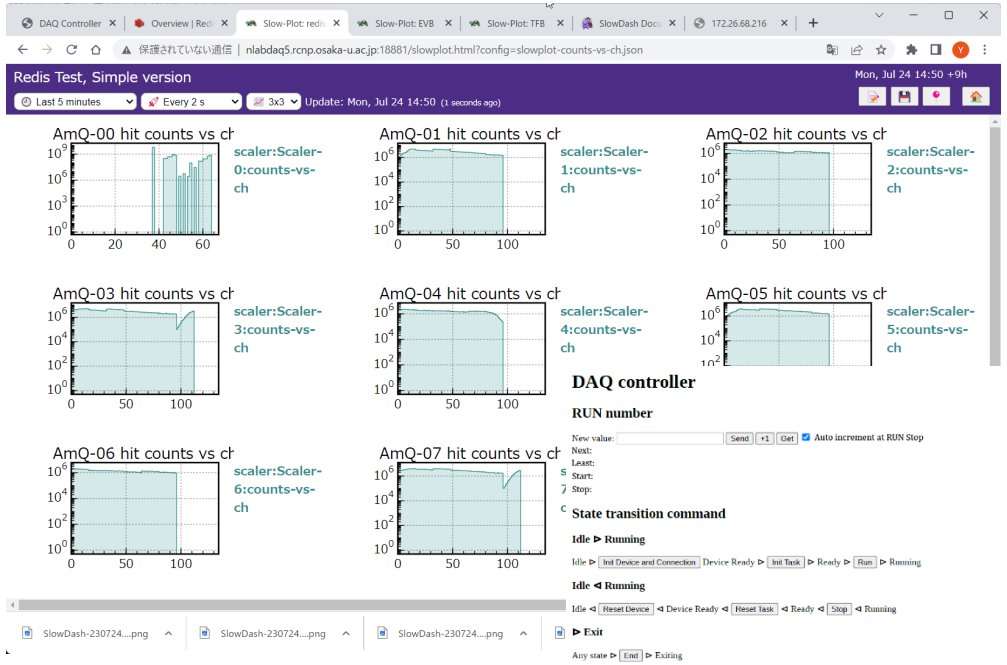


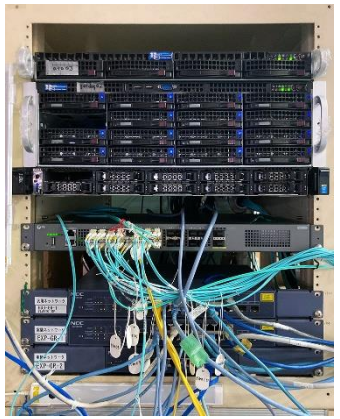
RCNP GR/WS E585

Appling a real physics data taking

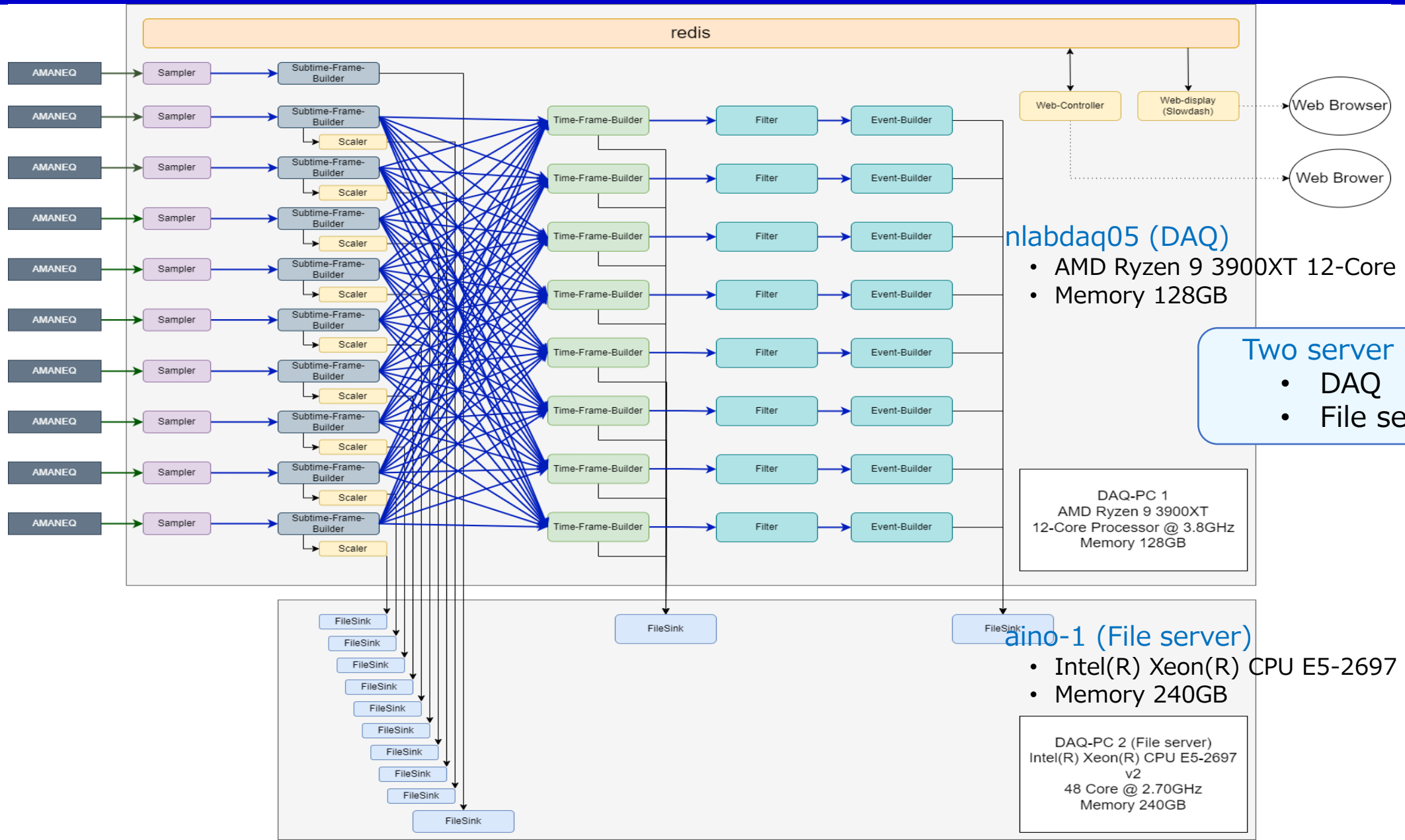
- General logic trigger filter
- Event builder (for SRO)
- Recording pre-scaled unbiased data
- Software scaler

- Web UI update
  - Auto inclement Run number
  - Run a device control script when the run starts and stops
- Online monitor by SlowDash
  - Data flow visualization
  - Software scaler
  - Issue flag display





RCNP GR/WS E585 software configuration



nlabdaq05 (DAQ)

- AMD Ryzen 9 3900XT 12-Core Processor @ 3.8GHz
- Memory 128GB

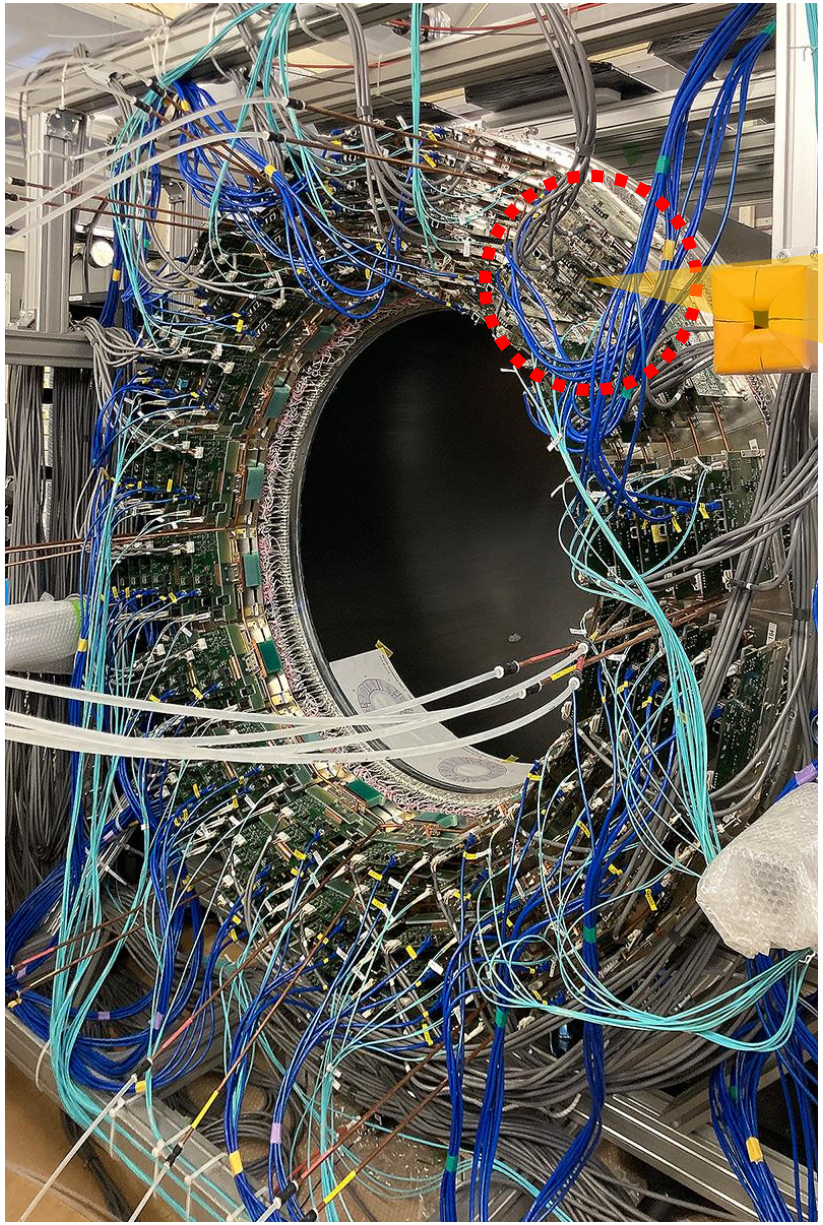
Two server PC configuration

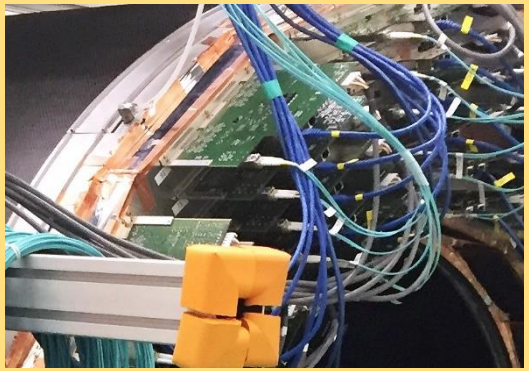
- DAQ
- File server

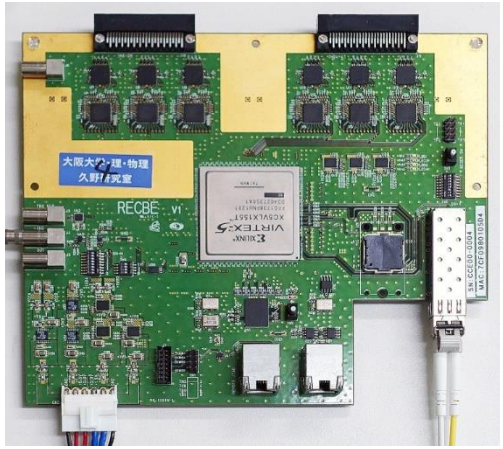
aino-1 (File server)

- Intel(R) Xeon(R) CPU E5-2697 v2 48 Core @ 2.70GHz
- Memory 240GB

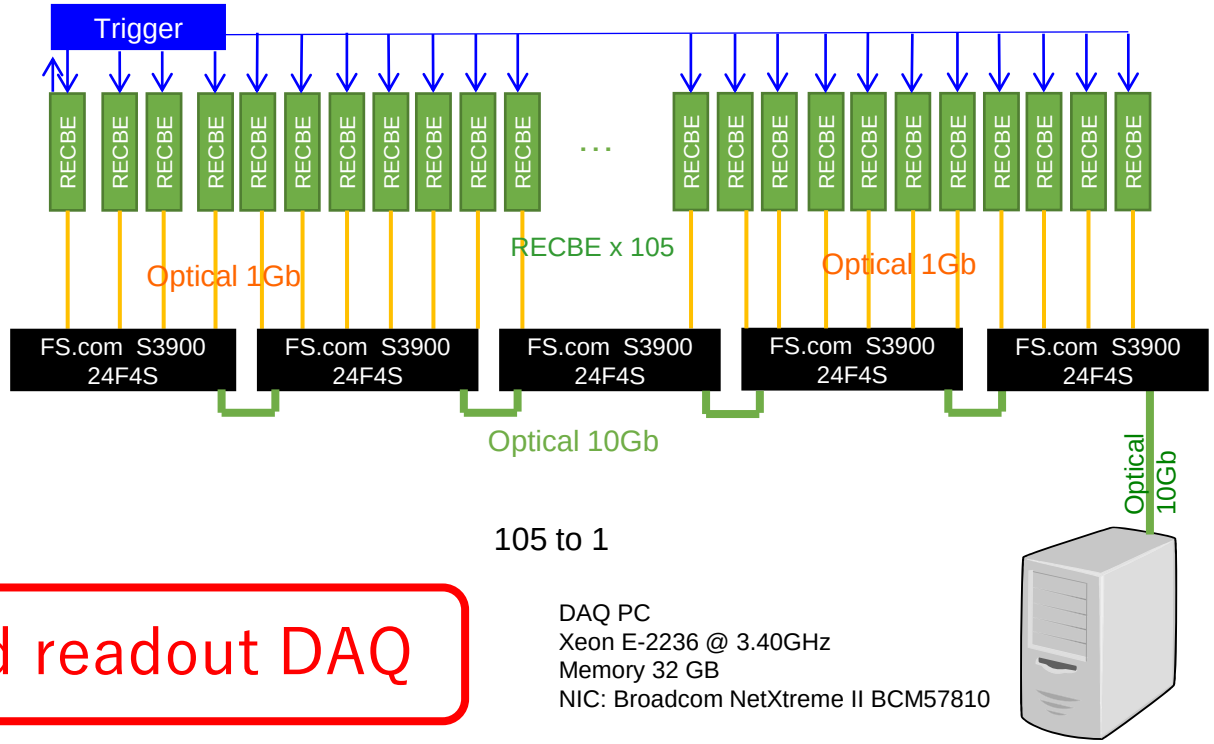
Trial to read a triggered DAQ (COMET CDC)







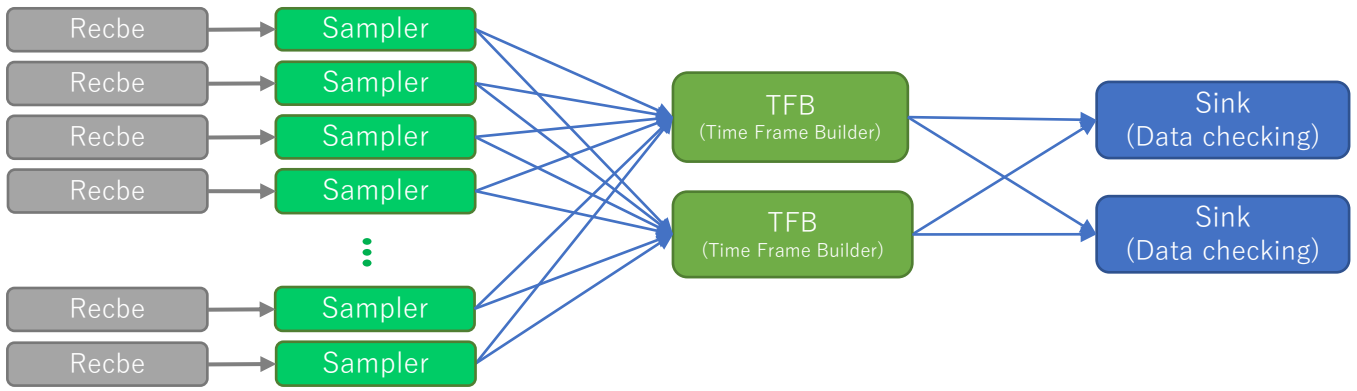
"Recbe" is a 48 channel read-out card for drift chamber.



Triggered readout DAQ

DAQ PC
Xeon E-2236 @ 3.40GHz
Memory 32 GB
NIC: Broadcom NetXtreme II BCM57810

Data flow throughput



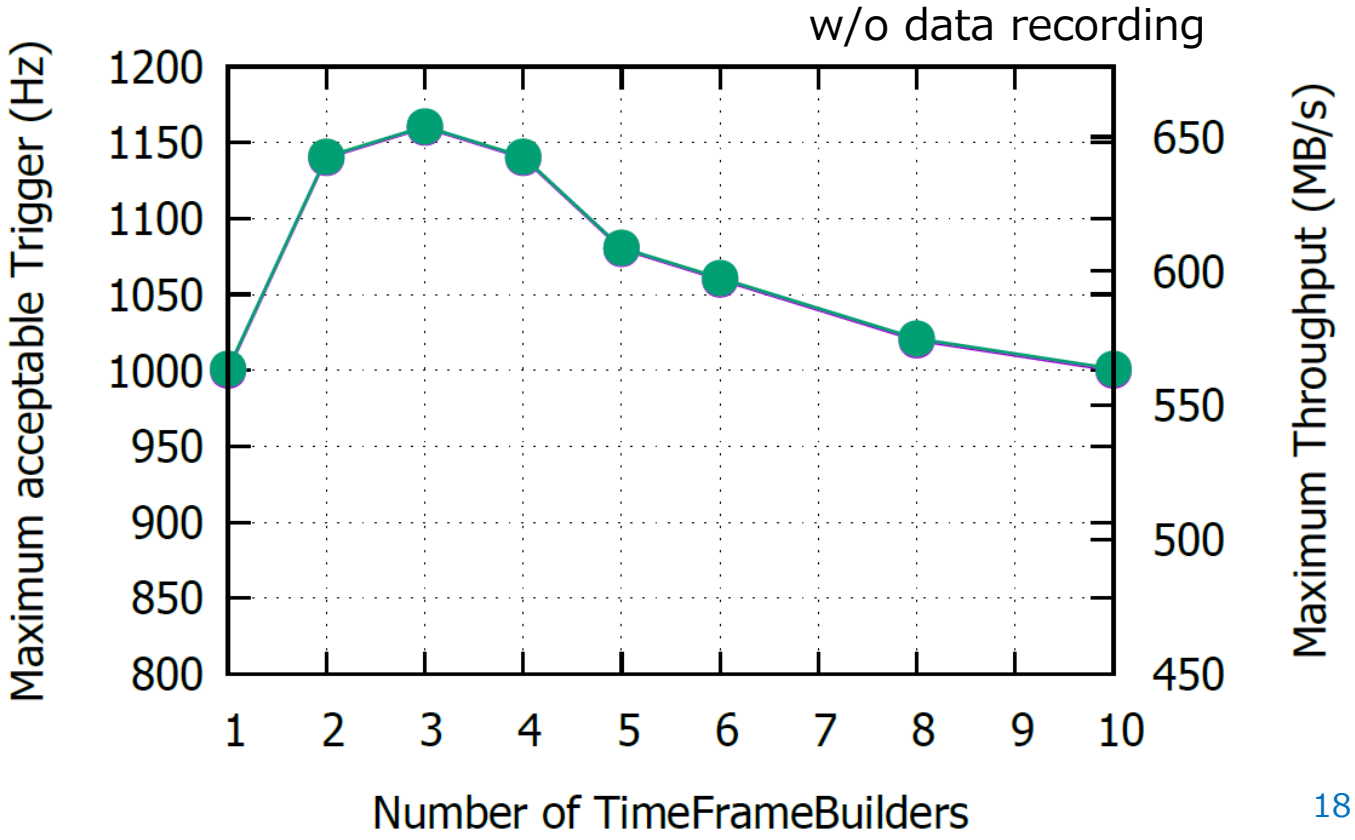
96 FEE and 96 Samplers

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

UDS was used for internal process communication instead of TCP/IP.

Using One DAQ PC

- Event size: 6156B
- Number of Recbes: 96
- DAQ PC
  - Xeon E-2236 @ 3.40GHz 6 Cores
  - Memory 32 GB
  - NIC: Broadcom NetXtreme II BCM57810
- 1G/10G network switch
  - FS.com S3900 24F4S



Summary

- A development of a streaming capable DAQ software framework "NestDAQ" based on FairMQ and redis is in progress.
- We have some experience with DAQ using this framework.
  - The DAQ framework works well (if you just want to take 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
    - We would like to manage logs from many processes.
    - We are evaluating popular log collectors like Fluentbit.
  - Centralized management and control of global DAQ state
    - Control the order of state changes in each process
  - Organized data handling for inter-process communication.
  - More effective and advanced online trigger filters
    - Tracking, GPU processing, …
  - Good UI…