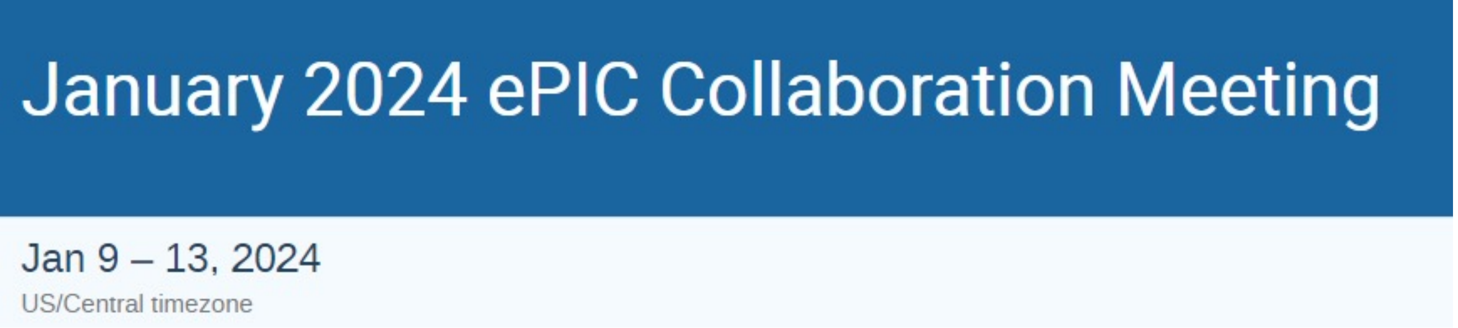


January 2024 ePIC Collaboration Meeting

Jan 9 – 13, 2024

US/Central timezone

Possible contributions from SPADI-Alliance in Japan

TAKU GUNJI

CENTER FOR NUCLEAR STUDY

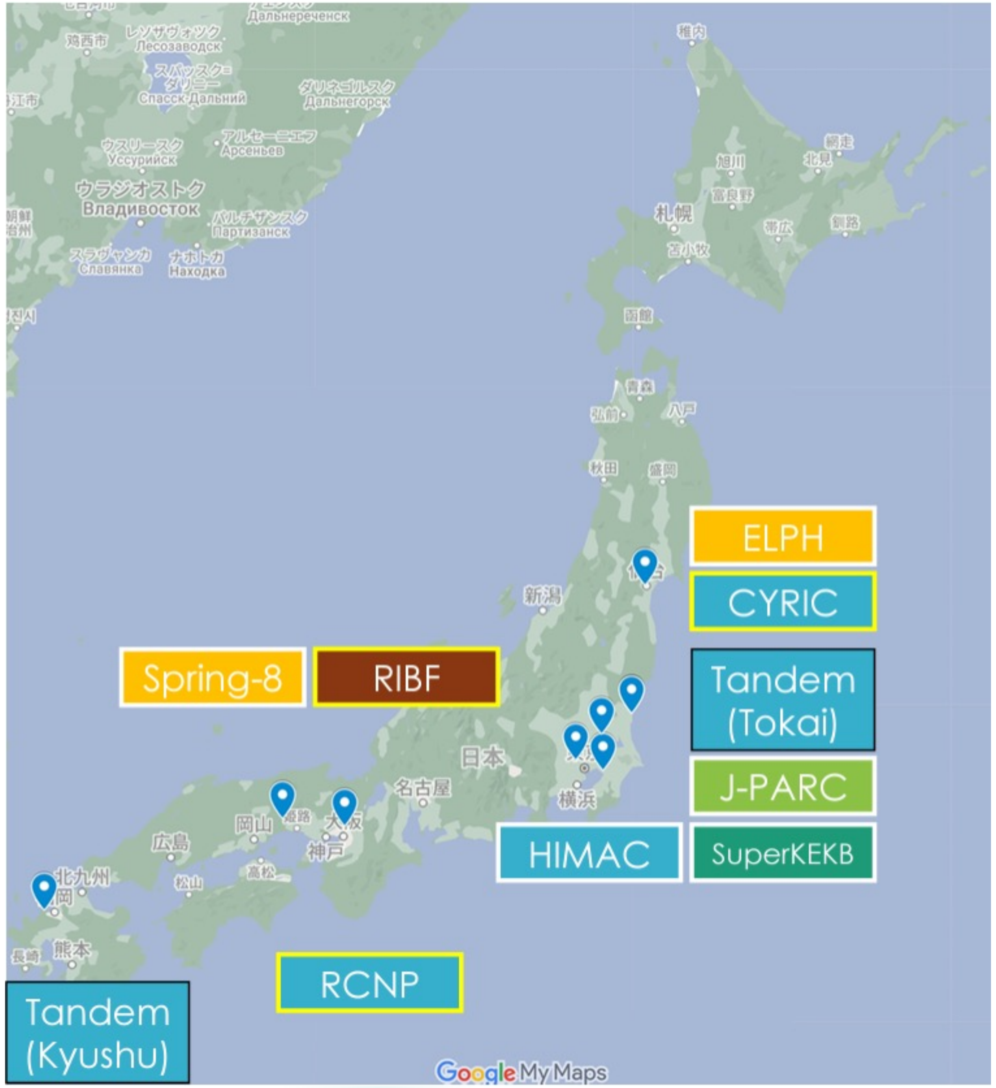
THE UNIVERSITY OF TOKYO

SPADI-Alliance

SPADI Alliance

Signal processing and data acquisition infrastructure alliance

- Streaming DAQ are becoming common needs in NP community
  - RIBF, RCNP, J-PARC, …
- SPADI-Alliance was established
  - For standardization of SRO DAQ in Japan
  - >120 researchers and 20 institutes from different experiments and from different facilities
- Synergies with EIC, ALICE, and HEP experiments (Belle2 and ATLAS)



History of SPADI-A

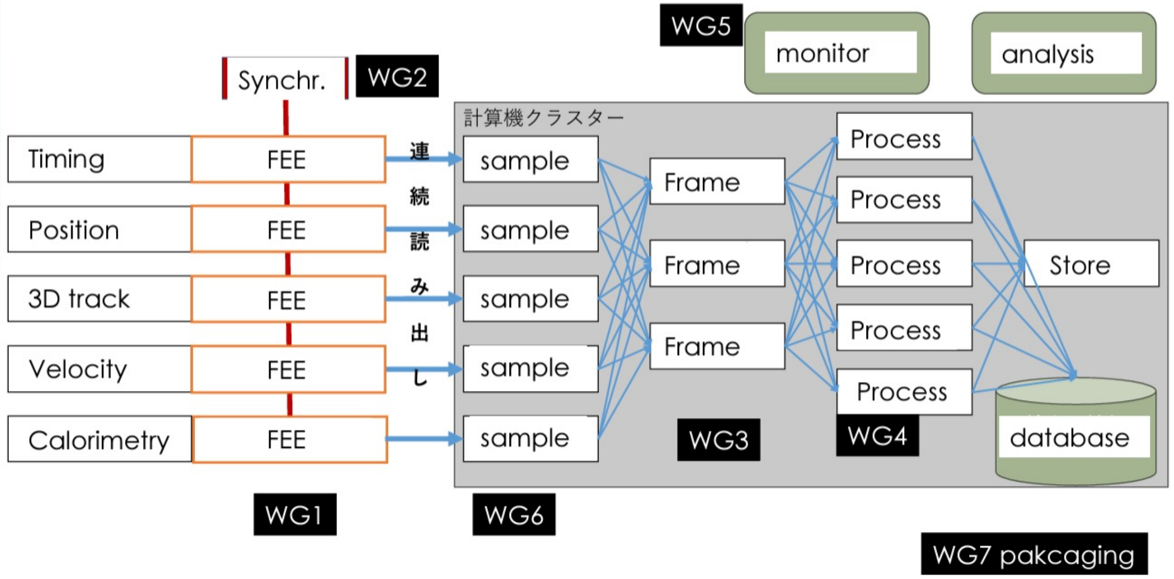
SPADI Alliance

Signal processing and data acquisition infrastructure alliance

- 2022.05 Initiated (by members from RIBF, RCNP, J-PARC, ELPH, and ALICE)
  - FY2022 Discussion for the implementation and FEE developments
- 2023.03 Town meeting in JPS and Annual workshop
- 2023.03 Test Implementation of Streaming-DAQ at RCNP
- 2023.06 Test Implementation of Streaming-DAQ at J-PARC
- 2023.06 Bylaws are issued
- 2023.07 First physics experiment with Streaming-DAQ at RCNP
- 2023.09 Streaming-DAQ implementation in Laboratories as exercise
- 2023.12 DAQ workshop (SPADI, ALICE, EIC/ePIC, sPHENIX) at the APS/JPS meeting
- 2023.12 SRO XI workshop (4 presentations from SPADI-Alliance)
  - Start discussing collaboration with EIC/ePIC SRO-DAQ team

Working Groups

- 7 working groups
  - FEE, timing distribution, streaming SW,
  - Online processing (including hardware accelerators, AI/ML)
  - UI, Computing, Packaging, Analysis SW



WG1
Frontend Electronics
Streaming type
Charge ASD board
Voltage ASD board
WF Digitizer board
Control Firmware dev.

WG2
Clock synch. / Data Transfer
General Clock Synch.
High throughput
Intra-board transfer

WG3
Acquisition software framework (NestDAQ + …)
Streaming type
FairMQ-based Scalable DAQ
Sampling, Time frame build,
Event build, Monitoring…
Format

WG4
Event processing
Acceleration using GPU/FPGA
Zero suppression
Calibration, Clustering,
Tracking, PID,

WG5
User Interface
Control, Monitor, Configure,

Trial with
SlowDash

WG6
Computing infrastr.
High throughput
Large volume
Flow and Archive
Power consumption
Interconnect
Networking
…

WG7
Packaging
Standalone system
Popularization
Standardization
Market research
User feedback

Analysis

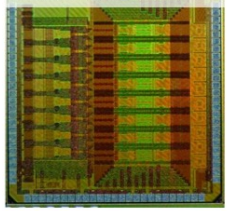
Trial with
Artemis

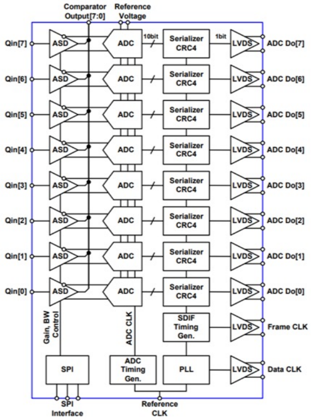
WG1&2 : FEE

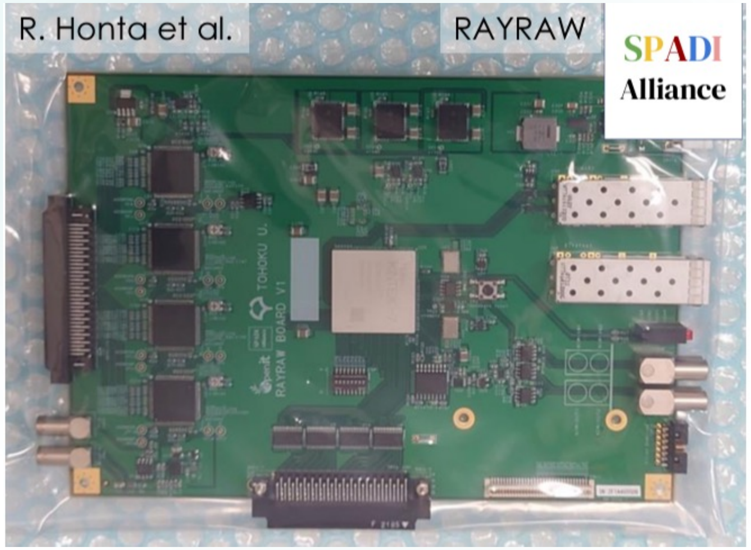
- Many types of FEEs for MPPC, gaseous detectors, Si readout, FADC, TDC, …

MPPC ASIC Board
YAENAMI ASIC (8ch)

65 nm Si CMOS 2 mm角
(Package: LQFP100)

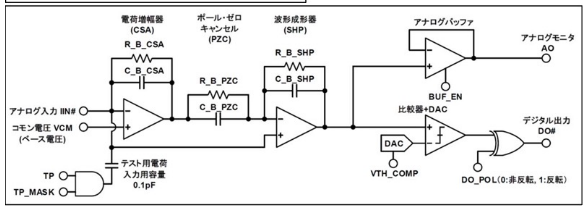


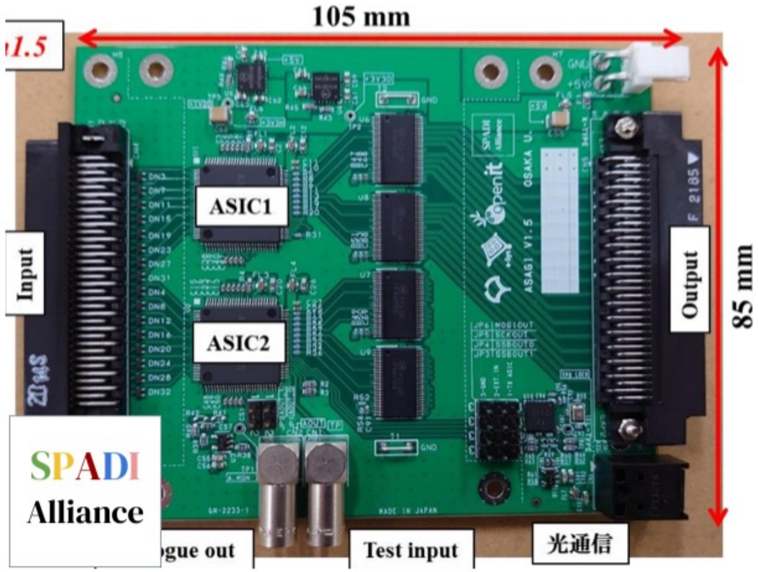




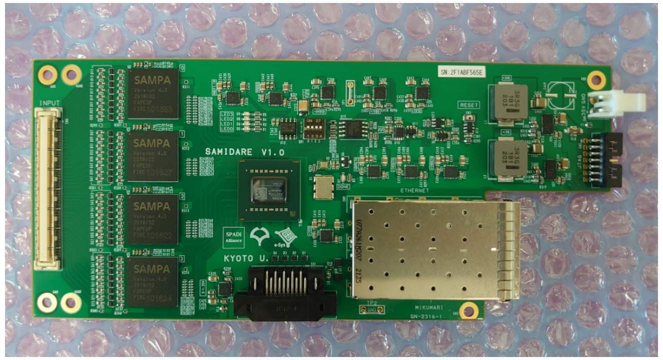
Gas chamber ASD
AGASA ASIC (16ch)

Electric circuit of 1 ASD channel





SAMPA chip board "SAMIDARE"



High resolution FADC MIRA

TDC, Timing distribution (AMANEQ)

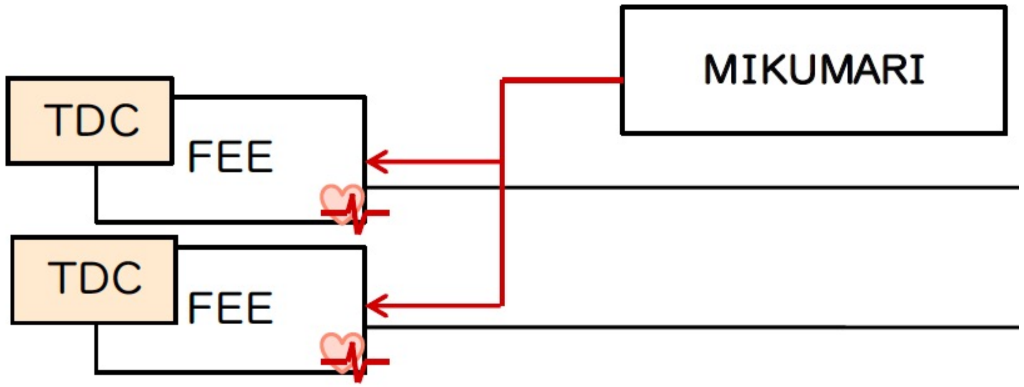
WG1+2: Timing Distribution

Synchronization

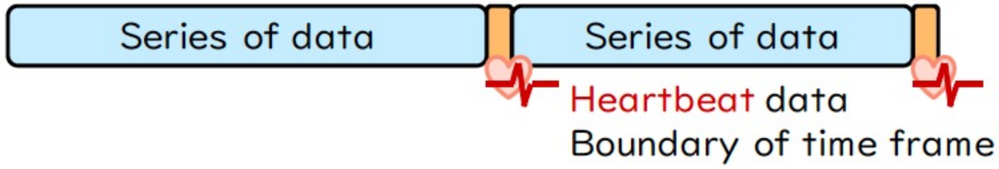
Simple & light-weight clock-data-recovery

Frequency synchronization

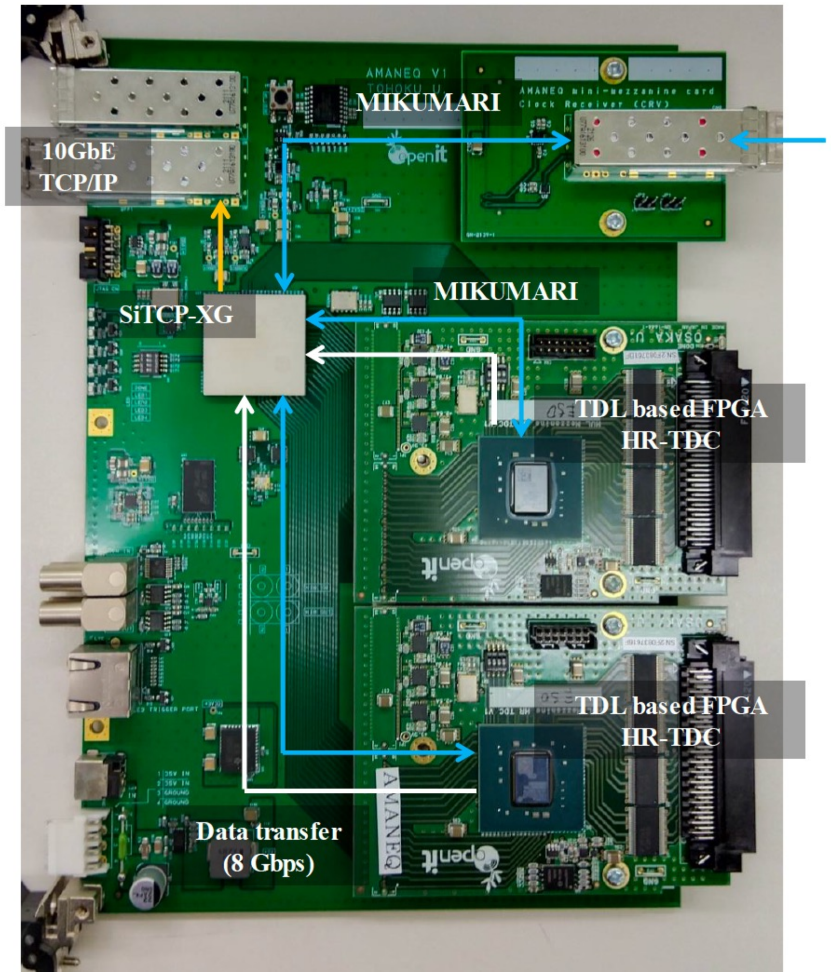
(MIKUMARI: 水分, R. Honda, IEEE TNS, 70 (6), 1102 (2023).



Time stamp is a unique way to reconstruct event



We started from streaming type TDC since it can be realized only in FPGA.



25 ps timing resolution including the synchronization precision.

WG3: nestDAQ

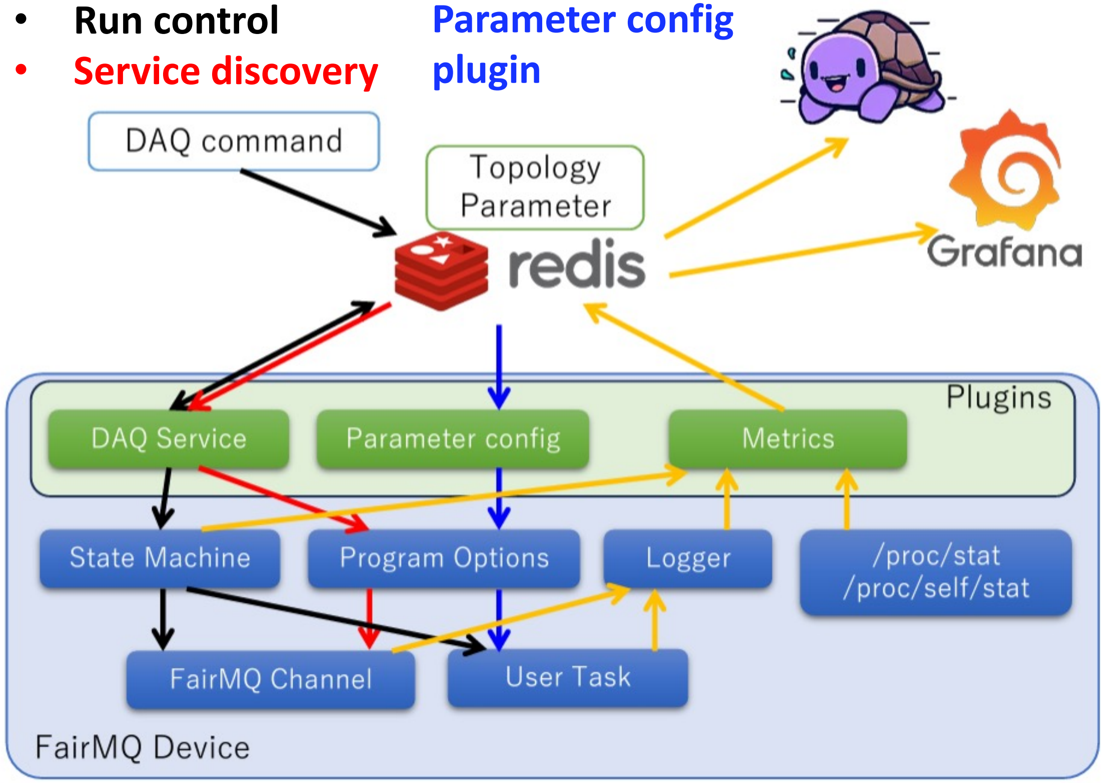
https://indico.bnl.gov/event/20010/contributions/79156/attachments/51299/87721/streamingws20231128_igarashi.pdf

- nestDAQ (**ne**twork based **st**reaming **DAQ**)
  - FairMQ (data transport based on zeroMQ, state machine control, plugins) + Redis (process management and control, in-memory access and fast response)

DAQ service plugin
- Run control
- Service discovery

Metrics plugin

Parameter config plugin



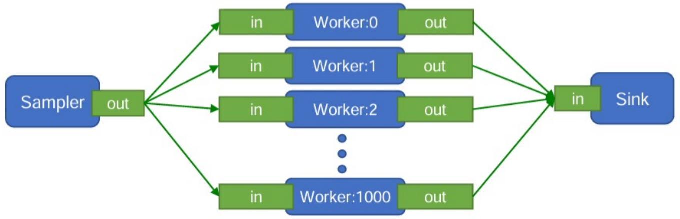
Semi-automatic topology generation based on service registry by Redis
- The database provides information about each process grouped as a function (service), its data channel-ports and their connections

Example: An arbitrary number of worker processes

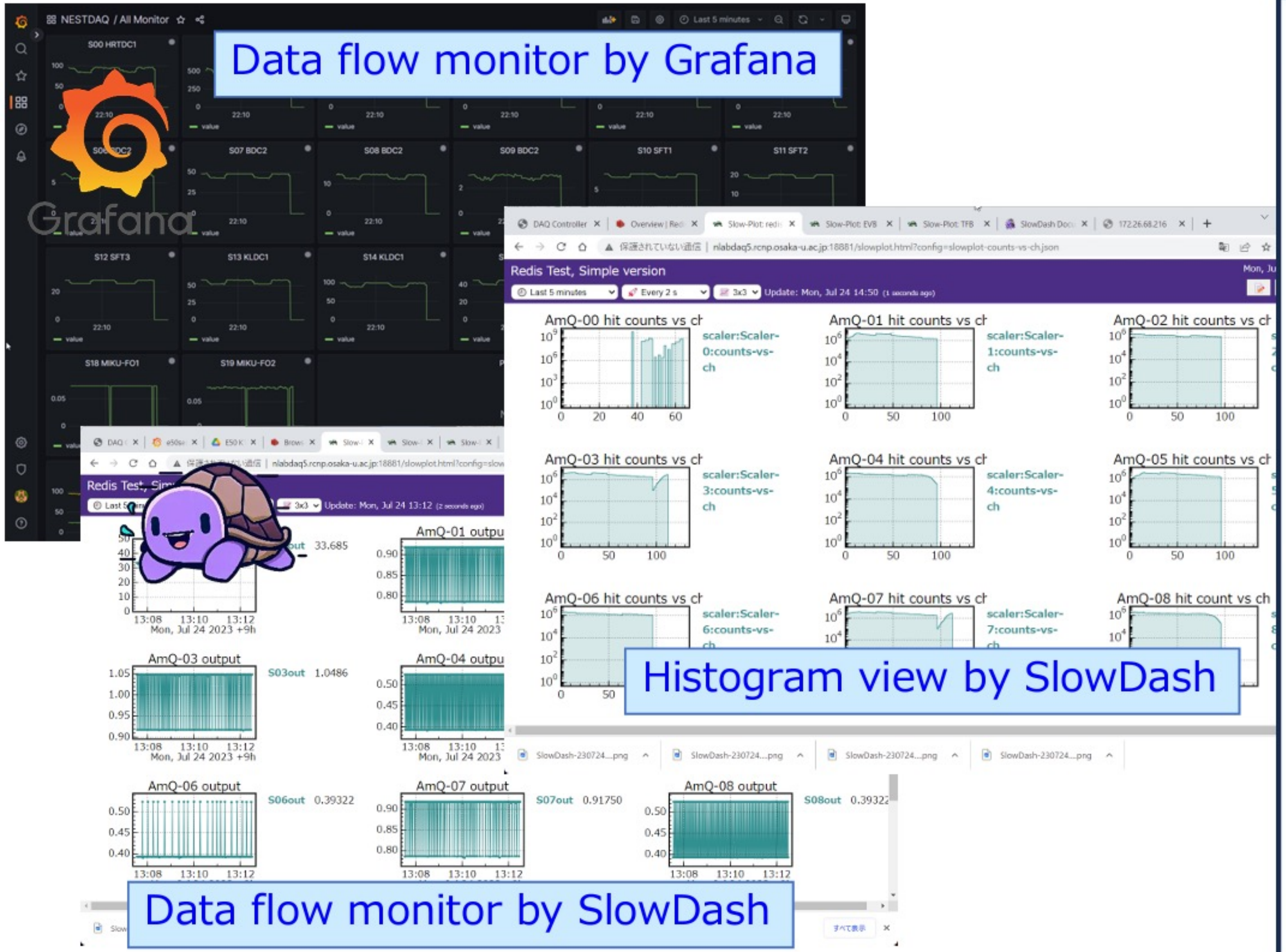
Topology data on the database

```
#-----------------------------------------------------------------
#           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



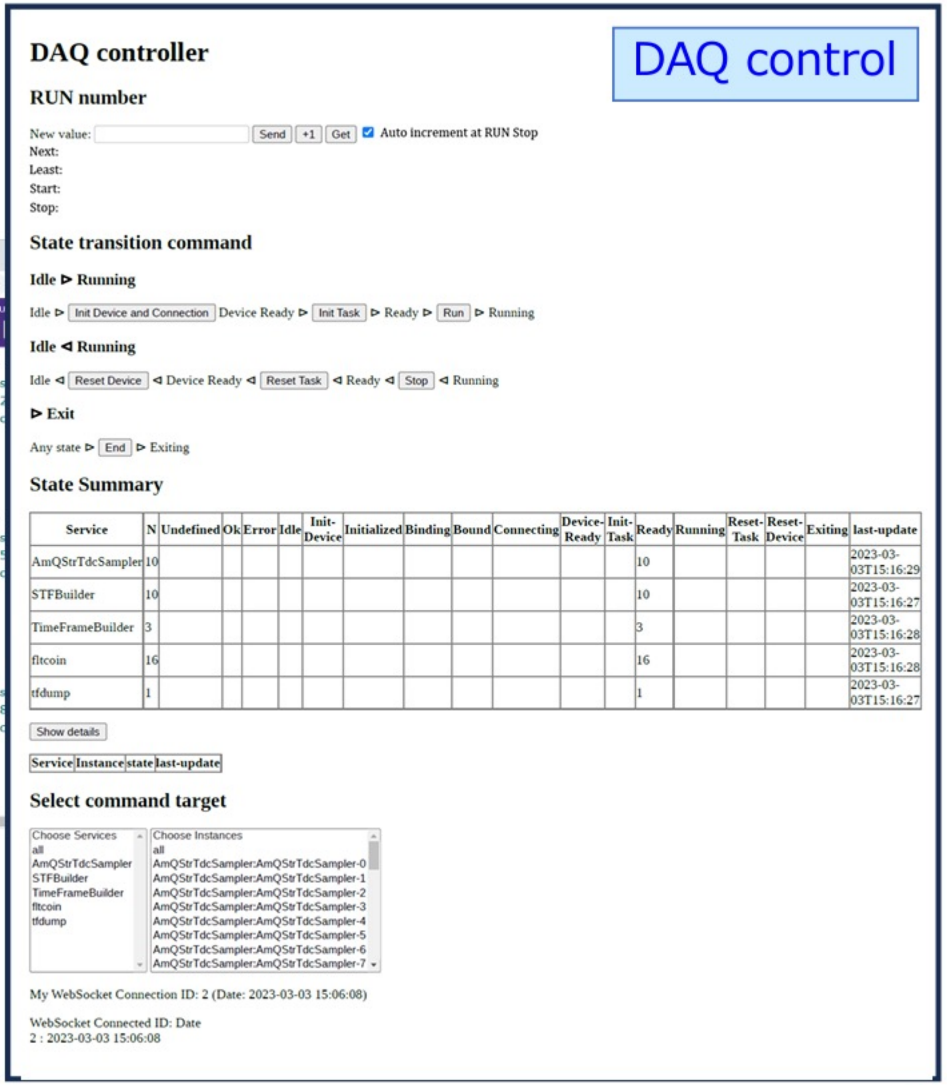
WG3: WebUI for nestDAQ



Data flow monitor by Grafana

Histogram view by SlowDash

Data flow monitor by SlowDash



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

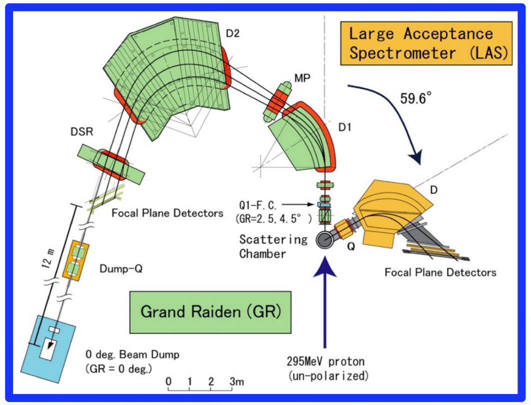
Choose Instances
all
AmQStrTdcSampler:AmQStrTdcSampler-0
AmQStrTdcSampler:AmQStrTdcSampler-1
AmQStrTdcSampler:AmQStrTdcSampler-2
AmQStrTdcSampler:AmQStrTdcSampler-3
AmQStrTdcSampler:AmQStrTdcSampler-4
AmQStrTdcSampler:AmQStrTdcSampler-5
AmQStrTdcSampler:AmQStrTdcSampler-6
AmQStrTdcSampler:AmQStrTdcSampler-7

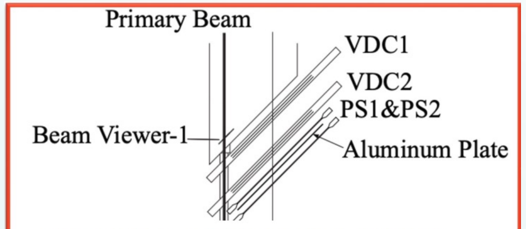
My WebSocket Connection ID: 2 (Date: 2023-03-03 15:06:08)

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

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

WG3: topologies tested at RCNP

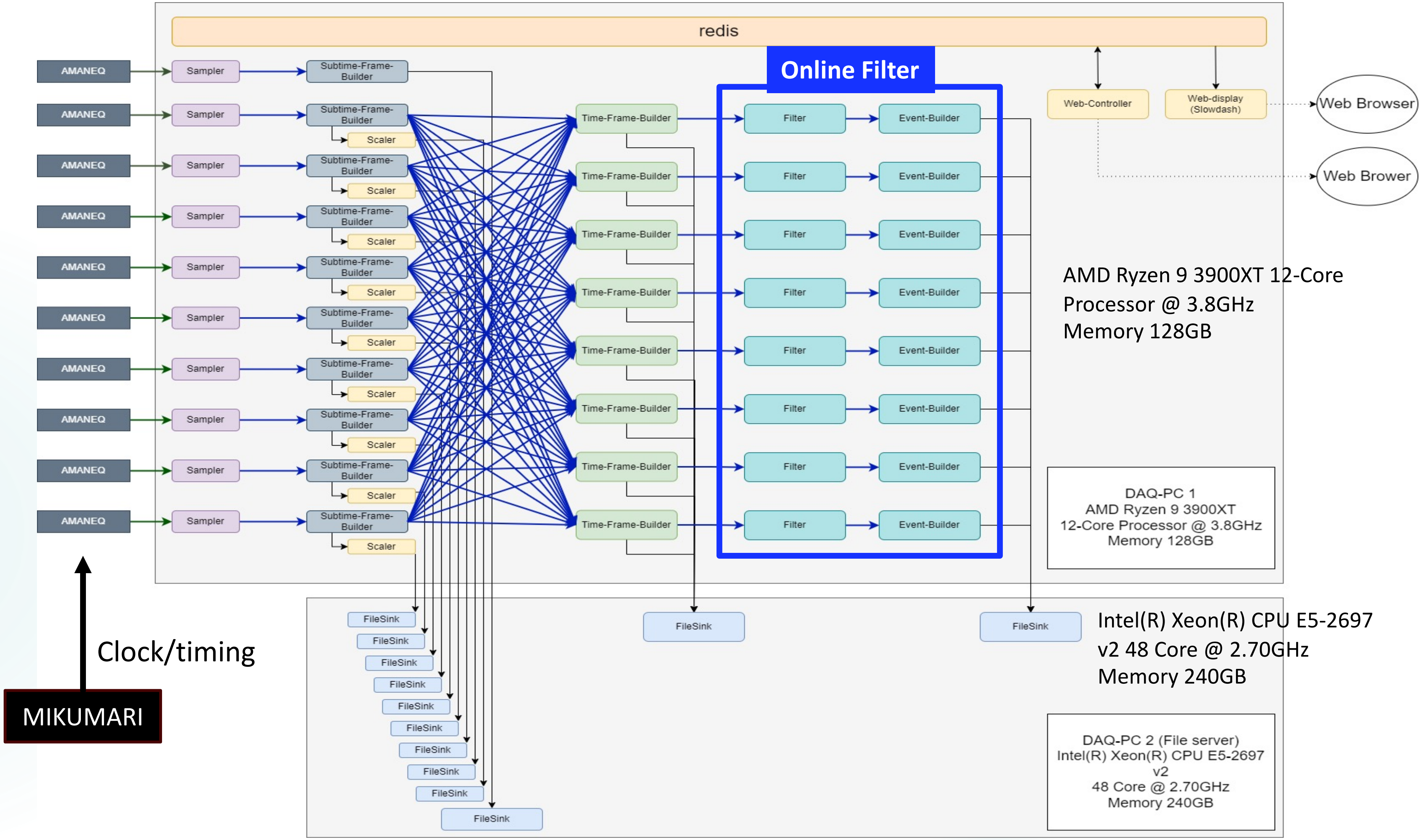




VDC (X1:192, U1:208, X2:192, U2:208)

PS (PS1: 2, PS2: 2)

RF timing, Current integrator

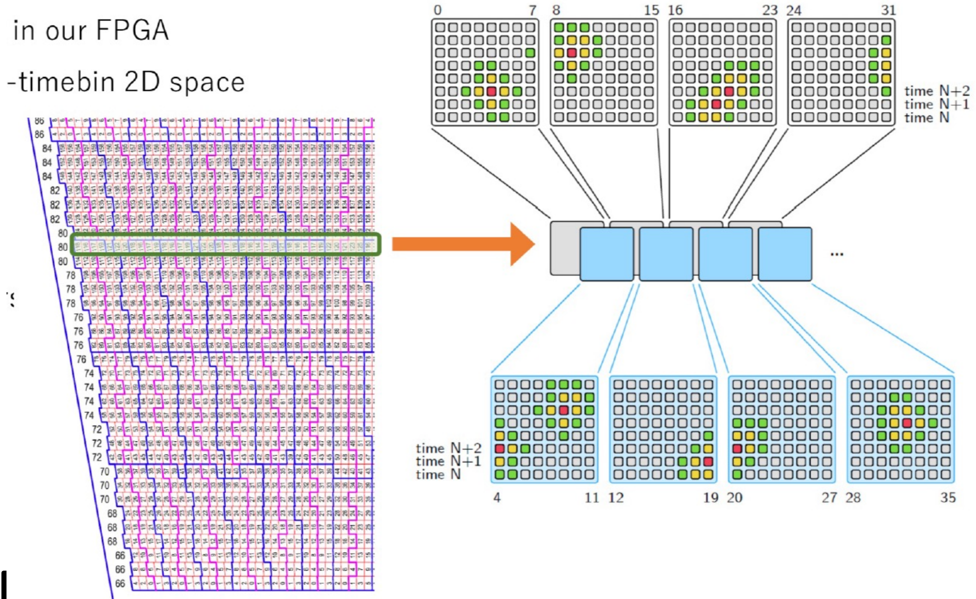


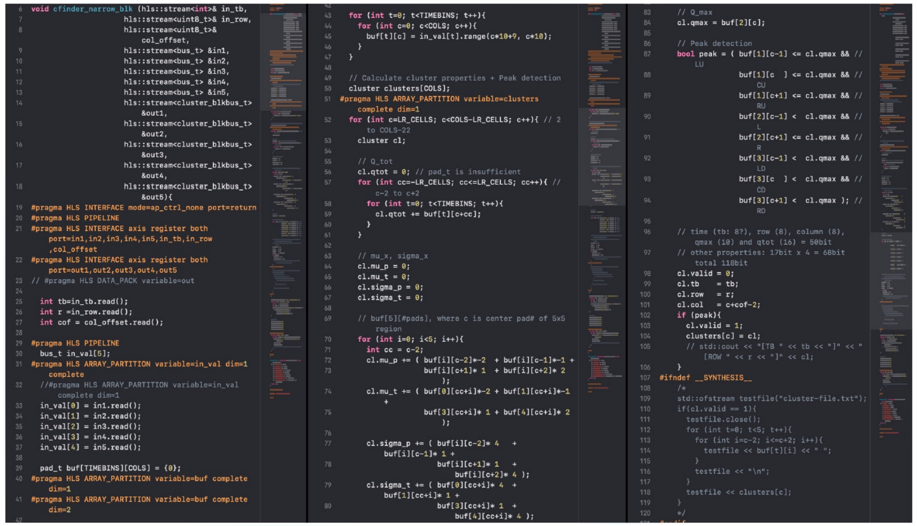
AMD Ryzen 9 3900XT 12-Core Processor @ 3.8GHz
Memory 128GB

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

WG4: online processing

- Several past and on-going activities
  - GPU processing
    - TPC Kalman filter tracking (for ALICE O2)
  - FPGA processing using HLS
    - ALICE TPC clustering (10 CRUs -> one Alveo)
    - Online PID from beamline detectors for RIBI
    - ML/AI tracking from MAPS detectors
- Joint efforts with HEP community (ATLAS, Belle2)

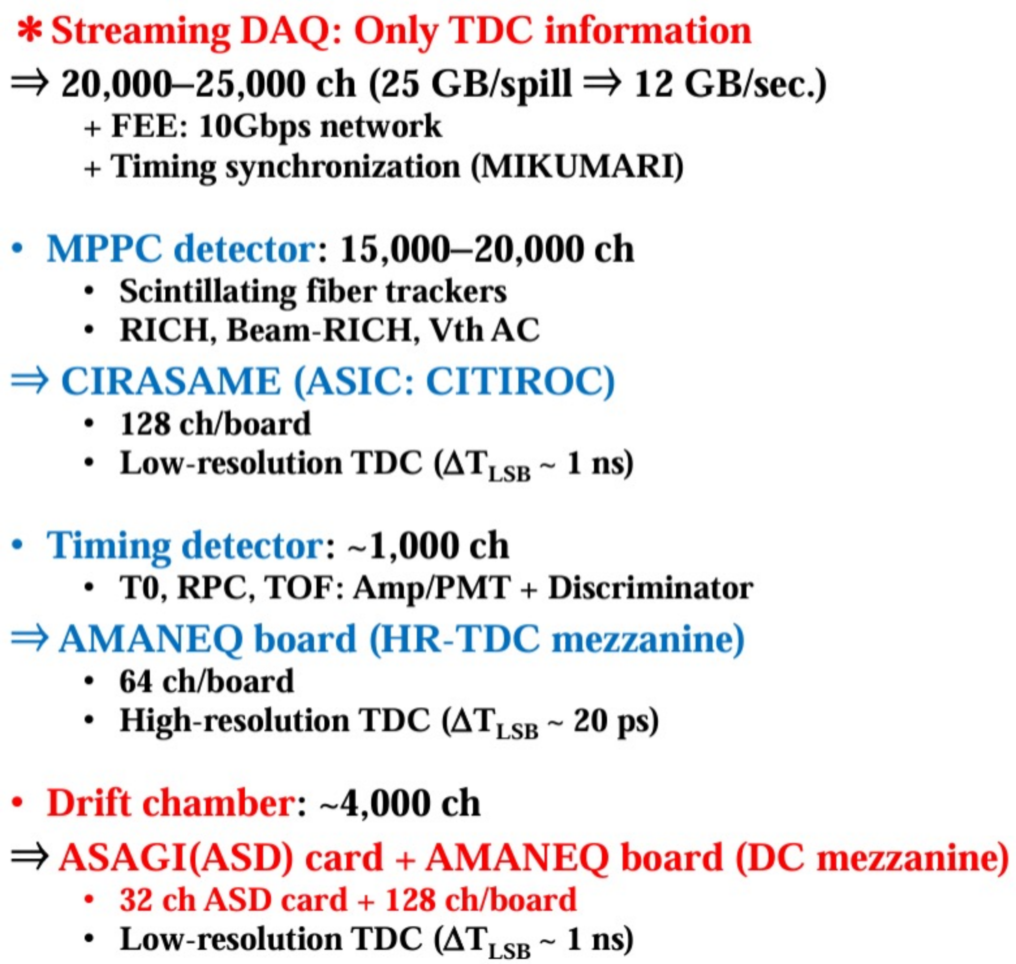




```
void cfinder_narrow_blk (hls::stream<int>& in_tb,
                         hls::stream<uint8_t>& in_row,
                         hls::stream<uint8_t>& col_offset,
                         hls::stream<bus_t> &in1,
                         hls::stream<bus_t> &in2,
                         hls::stream<bus_t> &in3,
                         hls::stream<bus_t> &in4,
                         hls::stream<bus_t> &in5,
                         hls::stream<cluster_blkbus_t> &out1,
                         hls::stream<cluster_blkbus_t> &out2,
                         hls::stream<cluster_blkbus_t> &out3,
                         hls::stream<cluster_blkbus_t> &out4,
                         hls::stream<cluster_blkbus_t> &out5){
#pragma HLS INTERFACE mode=ap_ctrl_none port=return
#pragma HLS PIPELINE
#pragma HLS INTERFACE axis register both port=in1,in2,in3,in4,in5,in_tb,in_row,col_offset
#pragma HLS INTERFACE axis register both port=out1,out2,out3,out4,out5
// #pragma HLS DATA_PACK variable=out

  int tb=in_tb.read();
  int r =in_row.read();
  int cof = col_offset.read();

#pragma HLS PIPELINE
  bus_t in_val[5];
#pragma HLS ARRAY_PARTITION variable=in_val dim=1 complete
  //#pragma HLS ARRAY_PARTITION variable=in_val complete dim=1
  in_val[0] = in1.read();
  in_val[1] = in2.read();
  in_val[2] = in3.read();
  in_val[3] = in4.read();
  in_val[4] = in5.read();

  pad_t buf[TIMEBINS][COLS] = {0};
#pragma HLS ARRAY_PARTITION variable=buf complete dim=1
#pragma HLS ARRAY_PARTITION variable=buf complete dim=2

  for (int t=0; t<TIMEBINS; t++){
    for (int c=0; c<COLS; c++){
      buf[t][c] = in_val[t].range(c*10+9, c*10);
    }
  }

  // Calculate cluster properties + Peak detection
  cluster clusters[COLS];
#pragma HLS ARRAY_PARTITION variable=clusters complete dim=1
  for (int c=LR_CELLS; c<COLS-LR_CELLS; c++){ // 2 to COLS-22
    cluster cl;

    // Q_tot
    cl.qtot = 0; // pad_t is insufficient
    for (int cc=-LR_CELLS; cc<=LR_CELLS; cc++){ // c-2 to c+2
      for (int t=0; t<TIMEBINS; t++){
        cl.qtot += buf[t][c+cc];
      }
    }

    // mu_x, sigma_x
    cl.mu_p = 0;
    cl.mu_t = 0;
    cl.sigma_p = 0;
    cl.sigma_t = 0;

    // buf[5][#pads], where c is center pad# of 5x5 region
    for (int i=0; i<5; i++){
      int cc = c-2;
      cl.mu_p += ( buf[i][c-2]*-2  + buf[i][c-1]*-1 +
                   buf[i][c+1]* 1  + buf[i][c+2]* 2 );
      cl.mu_t += ( buf[0][cc+i]*-2 + buf[1][cc+i]*-1 +
                   buf[3][cc+i]* 1 + buf[4][cc+i]* 2 );

      cl.sigma_p += ( buf[i][c-2]* 4   + buf[i][c-1]* 1 +
                      buf[i][c+1]* 1   + buf[i][c+2]* 4 );
      cl.sigma_t += ( buf[0][cc+i]* 4  + buf[1][cc+i]* 1 +
                      buf[3][cc+i]* 1  + buf[4][cc+i]* 4 );

    // Q_max
    cl.qmax = buf[2][c];

    // Peak detection
    bool peak = ( buf[1][c-1] <= cl.qmax && // LU
                  buf[1][c  ] <= cl.qmax && // CU
                  buf[1][c+1] <= cl.qmax && // RU
                  buf[2][c-1] <  cl.qmax && // L
                  buf[2][c+1] <= cl.qmax && // R
                  buf[3][c-1] <  cl.qmax && // LD
                  buf[3][c  ] <  cl.qmax && // CD
                  buf[3][c+1] <  cl.qmax ); // RD

    // time (tb: 8?), row (8), column (8), qmax (10) and qtot (16) = 50bit
    // other properties: 17bit x 4 = 68bit total 118bit
    cl.valid = 0;
    cl.tb    = tb;
    cl.row   = r;
    cl.col   = c+cof-2;
    if (peak){
      cl.valid = 1;
      clusters[c] = cl;
      // std::cout << "[TB " << tb << "]" << " [ROW " << r << "]" << cl;
    }
#ifndef __SYNTHESIS__
    /*
    std::ofstream testfile("cluster-file.txt");
    if(cl.valid == 1){
      testfile.close();
      for (int t=0; t<5; t++){
        for (int i=c-2; i<=c+2; i++){
          testfile << buf[t][i] << " ";
        }
        testfile << "\n";
      }
      testfile << clusters[c];
    }
    */
```

J-PARC E50

- It is important to build a testbed for ePIC SRO-DAQ (ex, mCBM at SIS18)
- J-PARC E50 (charmed baryon spectroscopy) could be used as a testbed for ePIC.



＊Streaming DAQ: Only TDC information

⇒ 20,000–25,000 ch (25 GB/spill ⇒ 12 GB/sec.)

+ FEE: 10Gbps network
+ Timing synchronization (MIKUMARI)

- **MPPC detector: 15,000–20,000 ch**
  - Scintillating fiber trackers
  - RICH, Beam-RICH, Vth AC

⇒ CIRASAME (ASIC: CITIROC)

- 128 ch/board
- Low-resolution TDC (ΔT_{LSB} ~ 1 ns)

- **Timing detector: ~1,000 ch**
  - T0, RPC, TOF: Amp/PMT + Discriminator

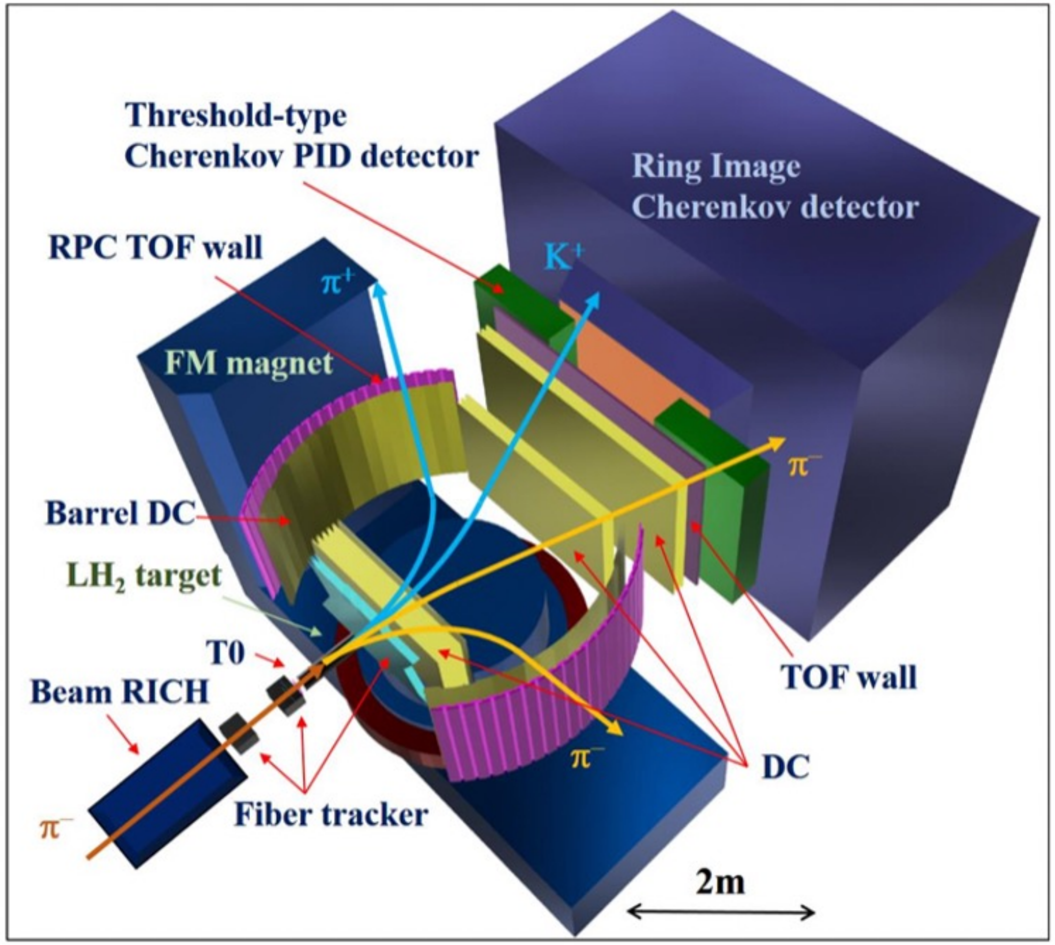
⇒ AMANEQ board (HR-TDC mezzanine)

- 64 ch/board
- High-resolution TDC (ΔT_{LSB} ~ 20 ps)

- **Drift chamber: ~4,000 ch**

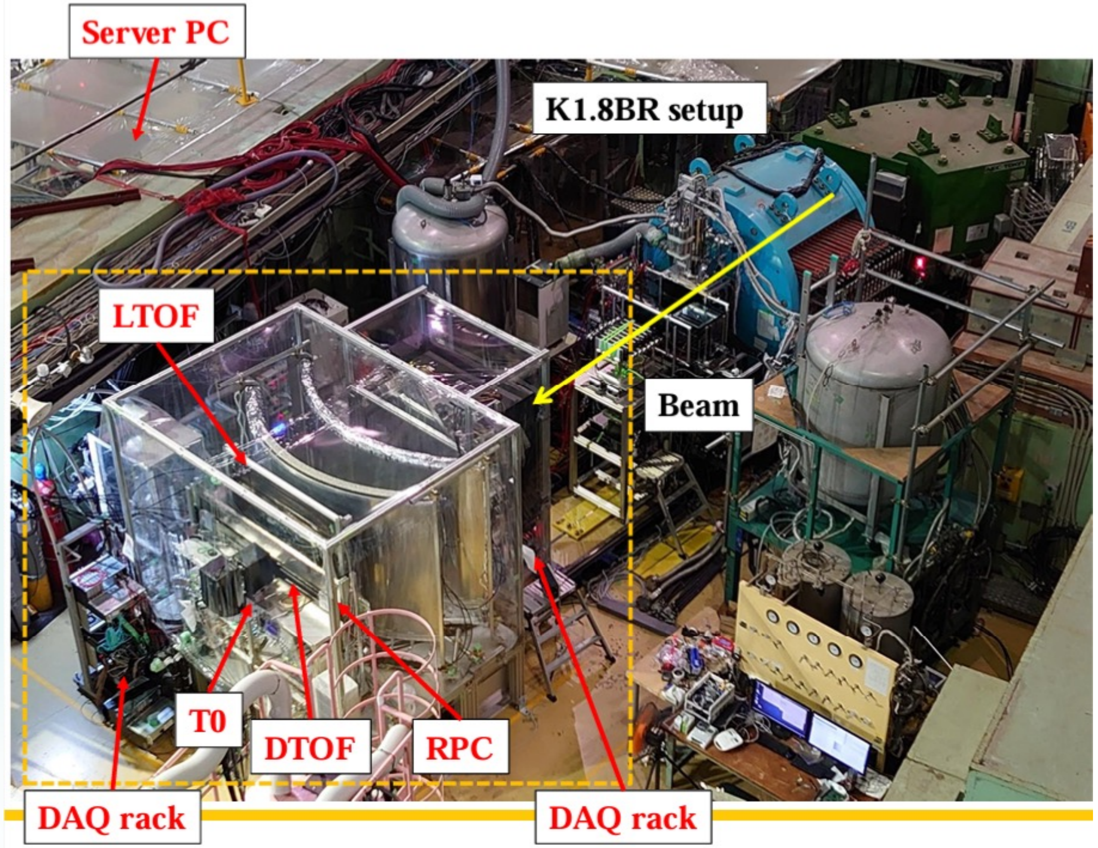
⇒ ASAGI(ASD) card + AMANEQ board (DC mezzanine)

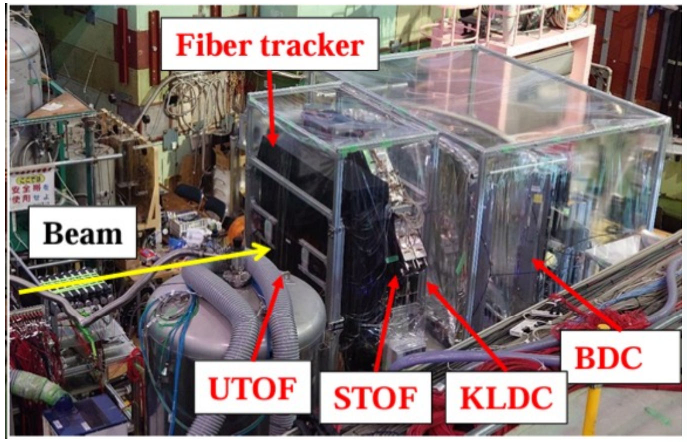
- 32 ch ASD card + 128 ch/board
- Low-resolution TDC (ΔT_{LSB} ~ 1 ns)

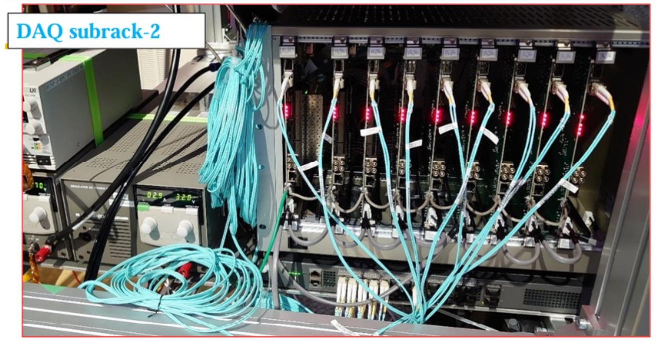


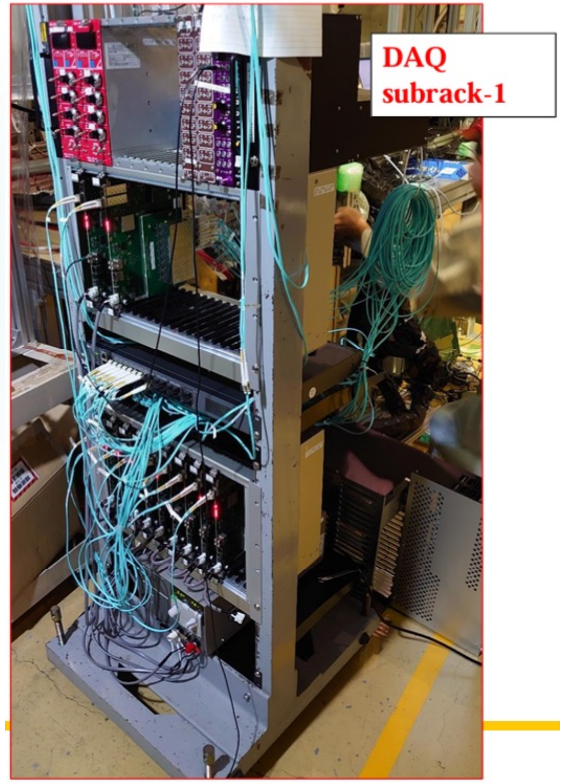
J-PARC E50

- J-PARC E50 (charmed baryon spectroscopy) could be used as a testbed for ePIC?









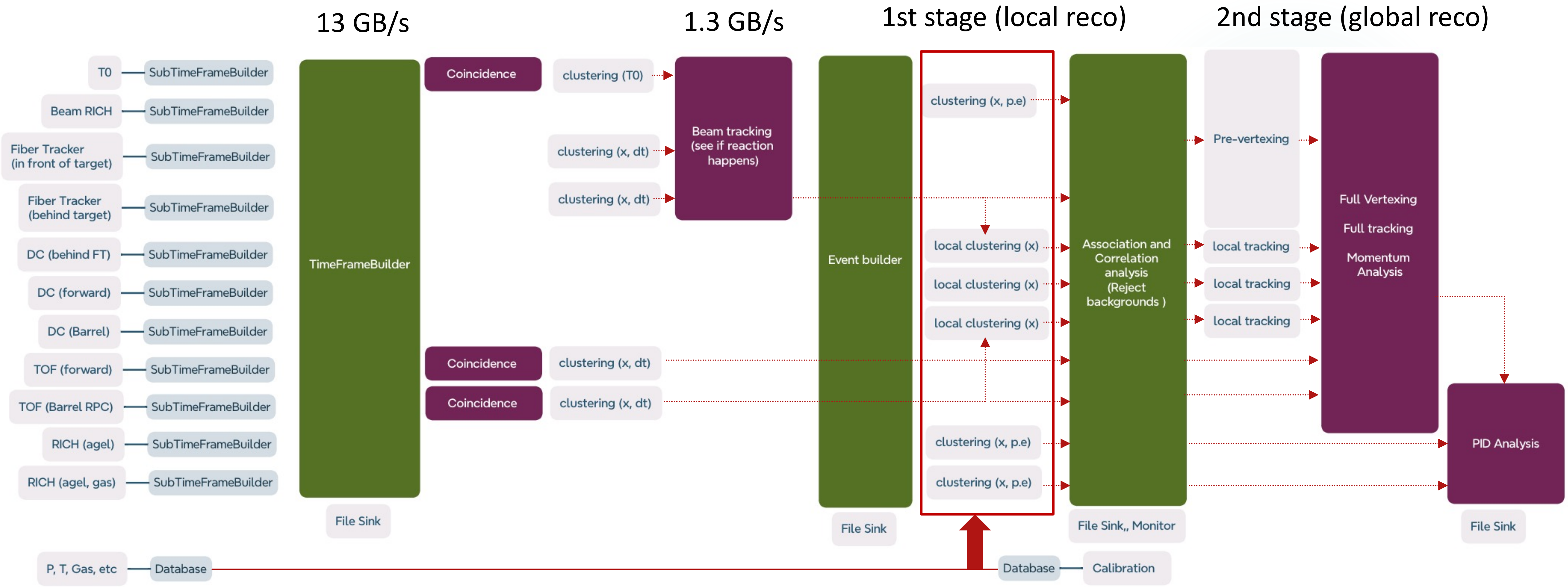
If so, need to integrate several key components or use our own resources

- Timing distribution system (GTM or MIKUMARI), FEC (with ePIC link protocol), aggregator DAM module, Computing nodes (mini-echelon0/1), online processing (noise filtering etc…)

J-PARC E50

30 MHz beam rate
5% reaction rate
TF length = 512usec
(60k beams, 3k events in TF)

- **Workflows for online processing**

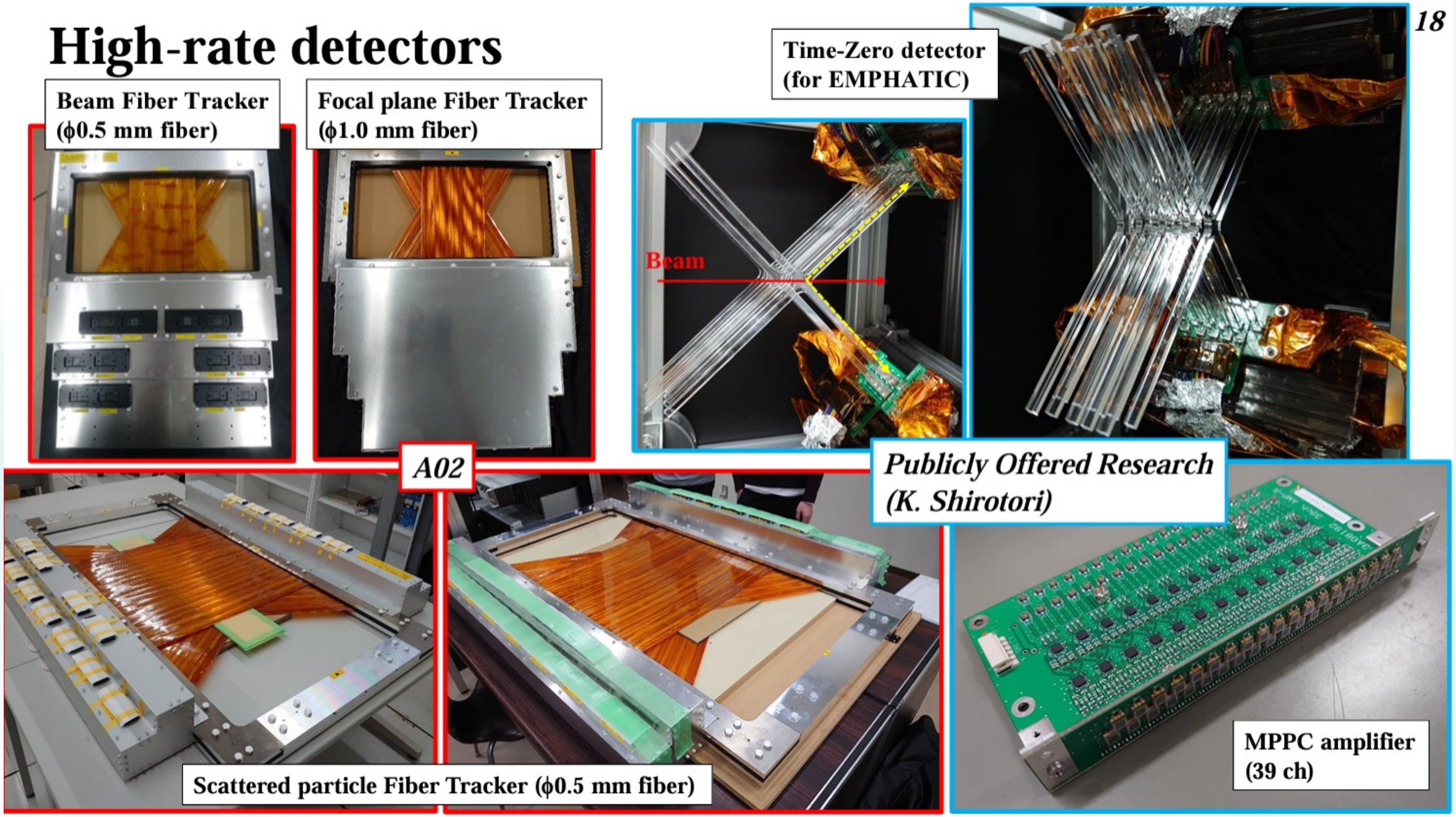


Summary

- In Japan, SPADI-Alliance was built to realize and standardization of SRO readout DAQ in many facilities.
  - Full-streaming readout and software based event selection was successfully performed
- It is natural to collaborate with international projects such as EIC/ePIC.
- Possible contributions from SPADI-Alliance (to achieve win-win relations).
  - ASIC testing and development of timing distribution system (GTM and MIKUMARI)
  - nestDAQ
  - Online processing on FELIX (AI/ML on Versal) and GPUs for benchmarking
  - Provide E50 (and Belle2 – will be discussed) as a testbed of ePIC Streaming DAQ
- It would be great if EIC/ePIC SRO team and SPADI-Alliance collaborate tightly to import and export each expertise and develop the SRO DAQ.

Backup I

E50: High Rate detectors



18

High-rate detectors

Beam Fiber Tracker (ϕ0.5 mm fiber)

Focal plane Fiber Tracker (ϕ1.0 mm fiber)

Time-Zero detector (for EMPHATIC)

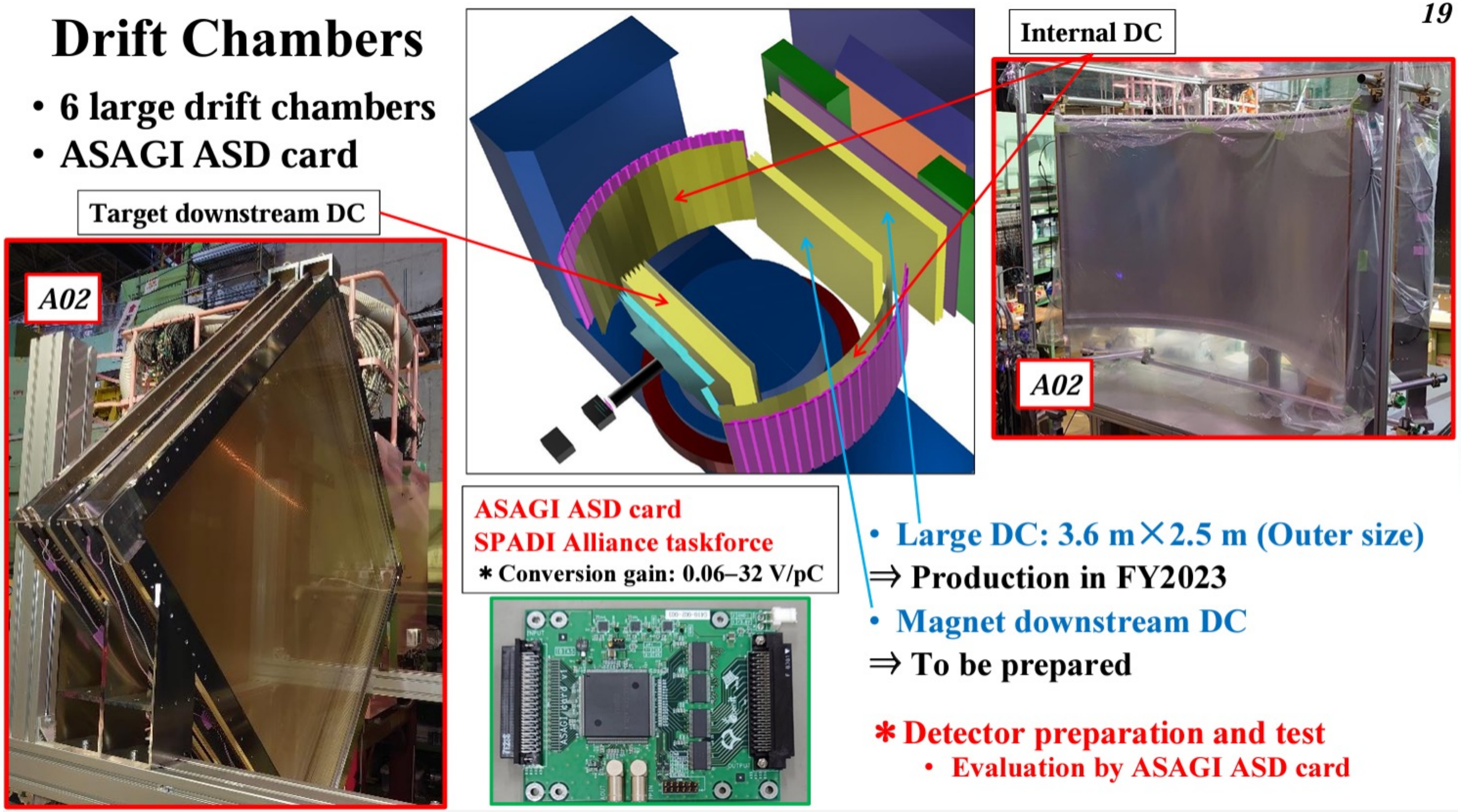
A02

Publicly Offered Research (K. Shirotori)

Scattered particle Fiber Tracker (ϕ0.5 mm fiber)

MPPC amplifier (39 ch)

E50: Drift chambers



19

Drift Chambers

- 6 large drift chambers
- ASAGI ASD card

ASAGI ASD card
SPADI Alliance taskforce
* Conversion gain: 0.06–32 V/pC

- Large DC: 3.6 m × 2.5 m (Outer size)

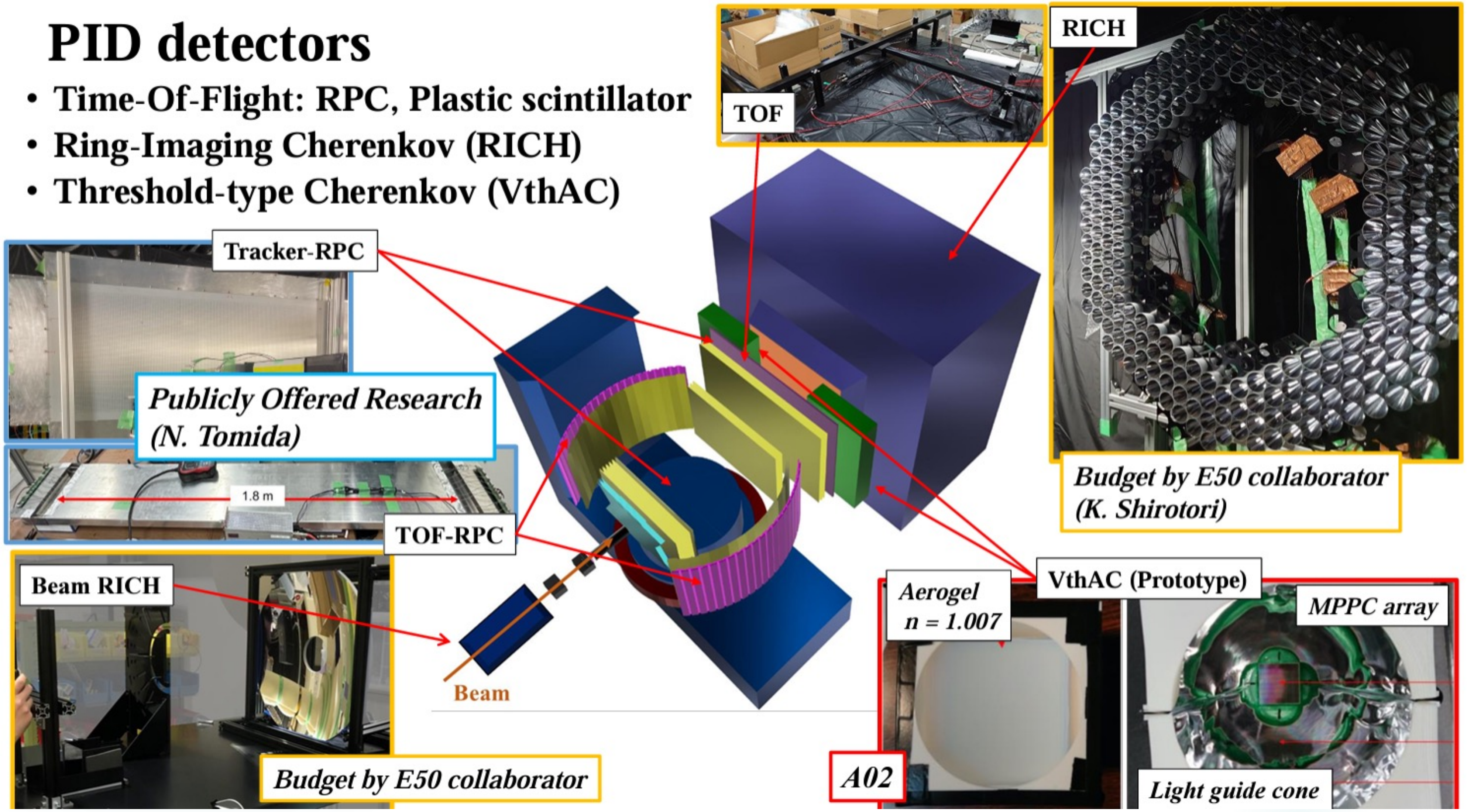
⇒ Production in FY2023

- Magnet downstream DC

⇒ To be prepared

* Detector preparation and test
  - Evaluation by ASAGI ASD card

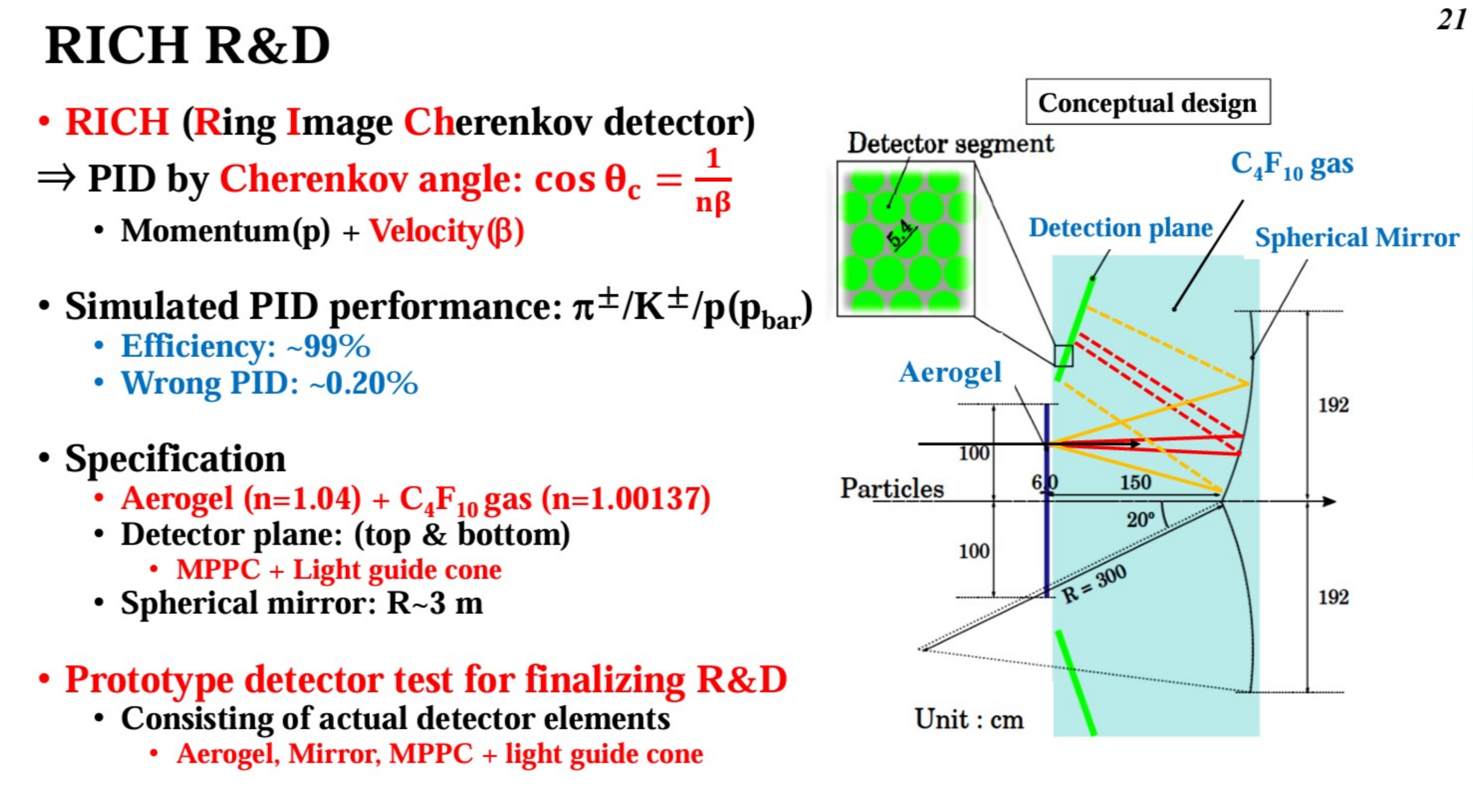
E50: PID detectors



PID detectors

- Time-Of-Flight: RPC, Plastic scintillator
- Ring-Imaging Cherenkov (RICH)
- Threshold-type Cherenkov (VthAC)

E50: RICH R&D



RICH R&D

- RICH (Ring Image Cherenkov detector)

⇒ PID by Cherenkov angle: $\cos\theta_c = \frac{1}{n\beta}$

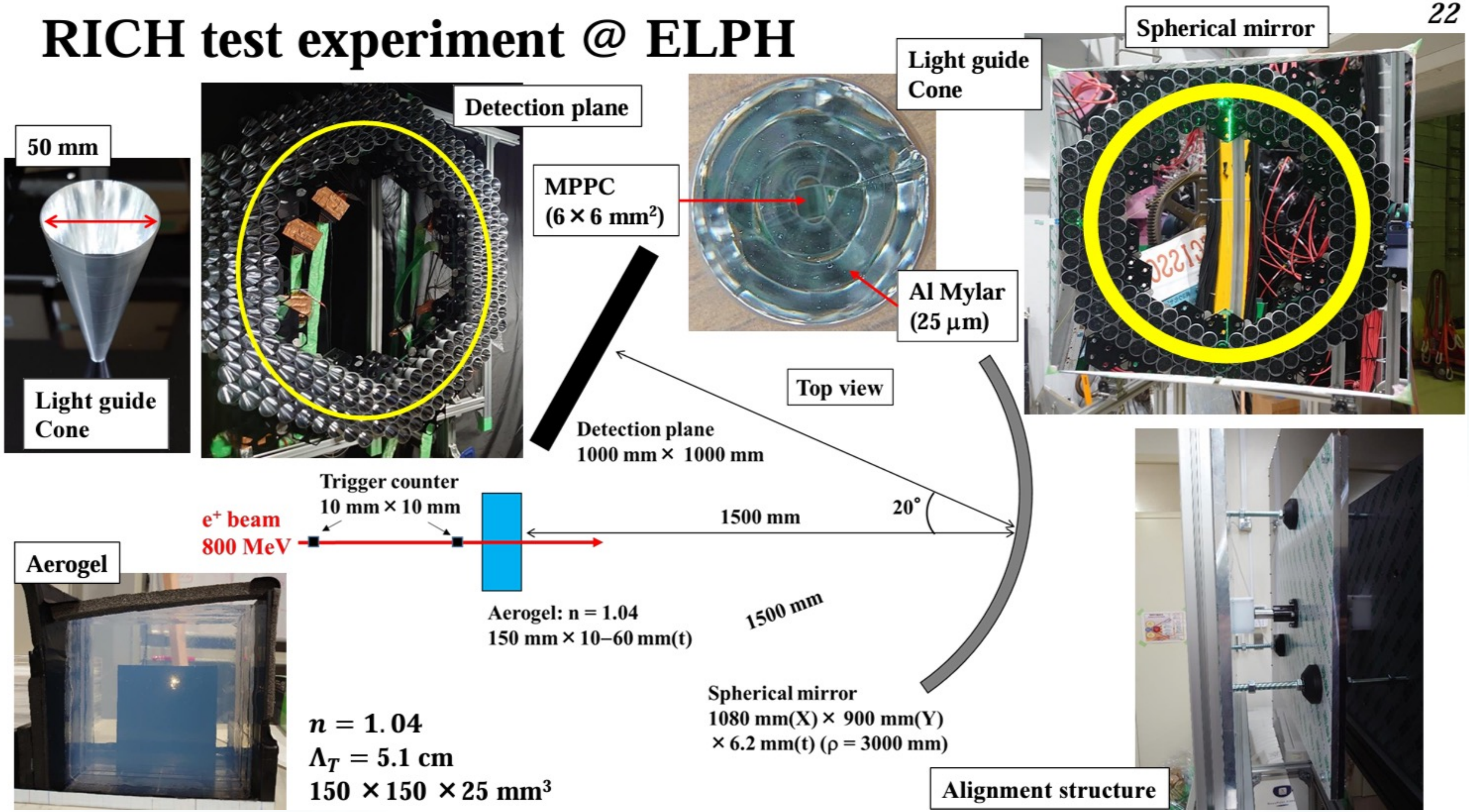
  - Momentum(p) + Velocity(β)

- Simulated PID performance: $\pi^{\pm}/K^{\pm}/p(p_{bar})$
  - Efficiency: ~99%
  - Wrong PID: ~0.20%

- Specification
  - Aerogel (n=1.04) + C_4F_{10} gas (n=1.00137)
  - Detector plane: (top & bottom)
    - MPPC + Light guide cone
  - Spherical mirror: R~3 m

- Prototype detector test for finalizing R&D
  - Consisting of actual detector elements
    - Aerogel, Mirror, MPPC + light guide cone

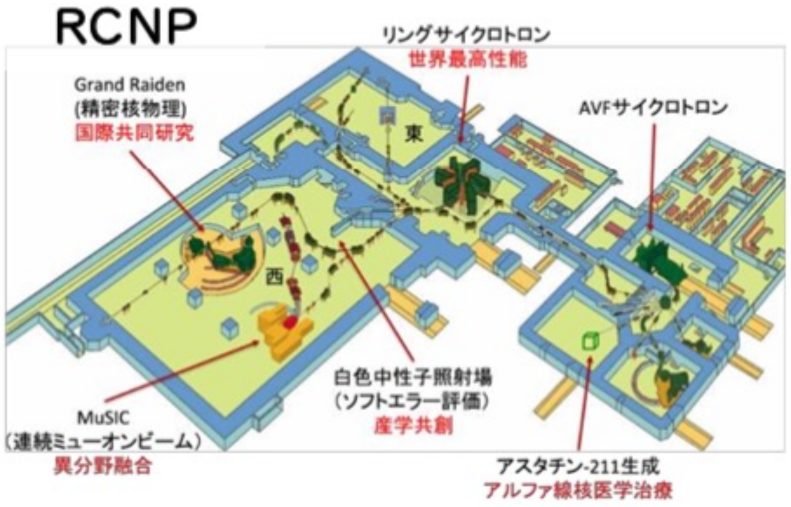
E50: RICH R&D



RICH test experiment @ ELPH

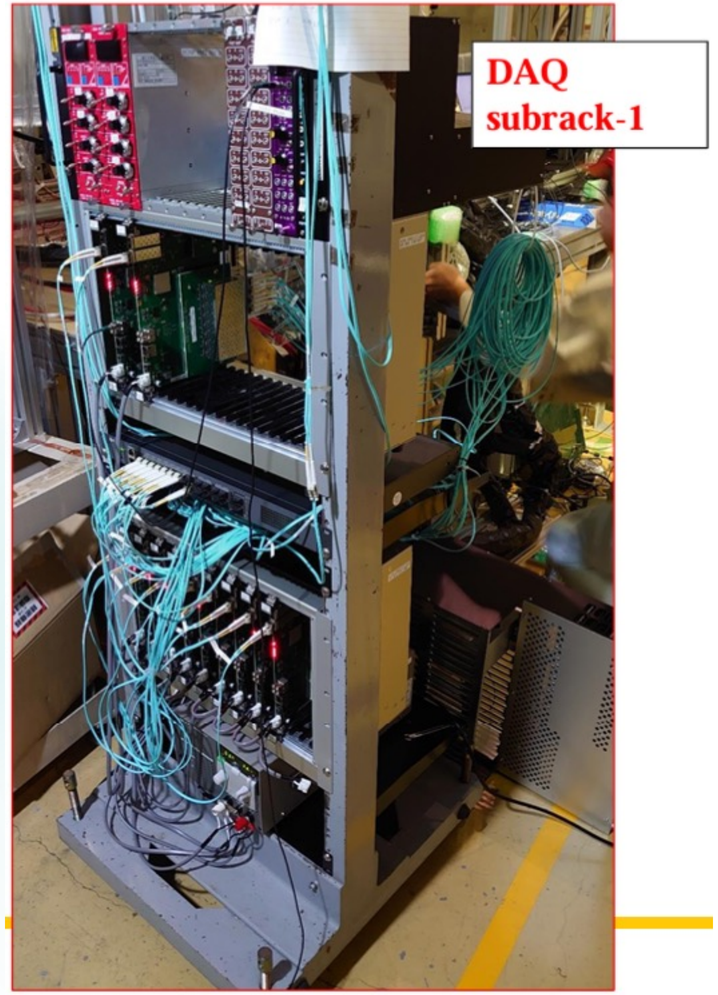
Tested in RCNP and J-PARC

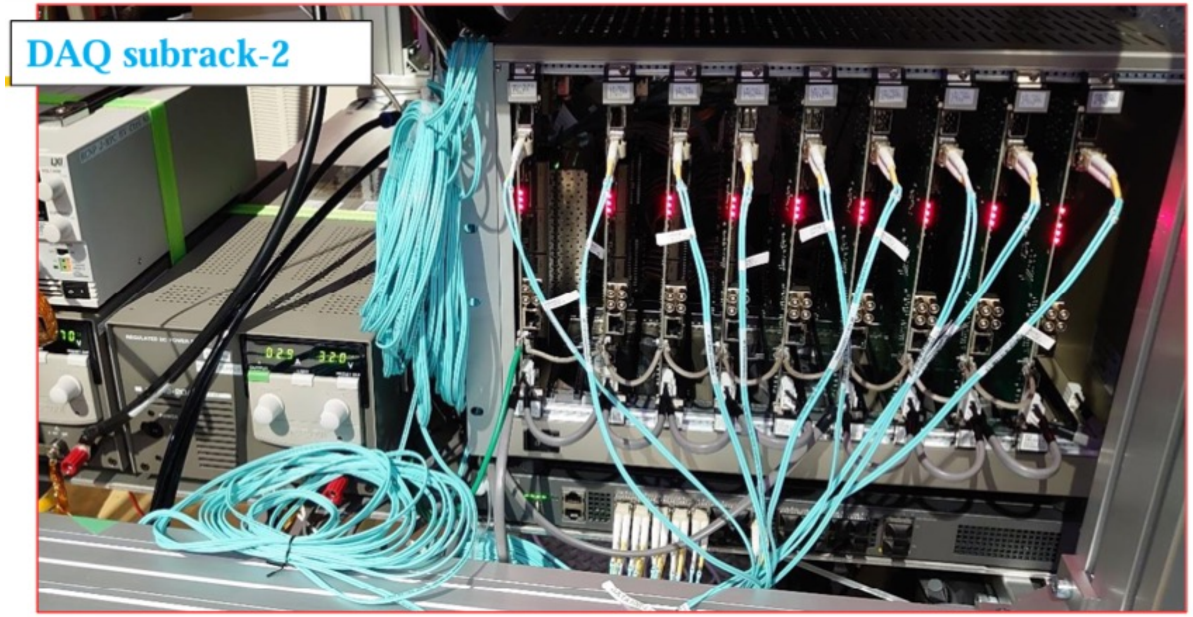
- Tested at RCNP Grand Raiden
  - Primary beams from Cyclotron
  - Test experiment and physics experiment
    - ~0.2 Gbps in total



- Tested at Hadron facility, K1.8 BR beamline at J-PARC
  - Secondary beams from Synchrotron (slow extraction)
    - Pi, k, proton
  - Test experiment
    - ~2 Gbps in total

Pictures of test exp. At J-PARC





Detectors

- BDC: 900 ch+ KLDC: 512 ch
- SFT: 384 ch
- TOF, MRPC, Timing counters: 62 ch

Read by Str-LRTDC (1ns)

Read by Str-HRTDC

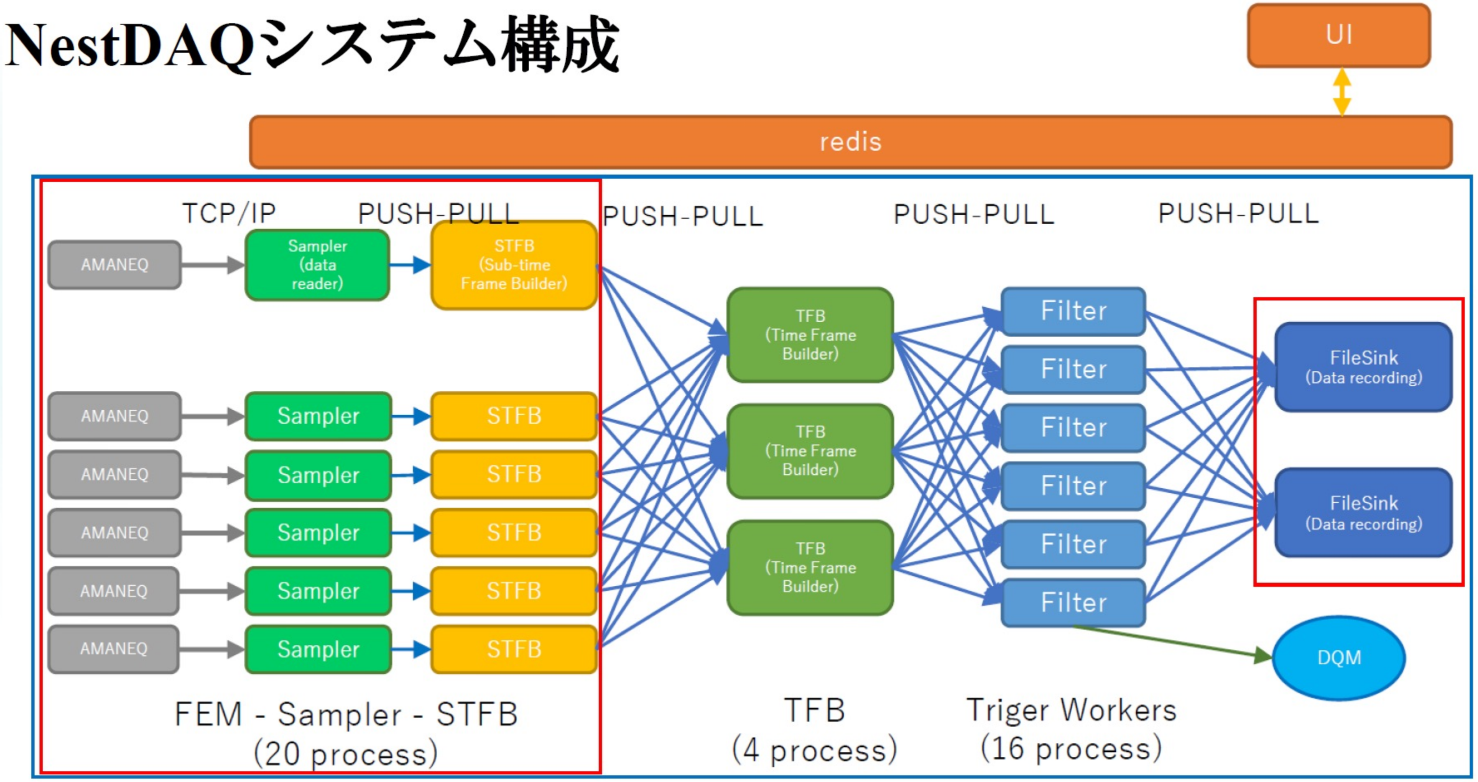
DAQ system

- FEE: AMANEQ x20 (Streaming TDCx17 + MIKUMARI x3)
  - Str-LRTDC AMANEQ x15
  - Str-HRTDC AMANEQ x2

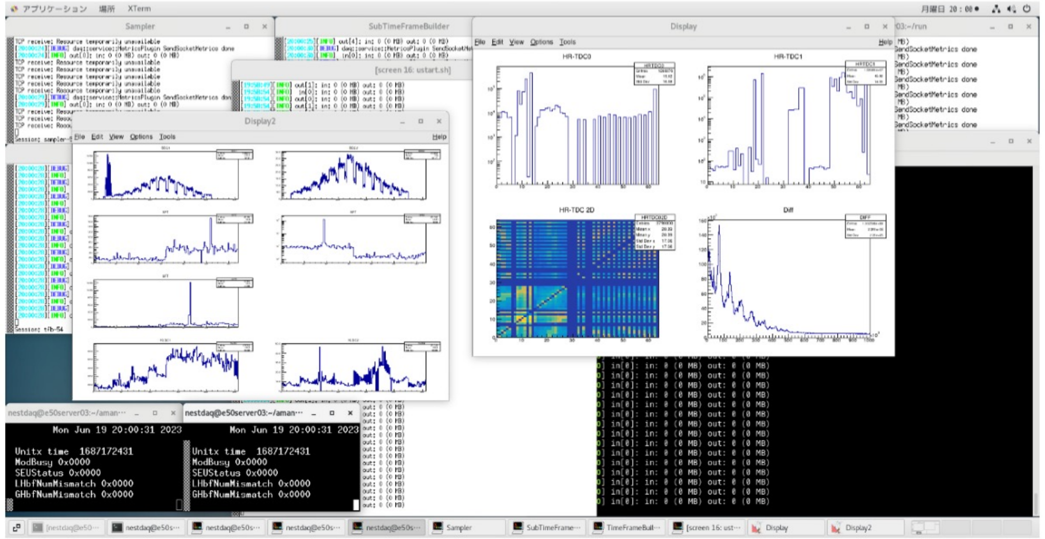
NestDAQsoftware running on a PC

nestDAQ of test exp. at J-PARC

- Beamtime in last June (19.6-21.6)
  - K^- ~ 200k/spill (2sec), π^- ~ 800k/spill
  - Tests of streaming DAQ with “nestDAQ” and online coincidence filtering (based on CPU)



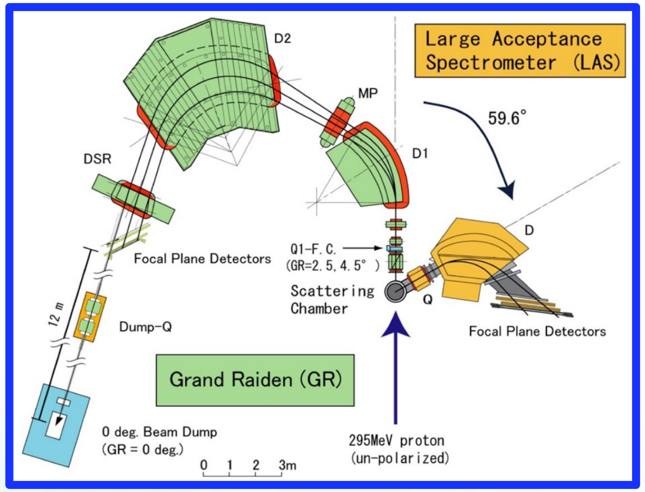
NestDAQシステム構成

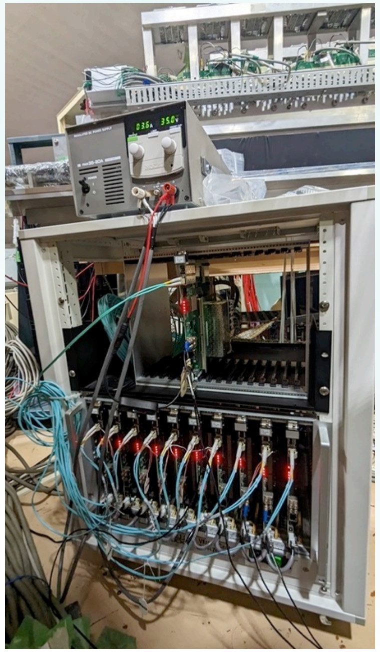


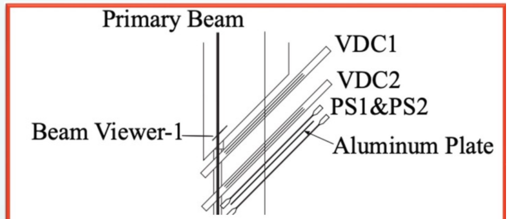
```
        Mon Jun 19 20:00:31 2023

 Unitx time  1687172431
 ModBusy 0x0000
 SEUStatus 0x0000
 LHbfNumMismatch 0x0000
 GHbfNumMismatch 0x0000
```

```
        Mon Jun 19 20:00:31 2023

 Unitx time  1687172431
 ModBusy 0x0000
 SEUStatus 0x0000
 LHbfNumMismatch 0x0000
 GHbfNumMismatch 0x0000
```

nestDAQ at RCNP physics data taking



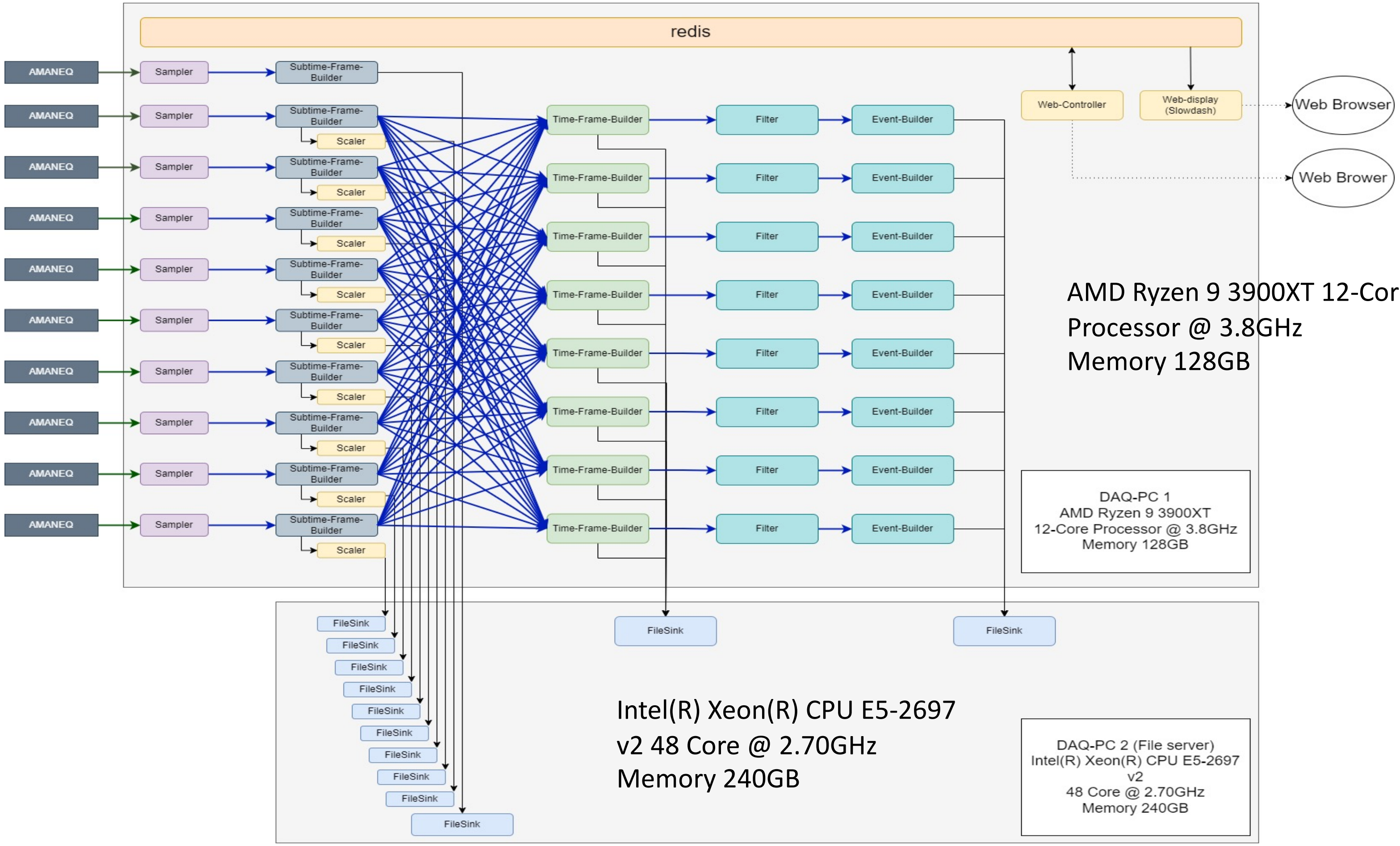




VDC (X1:192, U1:208, X2:192, U2:208)

PS (PS1: 2, PS2: 2)

RF timing, Current integrator



redis

AMANEQ → Sampler → Subtime-Frame-Builder

AMANEQ → Sampler → Subtime-Frame-Builder → Scaler

AMANEQ → Sampler → Subtime-Frame-Builder → Scaler

AMANEQ → Sampler → Subtime-Frame-Builder → Scaler

AMANEQ → Sampler → Subtime-Frame-Builder → Scaler

AMANEQ → Sampler → Subtime-Frame-Builder → Scaler

AMANEQ → Sampler → Subtime-Frame-Builder → Scaler

AMANEQ → Sampler → Subtime-Frame-Builder → Scaler

AMANEQ → Sampler → Subtime-Frame-Builder → Scaler

Time-Frame-Builder → Filter → Event-Builder (×8)

Web-Controller

Web-display (Slowdash)

Web Browser

Web Brower

FileSink (×10)

FileSink

FileSink

DAQ-PC 1
AMD Ryzen 9 3900XT
12-Core Processor @ 3.8GHz
Memory 128GB

DAQ-PC 2 (File server)
Intel(R) Xeon(R) CPU E5-2697 v2
48 Core @ 2.70GHz
Memory 240GB

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

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

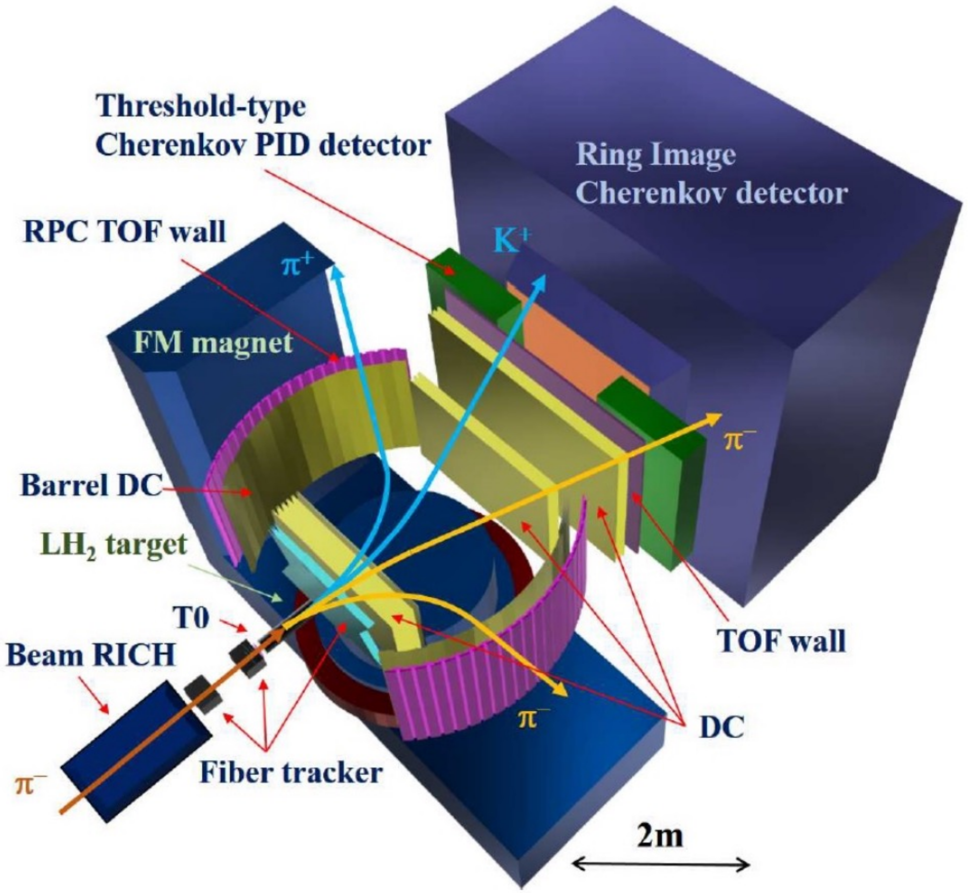
Throughput 200Mbps (x40 improvement from past DAQ system). 100% efficient for 100-200 kcps
Next step is to implement online tracking, multi-hit identification (ex (d, 2p)), and PID (ex, (α, ^{6}He))

Possible contributions for ePIC

- Timing distribution system
  - Development of ePIC-GTM system and Mikumari (if there are rooms for using Mikumari)
- nestDAQ
  - Could be a central DAQ system for ePIC?
- Online processing
  - AI/ML and HLS for FPGA on FELIX cards
  - GPU processing on echelon 0 or 1 or 2 (wherever applicable)
- Providing testbed system to develop SRO DAQ with real hardware and software
  - Propose to use J-PARC E50 experiment (and BELLE2 – will be discussed) as the testbed of Streaming DAQ for ePIC
  - Integrate timing system, some FECs (with small detector prototypes), DAM, and SRO software and hardware accelerators
  - It is important to consider of building such testbed systems for ePIC NOW.

J-PARC E50

- J-PARC E50 experiment as the testbed of Streaming DAQ for ePIC?

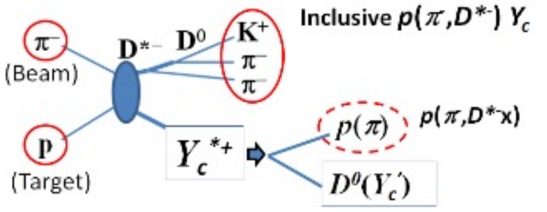


E50 : General detector system at J-PARC high-momentum beamline for various physics topics

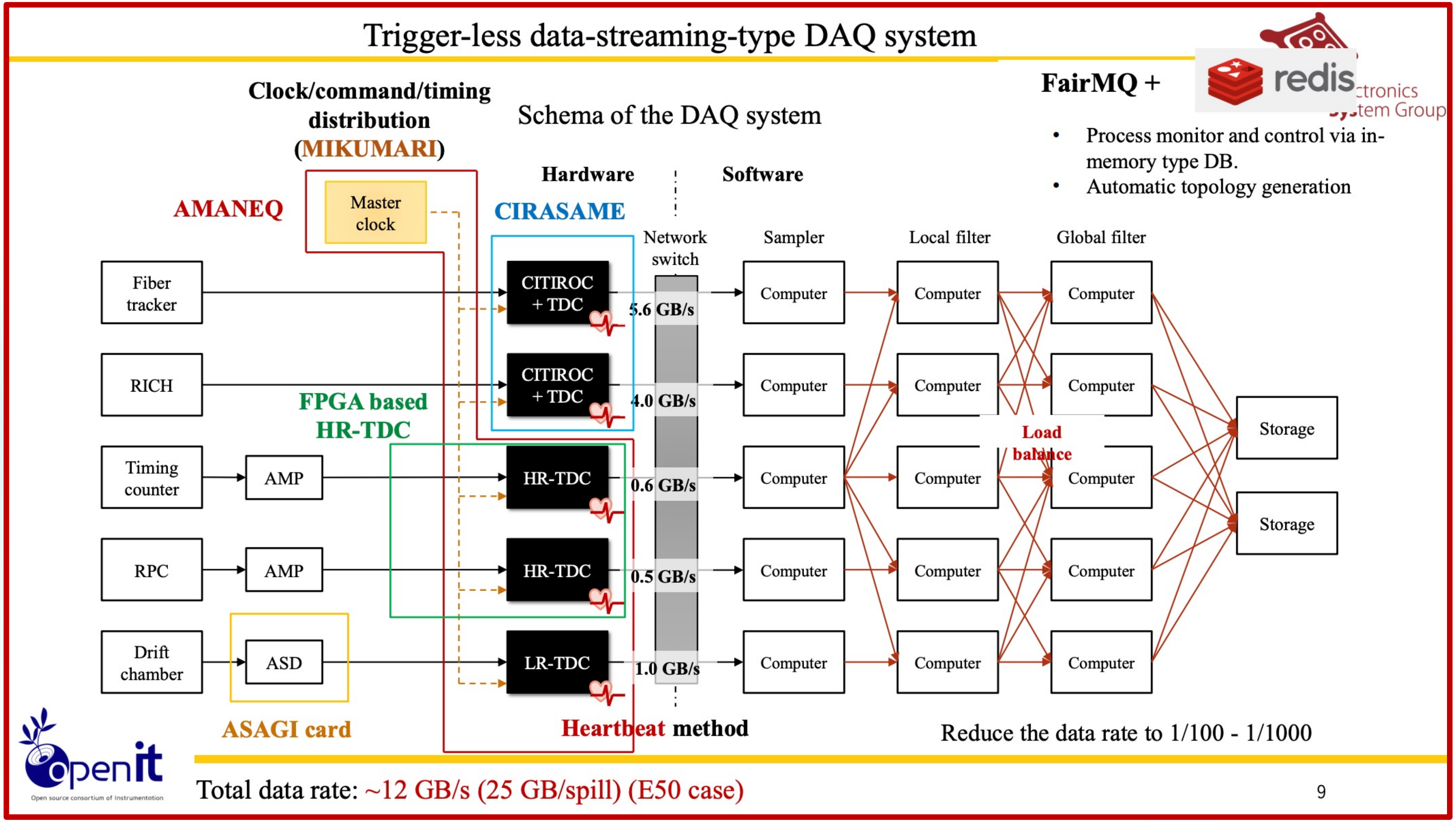
実験	ビーム粒子	運動量 [GeV/c]	強度 [MHz]
Charmed baryon spectroscopy (E50)	π^-	20	30
Non-strange di-baryon search (E79)	proton	2.85–4.00	> 1
Ξ baryon spectroscopy (E97)	K^-	5–8	> 0.5
Ω baryon spectroscopy (P85)	K^-	7–10	> 0.5
ϕ Meson production experiment (P95)	π^-	1.8–2.4	> 0.5
Exclusive Drell-Yan experiment (LoI)	π^-	15	30
Λ–p scattering experiment (LoI)	π^-	8.5	30
Double-K mesonic-nucleus production (LoI)	proton	8	30
Λ(1450) and Quark-counting rule	π^-	5–10	> 0.5

E50 will deploy trigger-less DAQ system.

- Software data filtering and software trigger



J-PARC E50



Trigger-less data-streaming-type DAQ system

FairMQ + redis

- Process monitor and control via in-memory type DB.
- Automatic topology generation

Schema of the DAQ system

Clock/command/timing distribution (MIKUMARI)

Hardware | **Software**

AMANEQ

CIRASAME

FPGA based HR-TDC

ASAGI card

Heartbeat method

Load balance

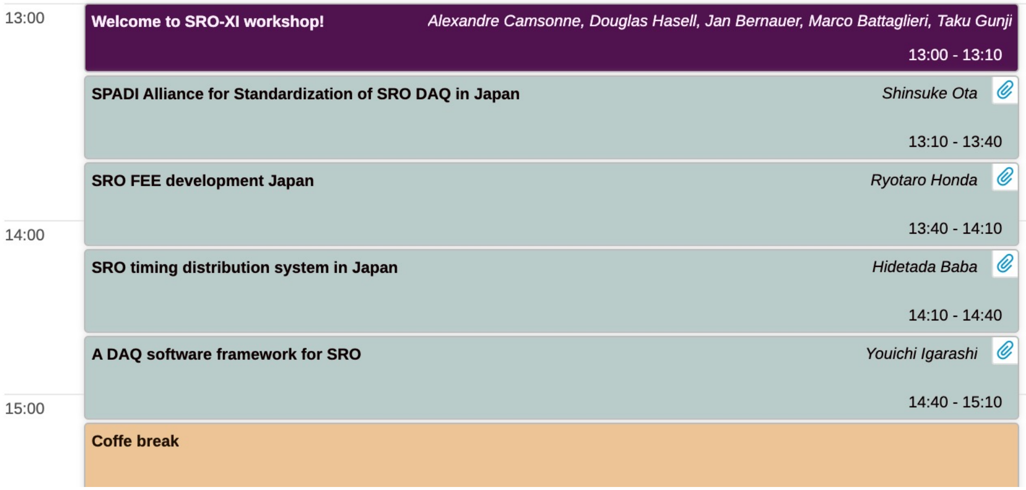
Reduce the data rate to 1/100 - 1/1000

Total data rate: ~12 GB/s (25 GB/spill) (E50 case)

Backup II (from APS/JPS meeting)

SPADI-Alliance

- Start collaborating with US
  - DAQ Workshop at APS/JPS joing meeting at Hawaii (2023.11.26-12.2)
  - SRO workshop XI (11.28, 12.2-12.3) (SRO workshop XII or XIII will be in Japan – under discussion)
- We will start making concrete plans on the collaboration with ePIC DAQ/SRO.
  - Online processing, timing distribution system, ASICs
  - J-PARC E50 as a testbed for the streaming readout of ePIC



13:00 Welcome to SRO-XI workshop! *Alexandre Camsonne, Douglas Hasell, Jan Bernauer, Marco Battaglieri, Taku Gunji* 13:00 - 13:10

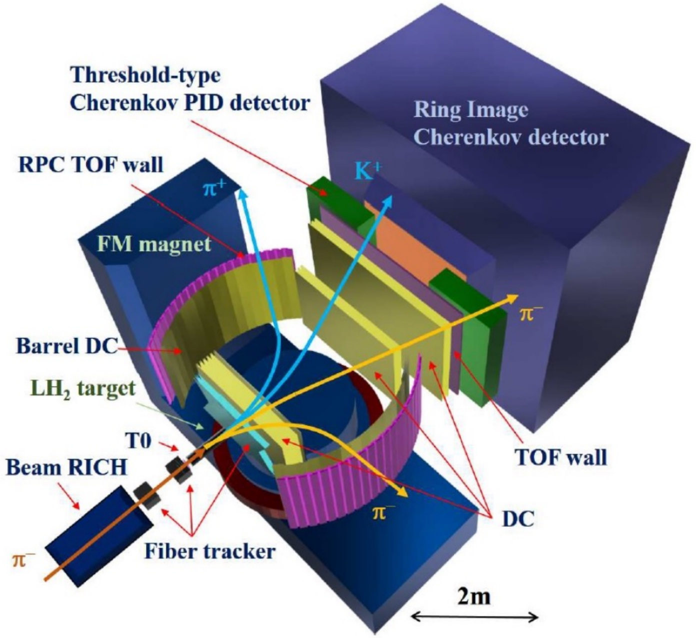
SPADI Alliance for Standardization of SRO DAQ in Japan *Shinsuke Ota* 13:10 - 13:40

SRO FEE development Japan *Ryotaro Honda* 13:40 - 14:10

14:00 SRO timing distribution system in Japan *Hidetada Baba* 14:10 - 14:40

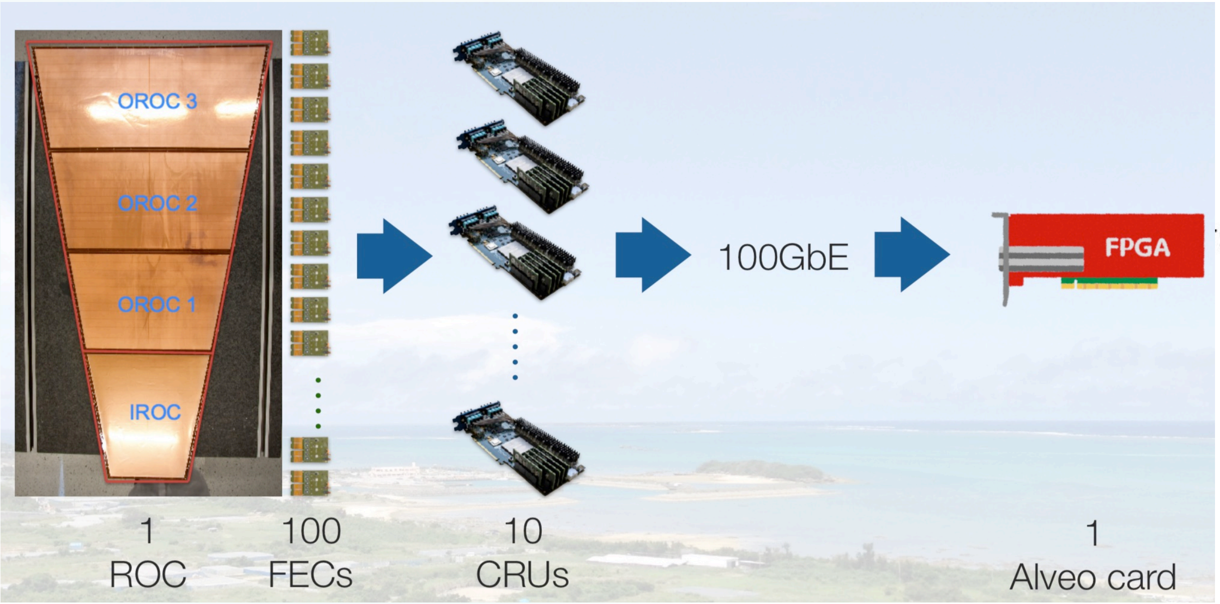
A DAQ software framework for SRO *Youichi Igarashi* 14:40 - 15:10

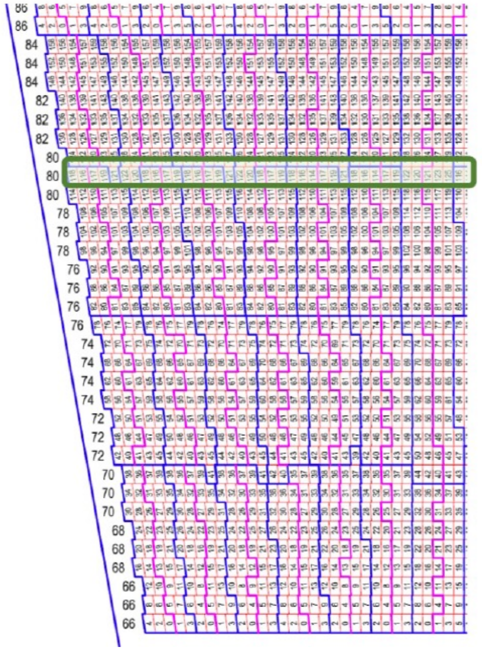
15:00 Coffe break

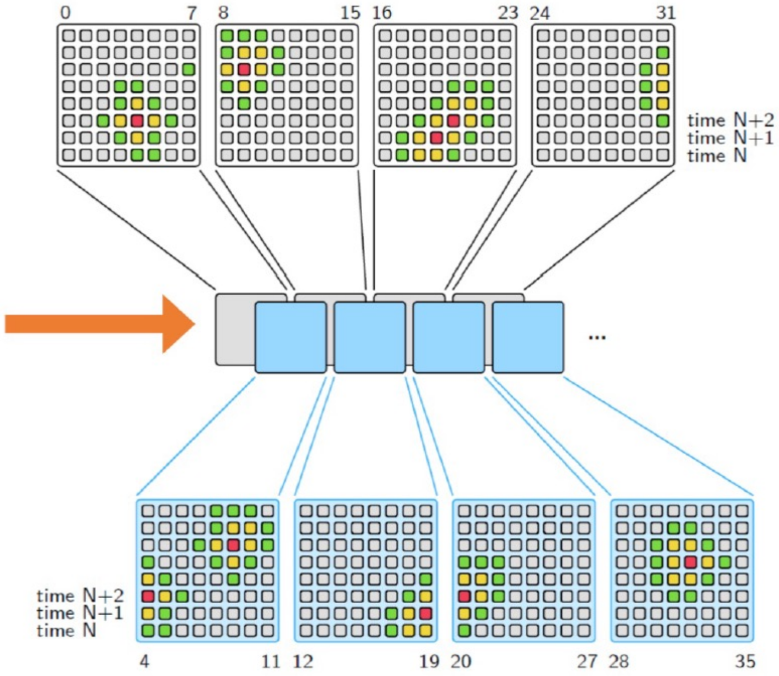


TPC Clustering in FPGA (HLS)

- TPC clustering for one sector (currently running on GPU) can be run in FPGA?
  - Find local maxima in pad-timebin 2D space
  - Scan rectangular region in the pipelined way
- Alveo U55C (VU47P, 3 SLRs, 16GB HBM)



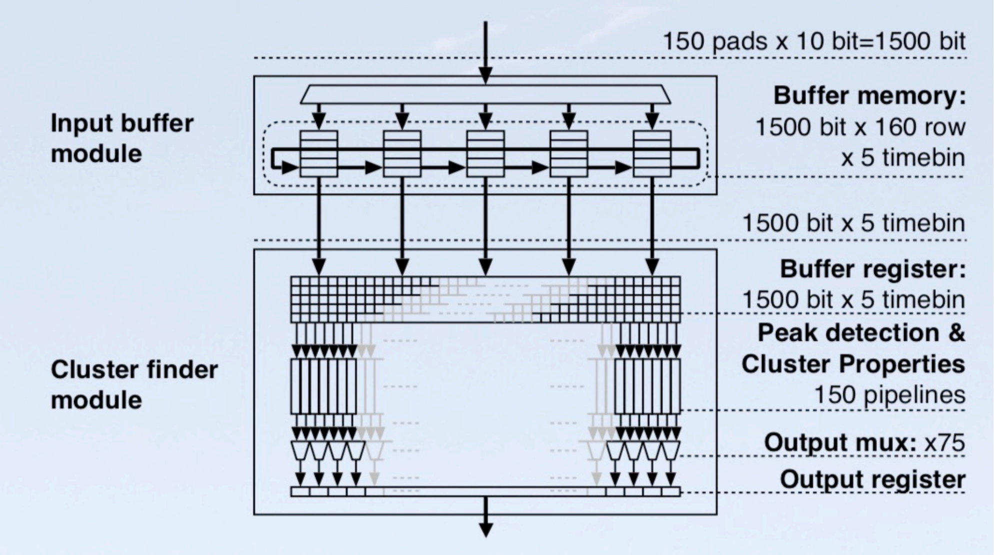




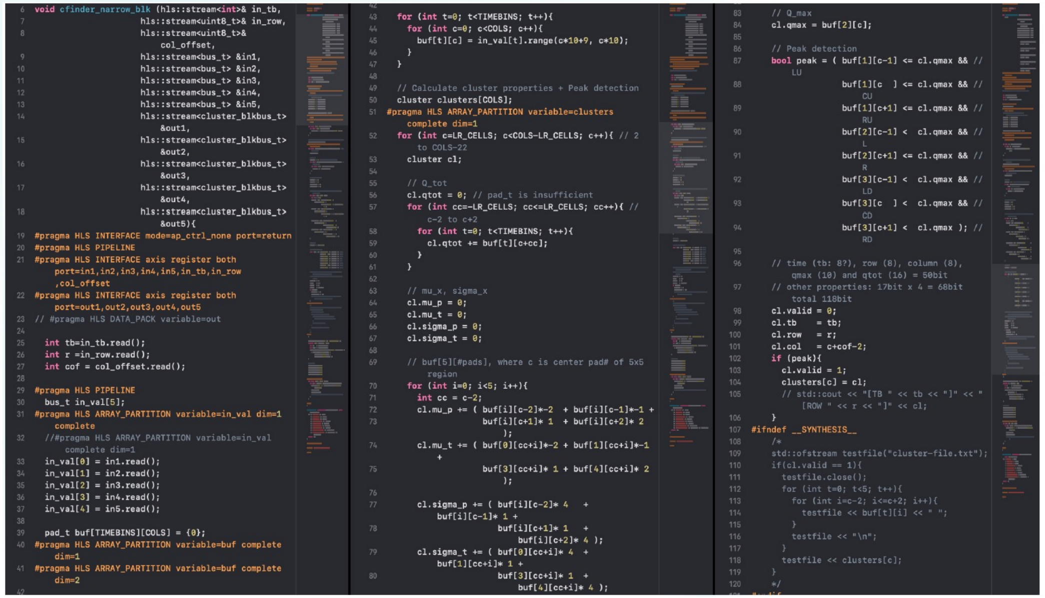
TPC Clustering in FPGA (HLS)

- Data is sent from data source via UDP (100Gbps)
- FPGA logic consists of
  - UDP packet decoder
  - Buffering (150 pads x 10 bits x 160 row x 5 timebins)
  - Clustering (150 pipelines) via pad x timebin [HLS]
  - Output to PCIe

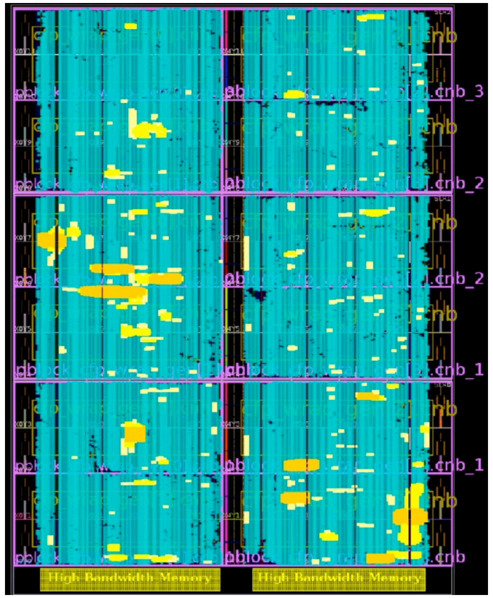
200ns/timebin



150 pads x 10 bit=1500 bit
Input buffer module
Buffer memory: 1500 bit x 160 row x 5 timebin
1500 bit x 5 timebin
Buffer register: 1500 bit x 5 timebin
Cluster finder module
Peak detection & Cluster Properties 150 pipelines
Output mux: x75
Output register



```cpp
void cfinder_narrow_blk (hls::stream<int>& in_tb,
                         hls::stream<uint8_t>& in_row,
                         hls::stream<uint8_t>& col_offset,
                         hls::stream<bus_t> &in1,
                         hls::stream<bus_t> &in2,
                         hls::stream<bus_t> &in3,
                         hls::stream<bus_t> &in4,
                         hls::stream<bus_t> &in5,
                         hls::stream<cluster_blkbus_t> &out1,
                         hls::stream<cluster_blkbus_t> &out2,
                         hls::stream<cluster_blkbus_t> &out3,
                         hls::stream<cluster_blkbus_t> &out4,
                         hls::stream<cluster_blkbus_t> &out5){
#pragma HLS INTERFACE mode=ap_ctrl_none port=return
#pragma HLS PIPELINE
#pragma HLS INTERFACE axis register both port=in1,in2,in3,in4,in5,in_tb,in_row,col_offset
#pragma HLS INTERFACE axis register both port=out1,out2,out3,out4,out5
// #pragma HLS DATA_PACK variable=out

  int tb=in_tb.read();
  int r =in_row.read();
  int cof = col_offset.read();

#pragma HLS PIPELINE
  bus_t in_val[5];
#pragma HLS ARRAY_PARTITION variable=in_val dim=1 complete
  //#pragma HLS ARRAY_PARTITION variable=in_val complete dim=1
  in_val[0] = in1.read();
  in_val[1] = in2.read();
  in_val[2] = in3.read();
  in_val[3] = in4.read();
  in_val[4] = in5.read();

  pad_t buf[TIMEBINS][COLS] = {0};
#pragma HLS ARRAY_PARTITION variable=buf complete dim=1
#pragma HLS ARRAY_PARTITION variable=buf complete dim=2

  for (int t=0; t<TIMEBINS; t++){
    for (int c=0; c<COLS; c++){
      buf[t][c] = in_val[t].range(c*10+9, c*10);
    }
  }

  // Calculate cluster properties + Peak detection
  cluster clusters[COLS];
#pragma HLS ARRAY_PARTITION variable=clusters complete dim=1
  for (int c=LR_CELLS; c<COLS-LR_CELLS; c++){ // 2 to COLS-22
    cluster cl;

    // Q_tot
    cl.qtot = 0; // pad_t is insufficient
    for (int cc=-LR_CELLS; cc<=LR_CELLS; cc++){ // c-2 to c+2
      for (int t=0; t<TIMEBINS; t++){
        cl.qtot += buf[t][c+cc];
      }
    }

    // mu_x, sigma_x
    cl.mu_p = 0;
    cl.mu_t = 0;
    cl.sigma_p = 0;
    cl.sigma_t = 0;

    // buf[5][#pads], where c is center pad# of 5x5 region
    for (int i=0; i<5; i++){
      int cc = c-2;
      cl.mu_p += ( buf[i][c-2]*-2  + buf[i][c-1]*-1 +
                   buf[i][c+1]* 1  + buf[i][c+2]* 2 );
      cl.mu_t += ( buf[0][cc+i]*-2 + buf[1][cc+i]*-1 +
                   buf[3][cc+i]* 1 + buf[4][cc+i]* 2 );

      cl.sigma_p += ( buf[i][c-2]* 4   + buf[i][c-1]* 1 +
                      buf[i][c+1]* 1   + buf[i][c+2]* 4 );
      cl.sigma_t += ( buf[0][cc+i]* 4  + buf[1][cc+i]* 1 +
                      buf[3][cc+i]* 1  + buf[4][cc+i]* 4 );

    // Q_max
    cl.qmax = buf[2][c];

    // Peak detection
    bool peak = ( buf[1][c-1] <= cl.qmax && // LU
                  buf[1][c  ] <= cl.qmax && // CU
                  buf[1][c+1] <= cl.qmax && // RU
                  buf[2][c-1] <  cl.qmax && // L
                  buf[2][c+1] <= cl.qmax && // R
                  buf[3][c-1] <  cl.qmax && // LD
                  buf[3][c  ] <  cl.qmax && // CD
                  buf[3][c+1] <  cl.qmax ); // RD

    // time (tb: 8?), row (8), column (8), qmax (10) and qtot (16) = 50bit
    // other properties: 17bit x 4 = 68bit total 118bit
    cl.valid = 0;
    cl.tb    = tb;
    cl.row   = r;
    cl.col   = c+cof-2;
    if (peak){
      cl.valid = 1;
      clusters[c] = cl;
      // std::cout << "[TB " << tb << "]" << "[ROW " << r << "]" << cl;
    }
#ifndef __SYNTHESIS__
    /*
    std::ofstream testfile("cluster-file.txt");
    if(cl.valid == 1){
      testfile.close();
      for (int t=0; t<5; t++){
        for (int i=c-2; i<=c+2; i++){
          testfile << buf[t][i] << " ";
        }
        testfile << "\n";
      }
      testfile << clusters[c];
    }
    */
```



- CRU 0-2: 48 rows
- CRU 3-5: 49 rows
- CRU 6-9: 55 rows

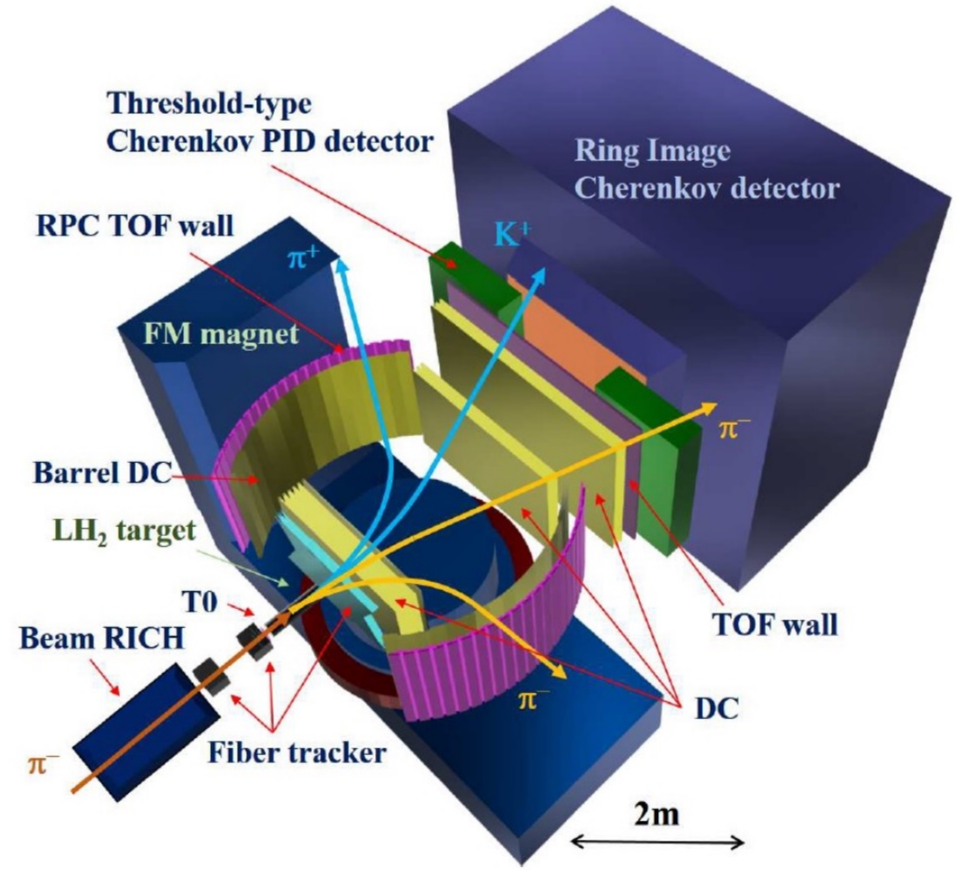
We achieved to run at 290 MHz

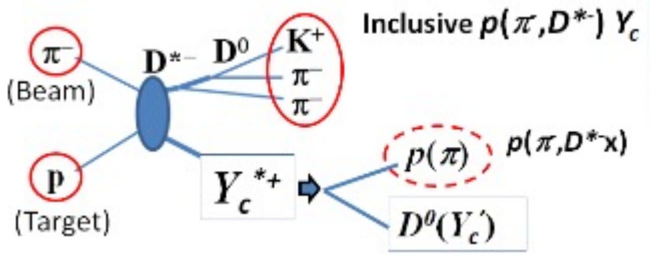
> 55 rows x 5 MSPS = 275 MHz

Clustering for one sector can run without latencies

	Used	Available	%
LUT	714,307	1,303,680	54.8
FF	814,523	2,607,360	31.2
BRAM	846.5	2,016	42.0

J-PARC : E50 experiment





E50 : General detector system at J-PARC high-momentum beamline for various physics topics

実験	ビーム粒子	運動量 [GeV/c]	強度 [MHz]
Charmed baryon spectroscopy (E50)	π^-	20	30
Non-strange di-baryon search (E79)	proton	2.85–4.00	> 1
Ξ baryon spectroscopy (E97)	K^-	5–8	> 0.5
Ω baryon spectroscopy (P85)	K^-	7–10	> 0.5
ϕ Meson production experiment (P95)	π^-	1.8–2.4	> 0.5
Exclusive Drell-Yan experiment (LoI)	π^-	15	30
Λ–p scattering experiment (LoI)	π^-	8.5	30
Double-K mesonic-nucleus production (LoI)	proton	8	30
Λ(1450) and Quark-counting rule	π^-	5–10	> 0.5

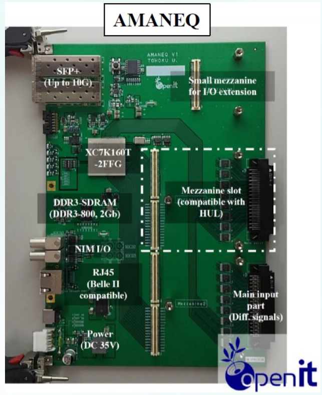
E50 will deploy trigger-less DAQ system.

- Software data filtering and software trigger for efficient usage of hadron beams at J-PARC

E50 Streaming readout

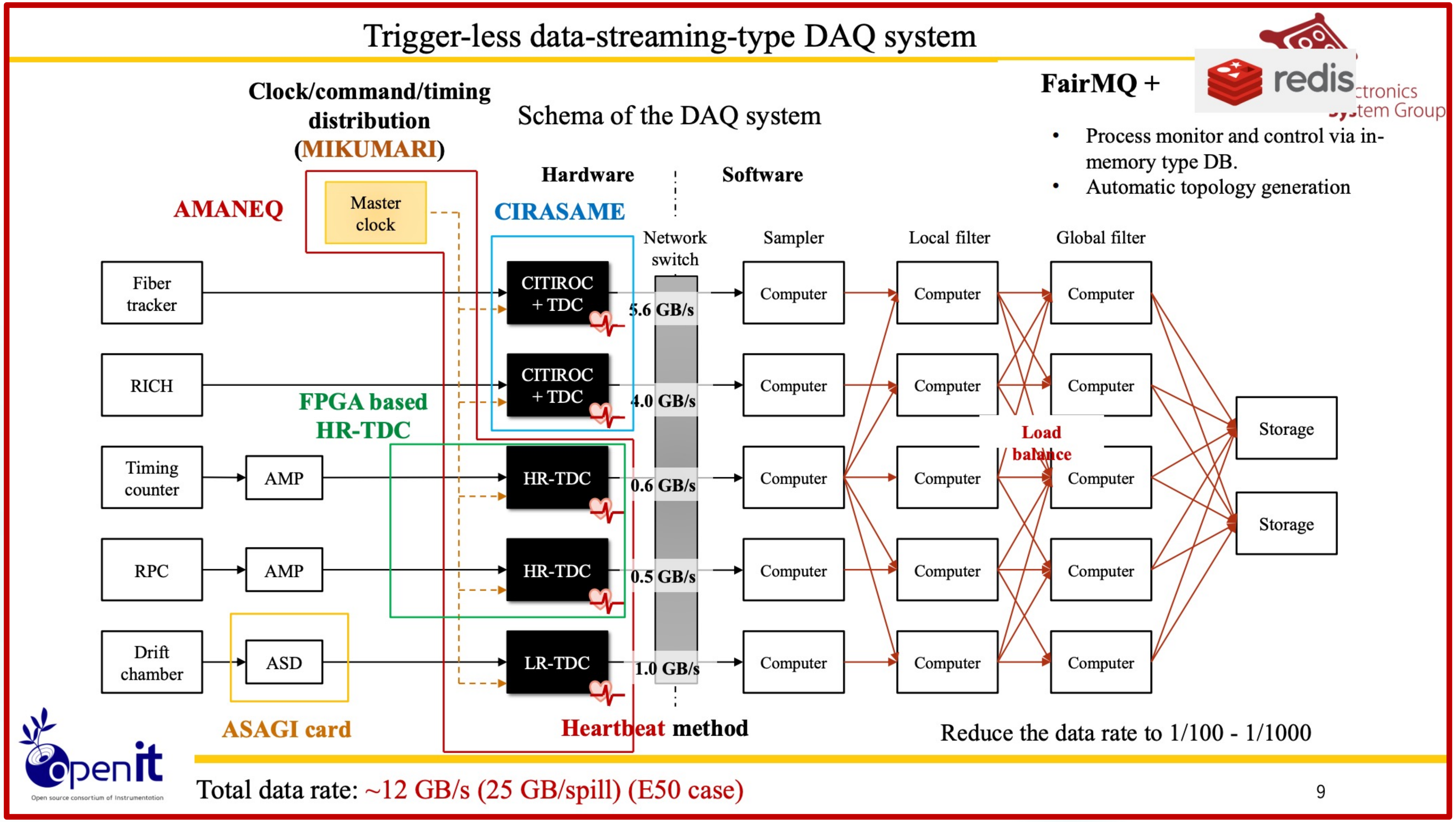
CIRASAME

AMANEQ


SFP+ (Up to 10G)
Small mezzanine for I/O extension
XC7K160T -2FFG
DDR3-SDRAM (DDR3-800, 2Gb)
Mezzanine slot (compatible with HUL)
NIM I/O
RJ45 (Belle II compatible)
Main input part (Diff. signals)
Power (DC 35V)
openit

Honda

Shirotori



Trigger-less data-streaming-type DAQ system

FairMQ + redis

ctronics
stem Group

- Process monitor and control via in-memory type DB.
- Automatic topology generation

Schema of the DAQ system

Clock/command/timing distribution (MIKUMARI)

AMANEQ

Master clock

Hardware

Software

CIRASAME

Network switch

Sampler

Local filter

Global filter

Fiber tracker → CITIROC + TDC → 5.6 GB/s → Computer → Computer → Computer

RICH → CITIROC + TDC → 4.0 GB/s → Computer → Computer → Computer

FPGA based HR-TDC

Timing counter → AMP → HR-TDC → 0.6 GB/s → Computer → Computer → Computer

RPC → AMP → HR-TDC → 0.5 GB/s → Computer → Computer → Computer

Drift chamber → ASD → LR-TDC → 1.0 GB/s → Computer → Computer → Computer

Load balance

Storage

Storage

ASAGI card

Heartbeat method

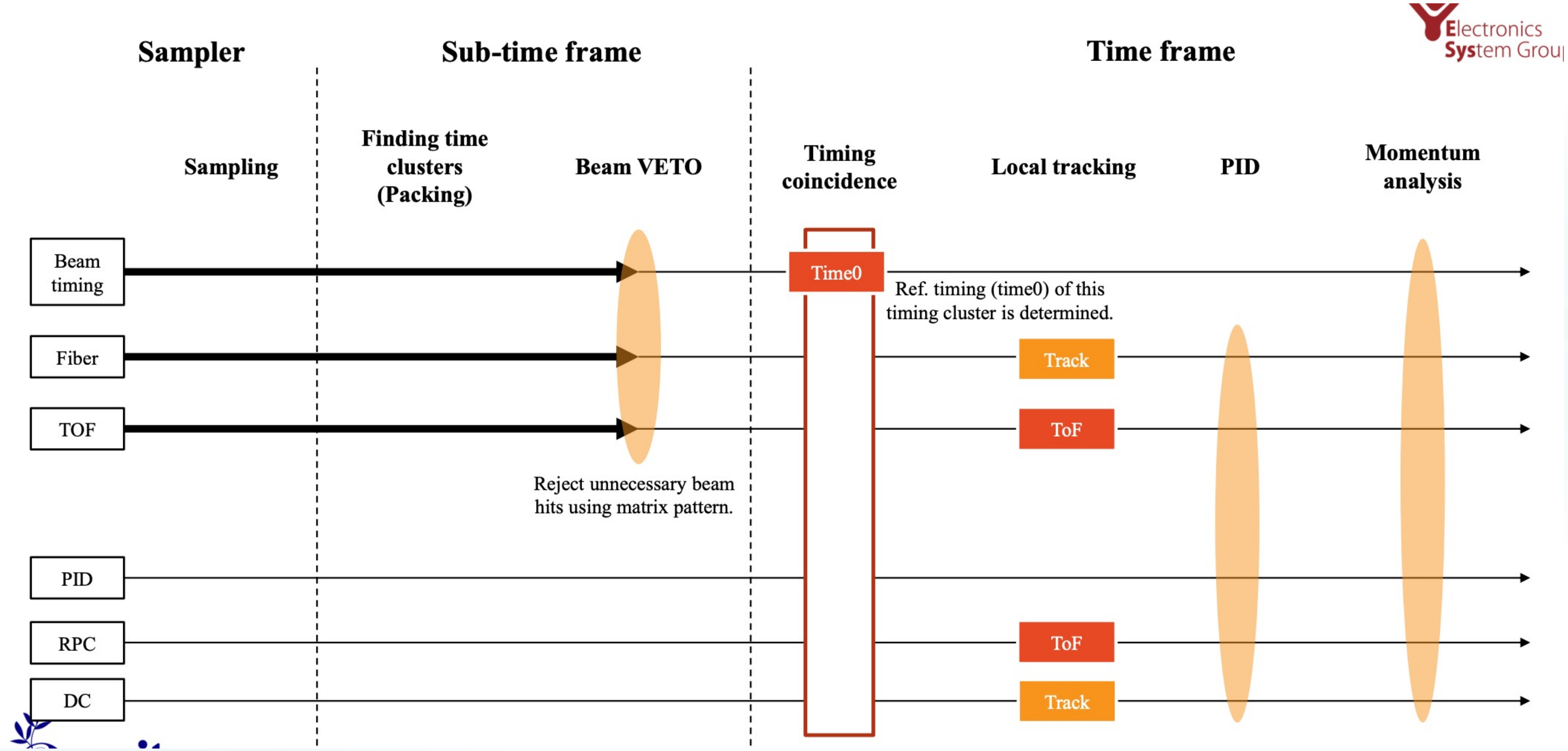
Reduce the data rate to 1/100 - 1/1000

openit
Open source consortium of Instrumentation

Total data rate: ~12 GB/s (25 GB/spill) (E50 case)

9

E50 Streaming readout



Ref. timing (time0) of this timing cluster is determined.

Reject unnecessary beam hits using matrix pattern.

FPGA Processing

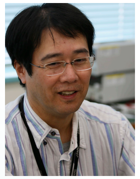
- Time walk correction, Timing offset
- Energy calibration
- Clustering

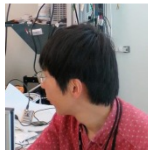
GPU processing

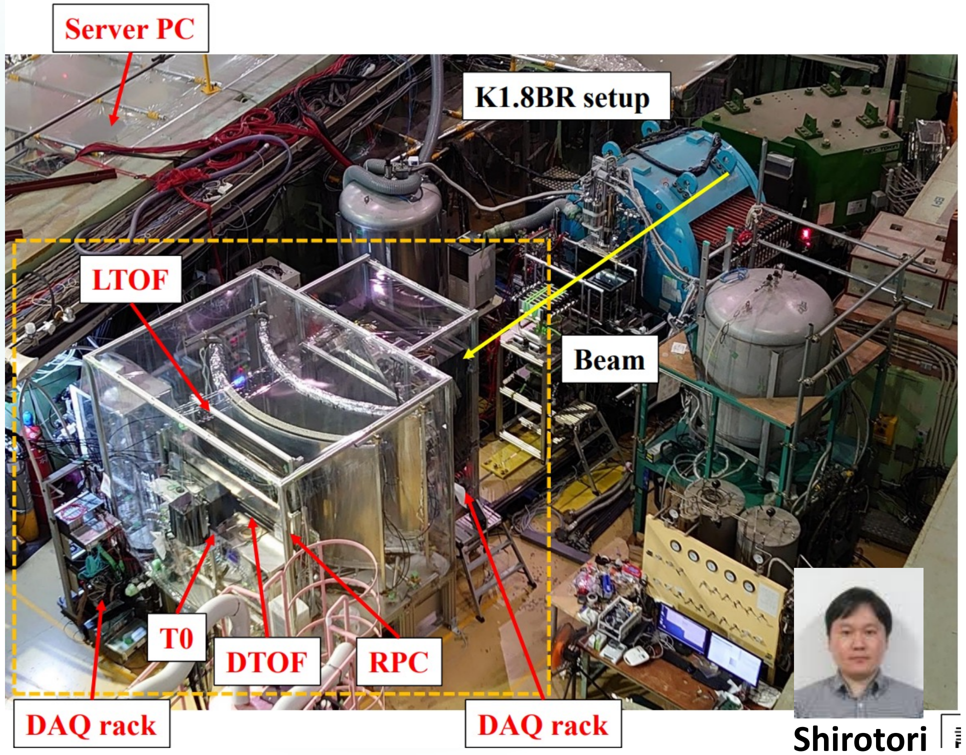
- Tracking, TOF calculation, Particle identification
- Momentum analysis
- Missing mass/Invariant mass

E50 test bench

- Beamtime in last June (19.6-21.6)
  - K^- ~ 200k/spill (2sec), π^- ~ 800k/spill
  - Tests of streaming DAQ with “nestDAQ” and online coincidence filtering (based on CPU)

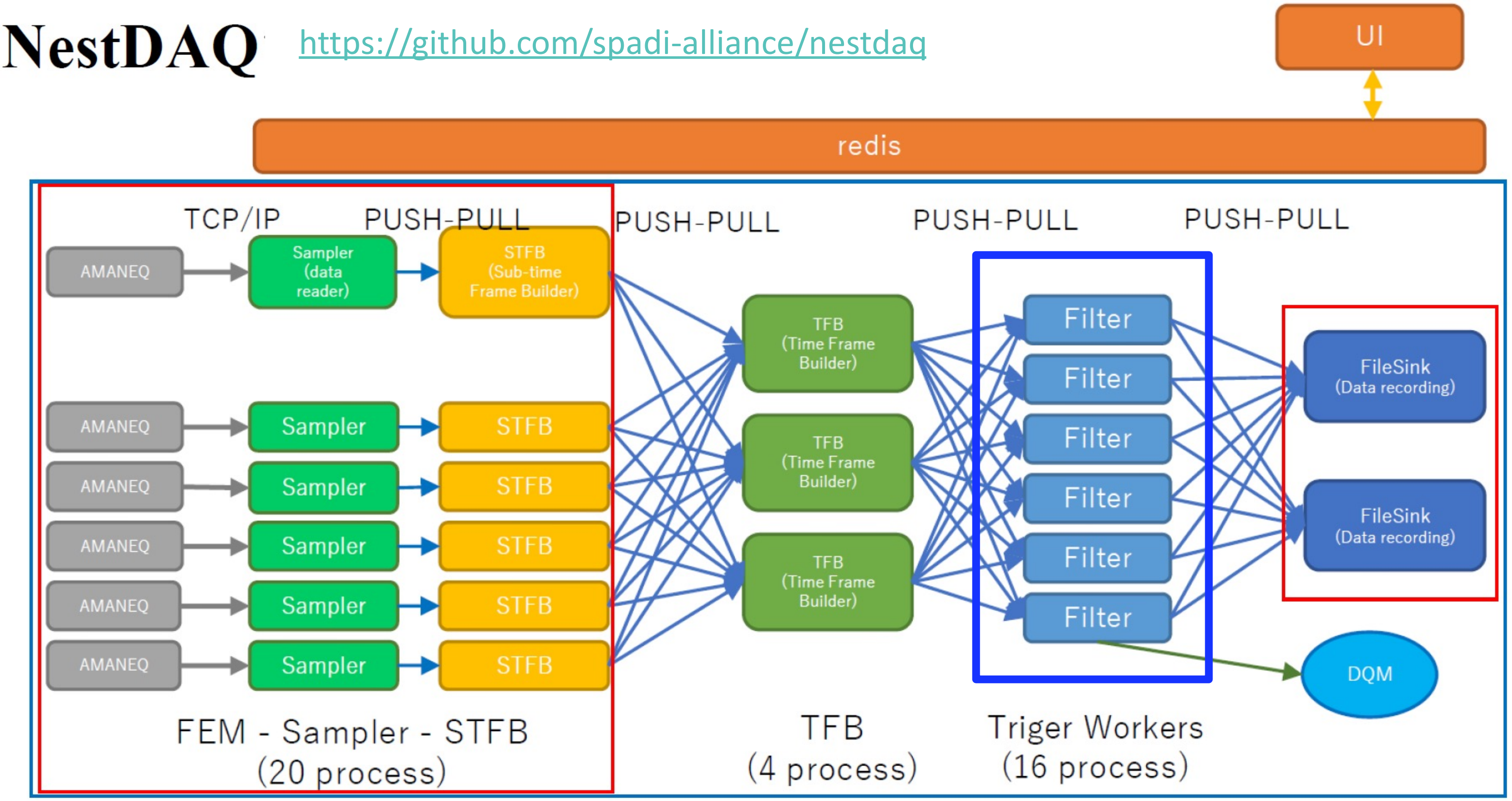

Igarashi


Takahashi



Shirotori

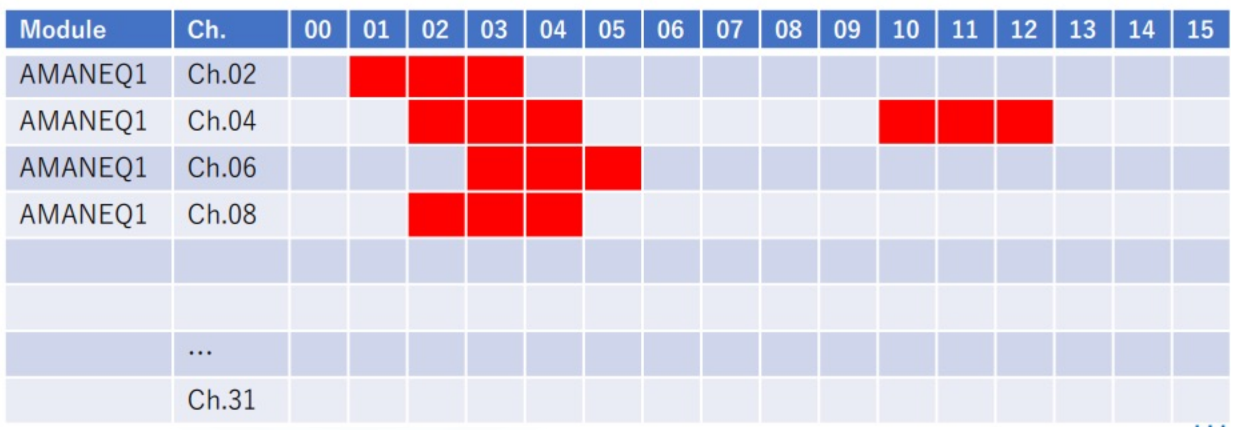
NestDAQ https://github.com/spadi-alliance/nestdaq



Online coincidence filter

General Purpose Logic Filter

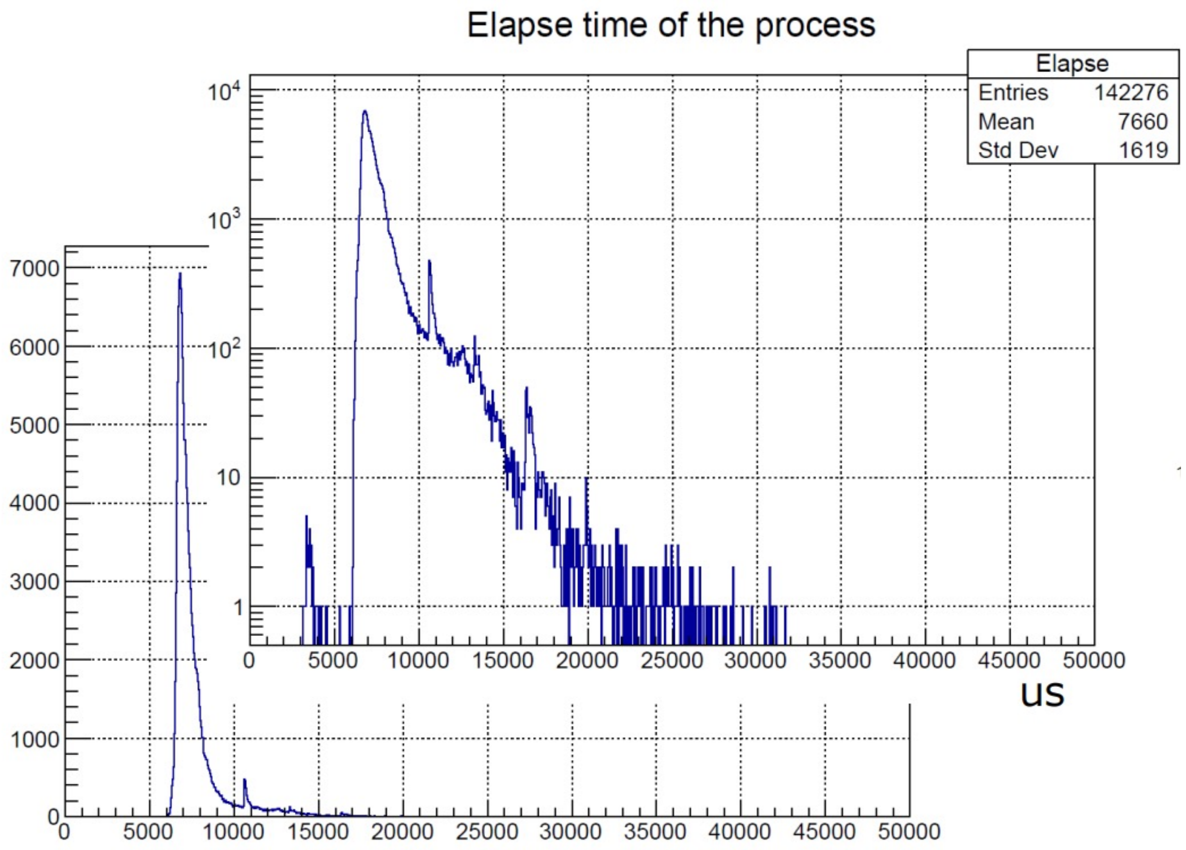
- Create LUT https://github.com/spadi-alliance/nestdaq-user-impl
- Make hit markers (every 4nsec – from TDC) in the array for Heart-beat frame (524usec)
- Scan the array and see if all entries are fired (LUT = true)
- **Running on CPU (will be ported in GPU)**



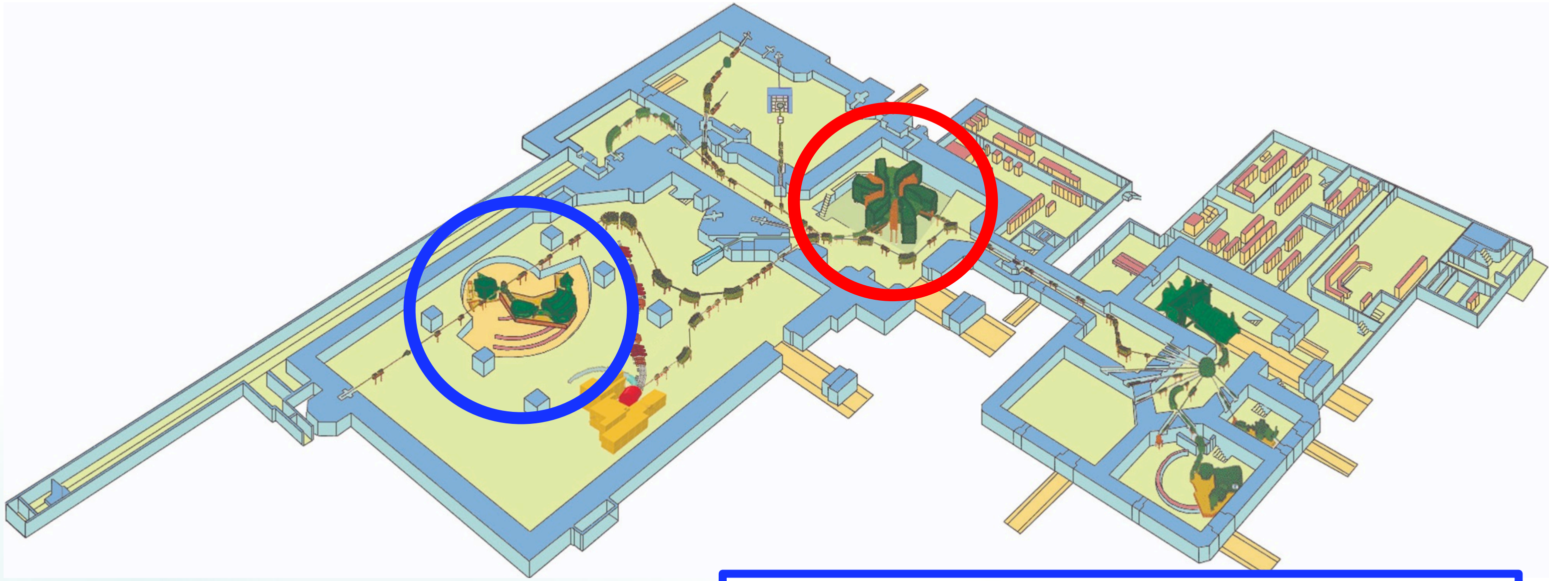
Module	Ch.	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
AMANEQ1	Ch.02		🟥	🟥	🟥												
AMANEQ1	Ch.04			🟥	🟥	🟥						🟥	🟥	🟥			
AMANEQ1	Ch.06				🟥	🟥	🟥										
AMANEQ1	Ch.08			🟥	🟥	🟥											
	…																
	Ch.31																

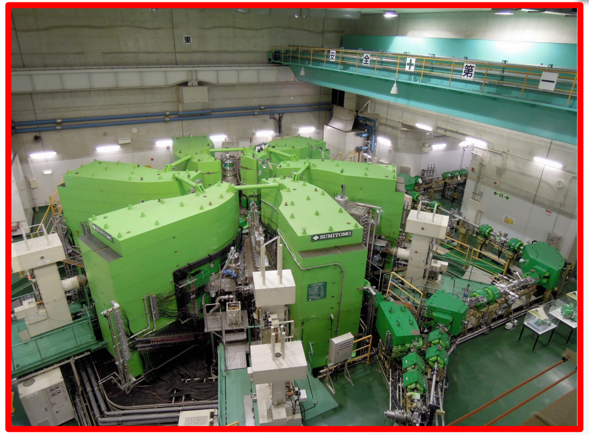
Processing time for 1 HBF (524us)

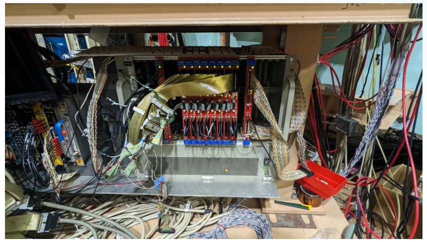
- Average 7.6msec -> 15 processors

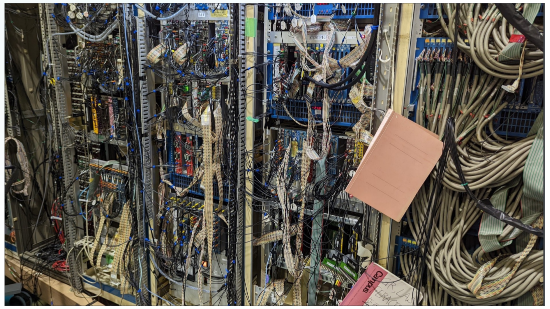


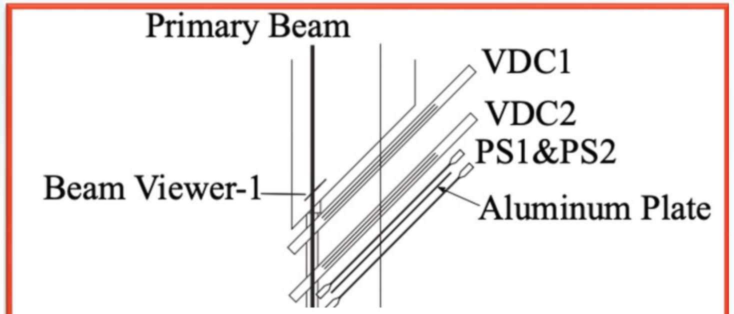
RCNP and Streaming DAQ







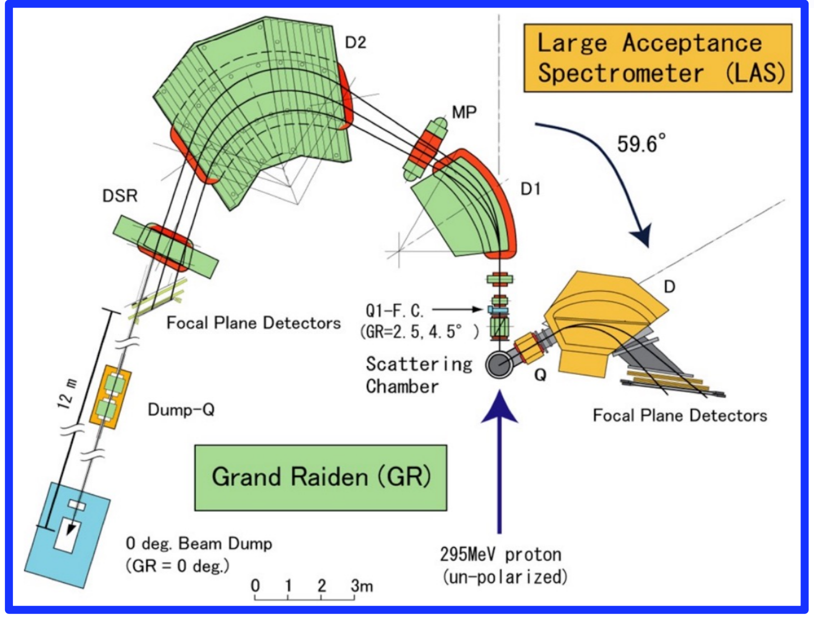




VDC (X1:192, U1:208, X2:192, U2:208)

PS (PS1: 2, PS2: 2)

RF timing, Current integrator

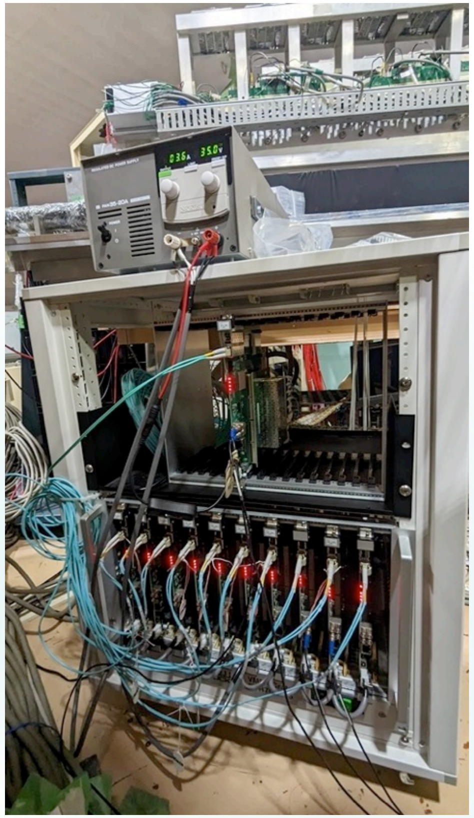


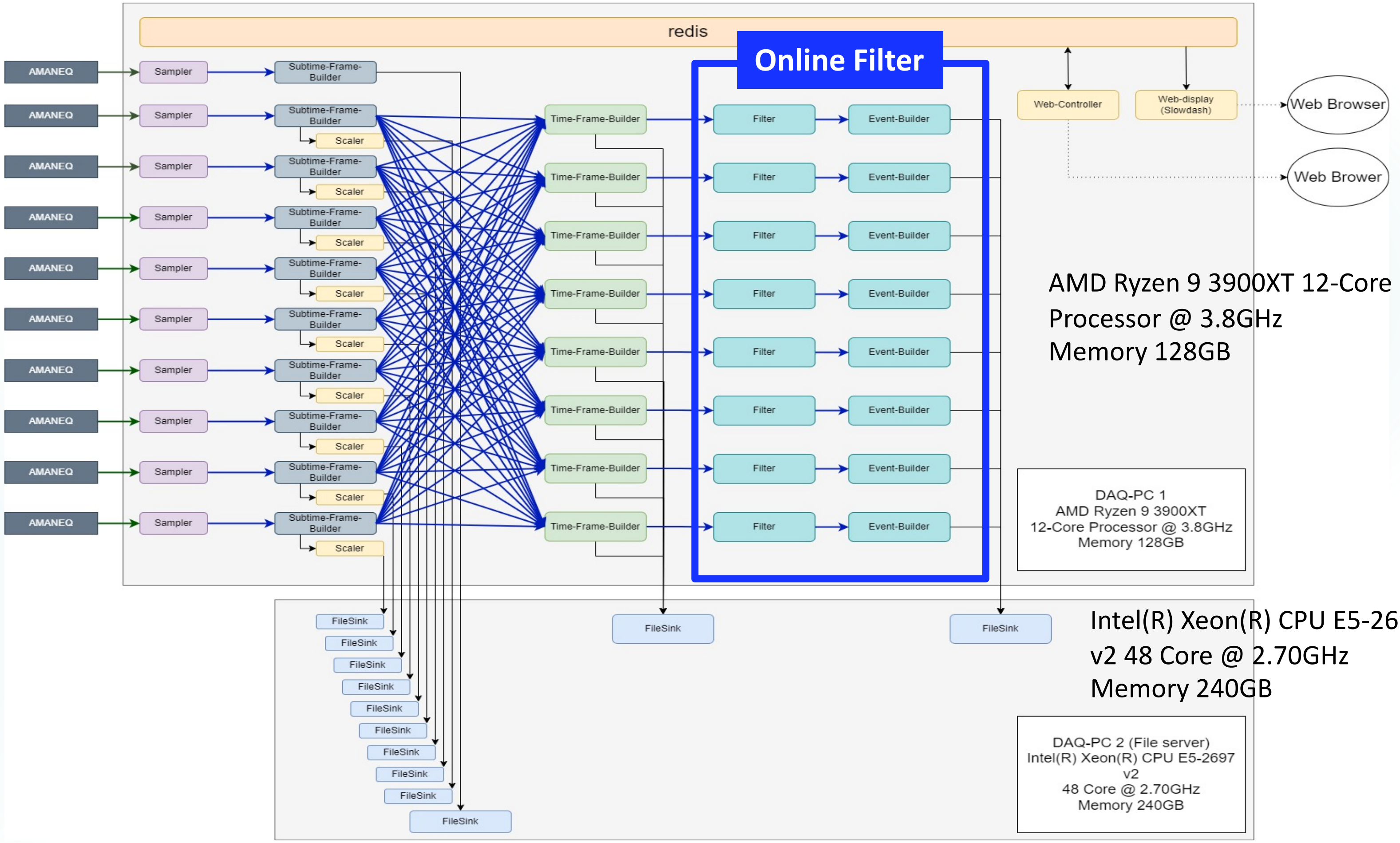
Using multi-event buffering in VME / CAMAC modules, readout is limited up to 50kHz but DAQ rate is still limited by the DAQ busy.

We aim at reading up to O(100kHz) - 1MHz using streaming DAQ.

Physics runs using online filters

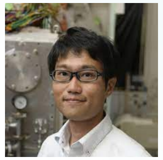
- Physics runs at GR beamline using nest DAQ





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

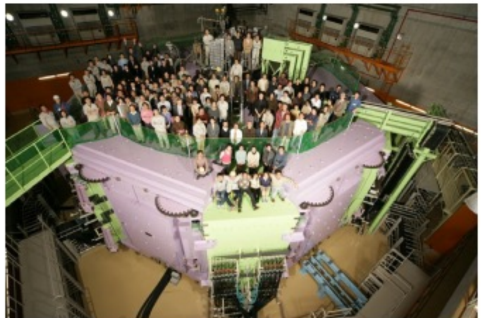
Intel(R) Xeon(R) CPU E5-2697 v2 48 Core @ 2.70GHz
Memory 240GB


Ota

Kobayashi

Throughput 200Mbps (x40 improvement from past DAQ system). 100% efficient for 100-200 kcps
Next step is to implement online tracking, multi-hit identification (ex (d, 2p)), and PID (ex, (α, ^{6}He))

RIBF and Streaming DAQ

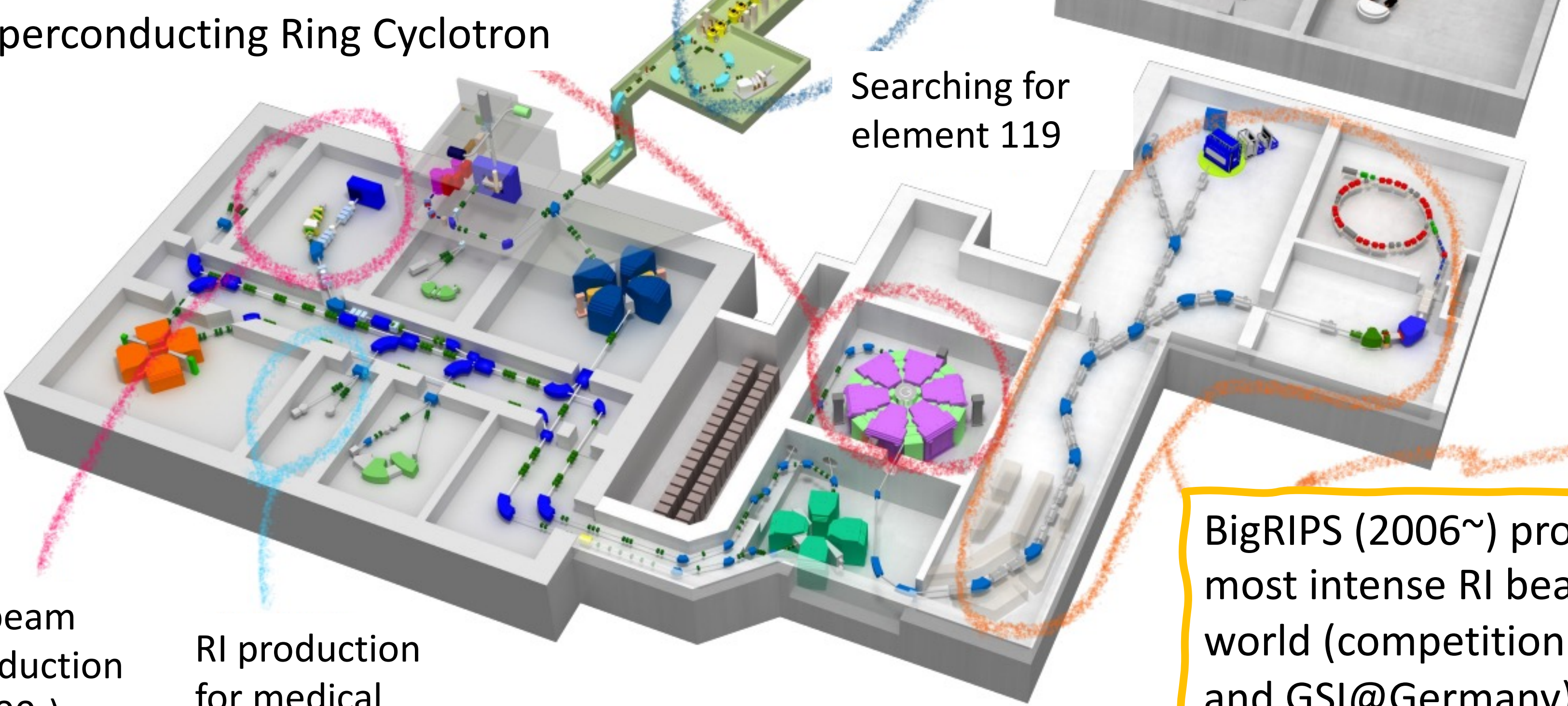


Superconducting Ring Cyclotron

Nihonium

113
Nh
nihonium

RIBF is a next-generation heavy-ion research facility to establish the ultimate picture of nuclei and thus propel forward our understanding of how heavy elements were first formed in the universe.



Searching for element 119

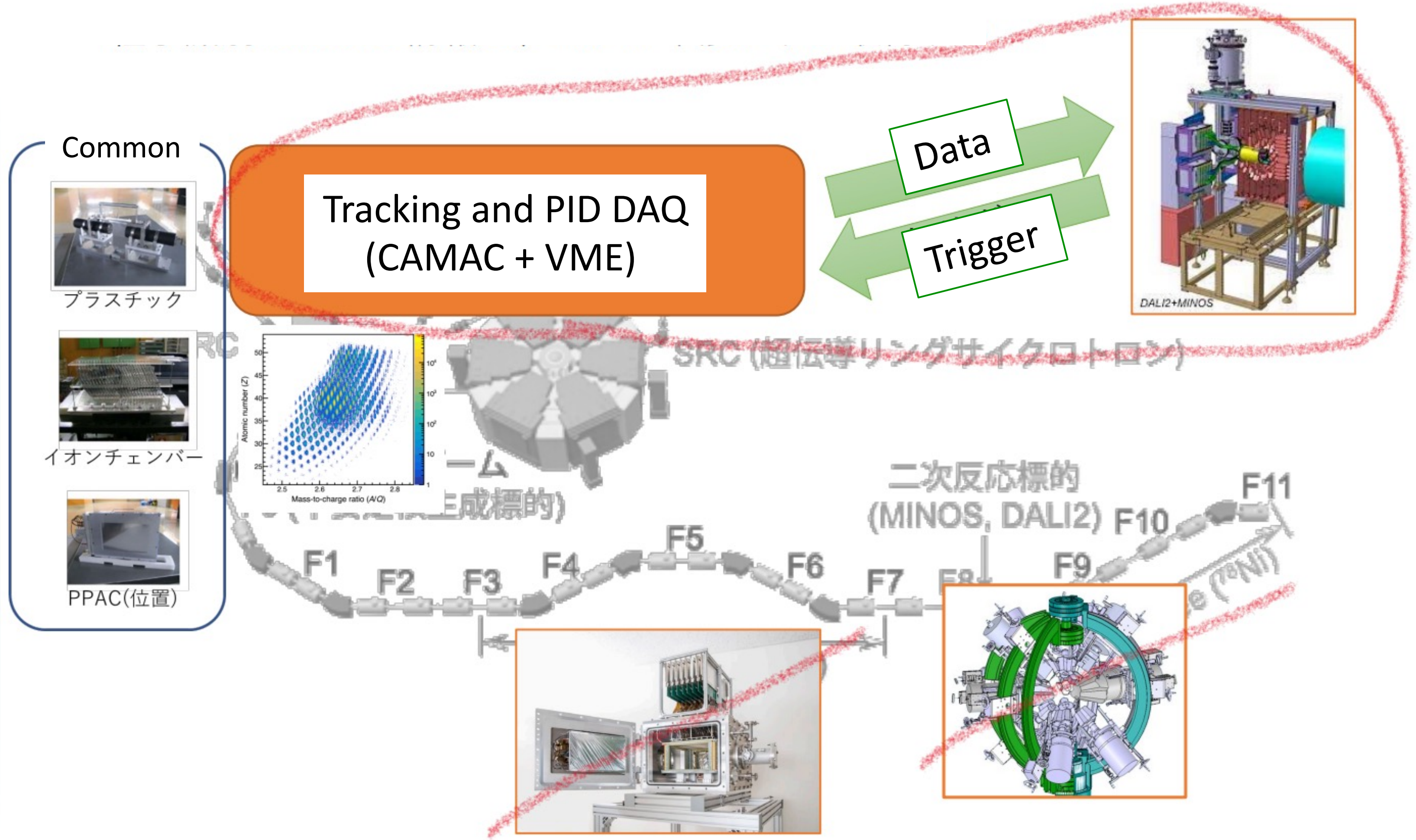
RI beam production (1990-)

RI production for medical purpose

BigRIPS (2006~) provides the most intense RI beam in the world (competition with FRIB@US and GSI@Germany)

Current RIBF running strategies

- **Common beamline detectors (Plastic, ion chamber, PPAC) are occupied by only one experiment**
- This should be shared with other experiments without any interferences → Streaming DAQ

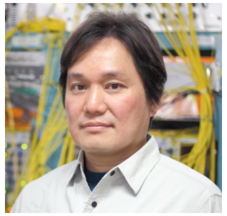


Common

Tracking and PID DAQ
(CAMAC + VME)

Data

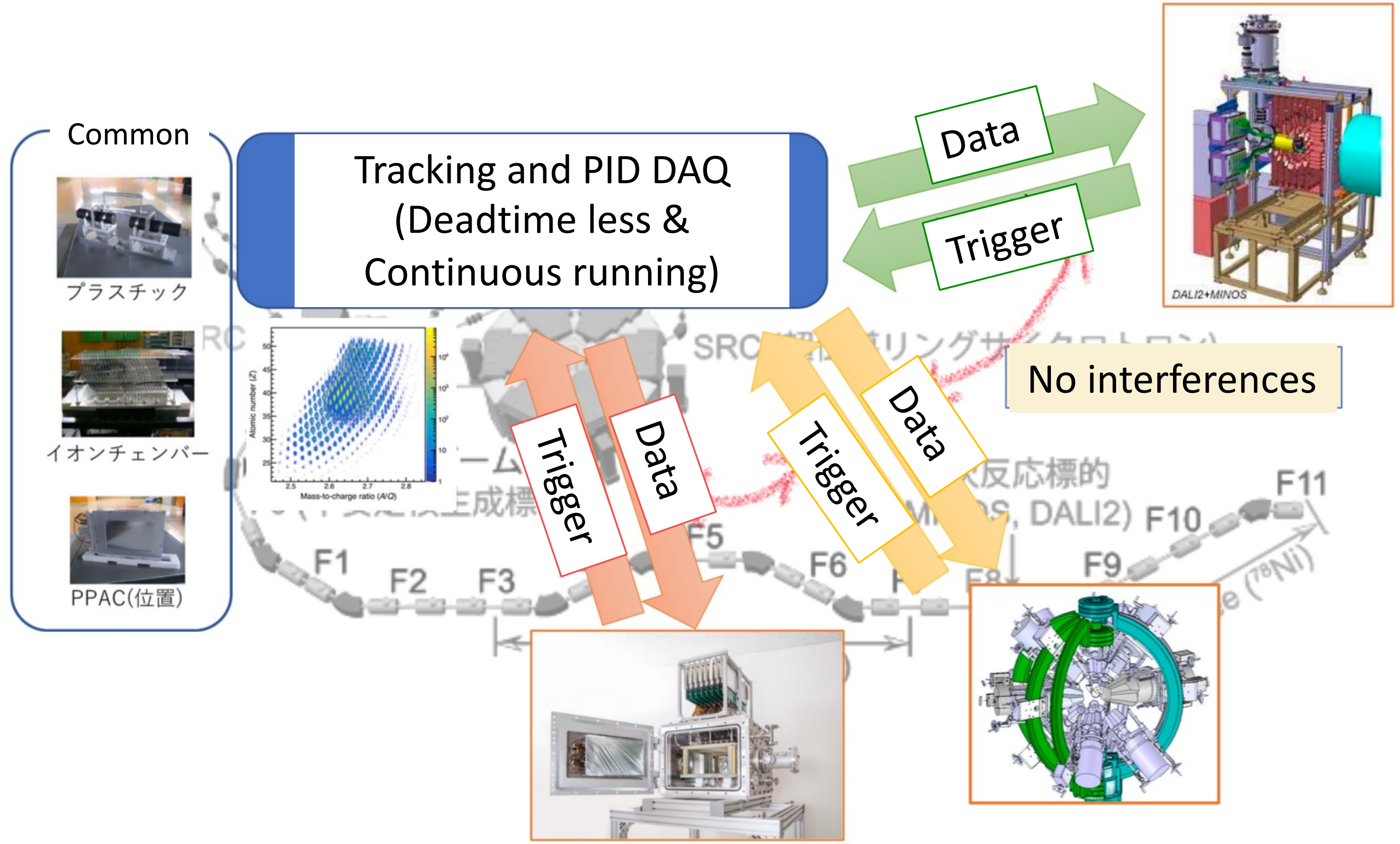
Trigger



Baba

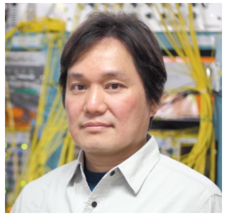
Current RIBF running strategies

- Common beamline detectors (Plastic, ion chamber, PPAC) are occupied by only one experiment
- **This should be shared with other experiments without any interferences → Streaming DAQ**



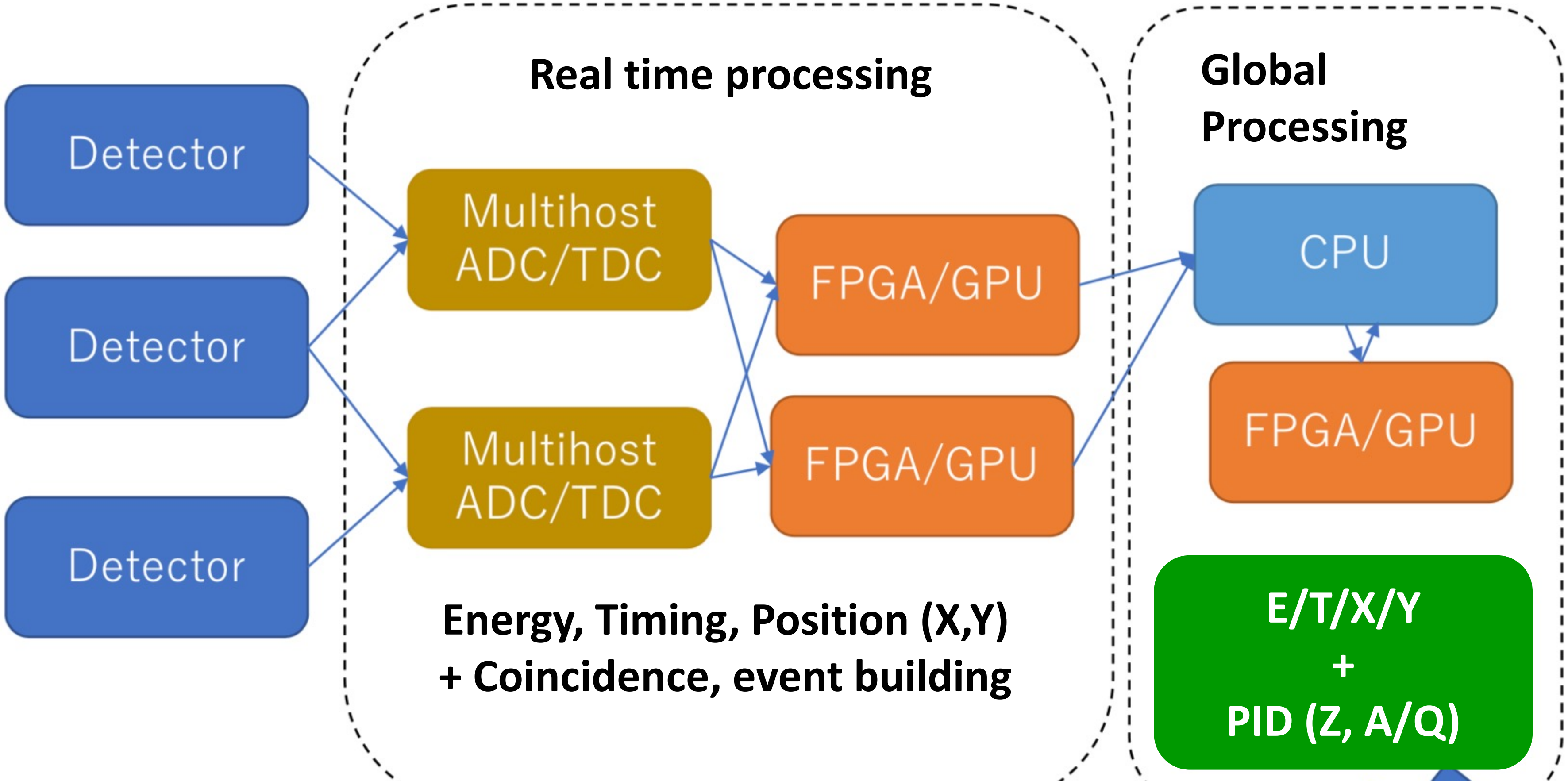
Tracking and PID DAQ
(Deadtime less &
Continuous running)

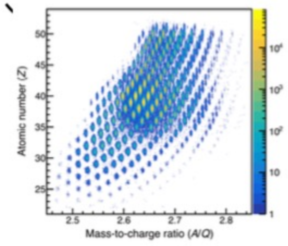
No interferences



Baba

Streaming readout in RIBF

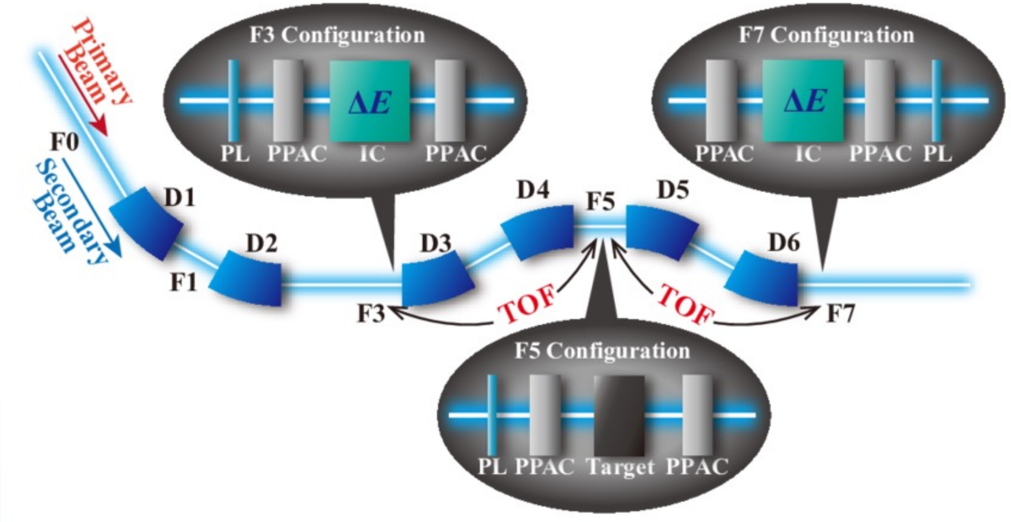




Estimated data rate ~ 10 MB/s
100 MB/s as maximum?

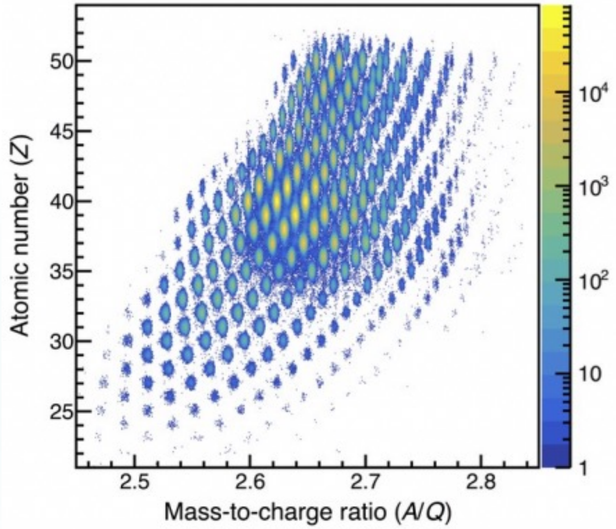
FPGA Processing of BLD data

Ichinohe (C07.4)



BigRIPS

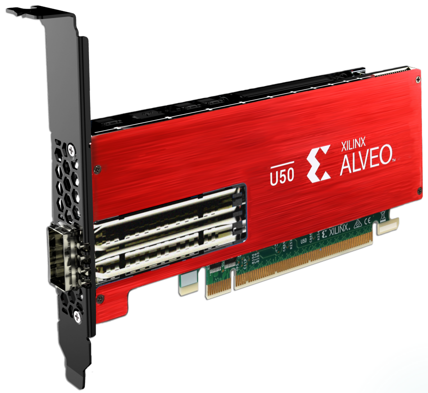
- Plastic (F3, F7) TOF -> β
- PPAC (F3, F5, F7) Tracking -> Bρ
- Ion chamber (F7) dE + β -> Z
- Bρ + β -> A/Q

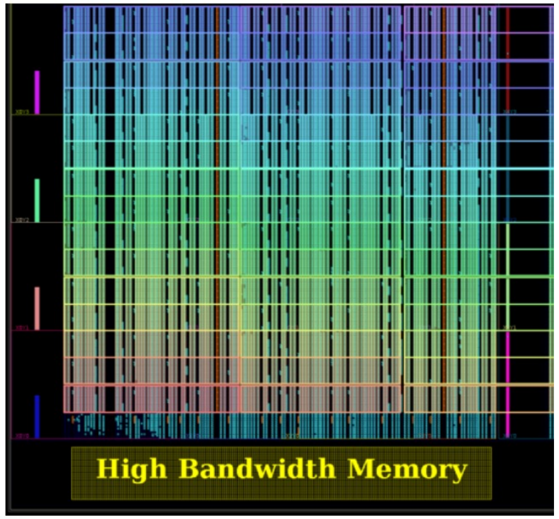


Online process consists of

- PPAC (F3, F5, F7)
  - 4 layers of (x,y) : tracking (angle, position) through chi2 fits
  - Combine F3-F7 to get Bρ
- Plastic (F3, F7)
  - Average timing from 2 PMTs
  - β from F3 ad F7
- Ion chamber (F7)
  - 6 layers / IC : take geometric mean of raw – pedestal
  - Extraction of Z from Bethe-Bloch
- A/Q vs. Z

Alveo U50(8GB HBM)

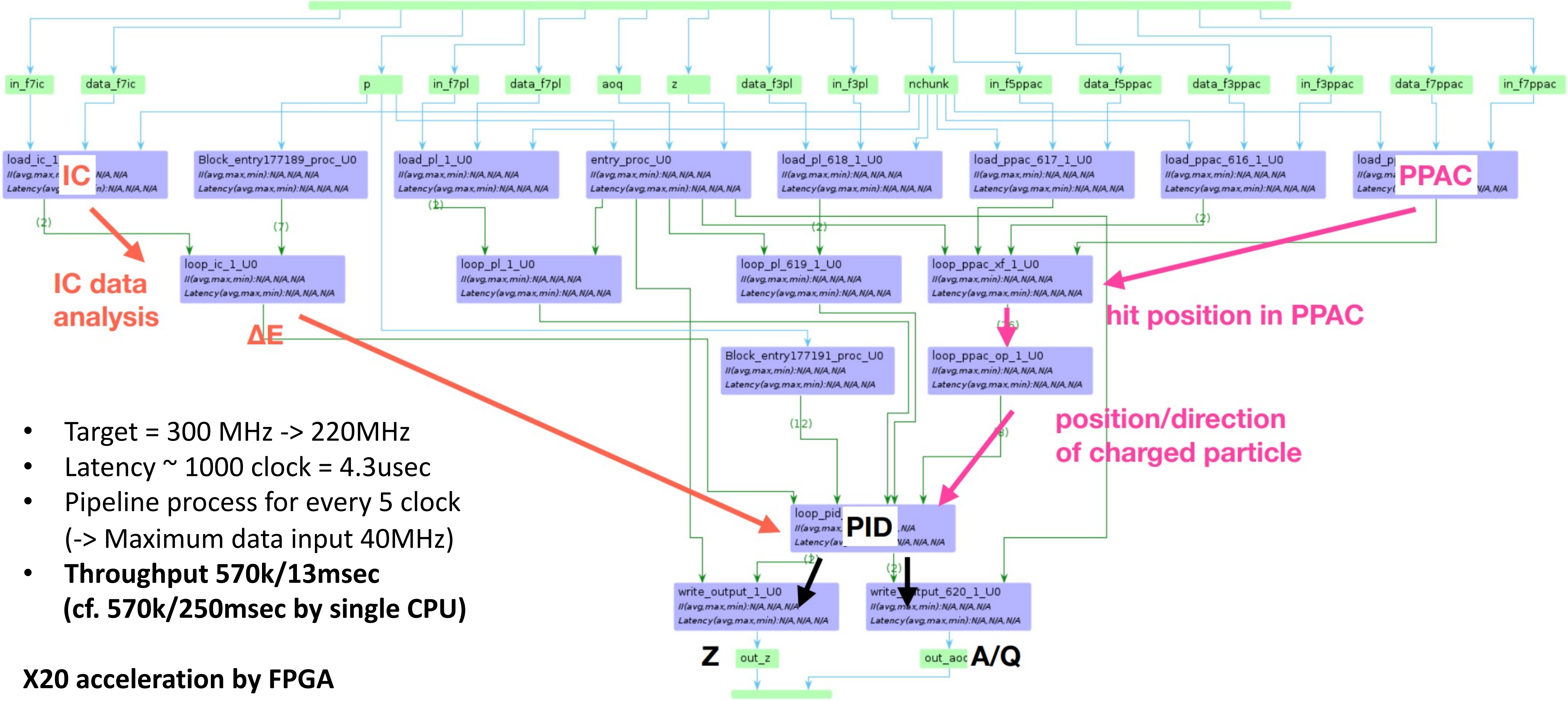




FPGA Processing of BLD data

Ichinohe (C07.4)

Full dataflow of PID procedure

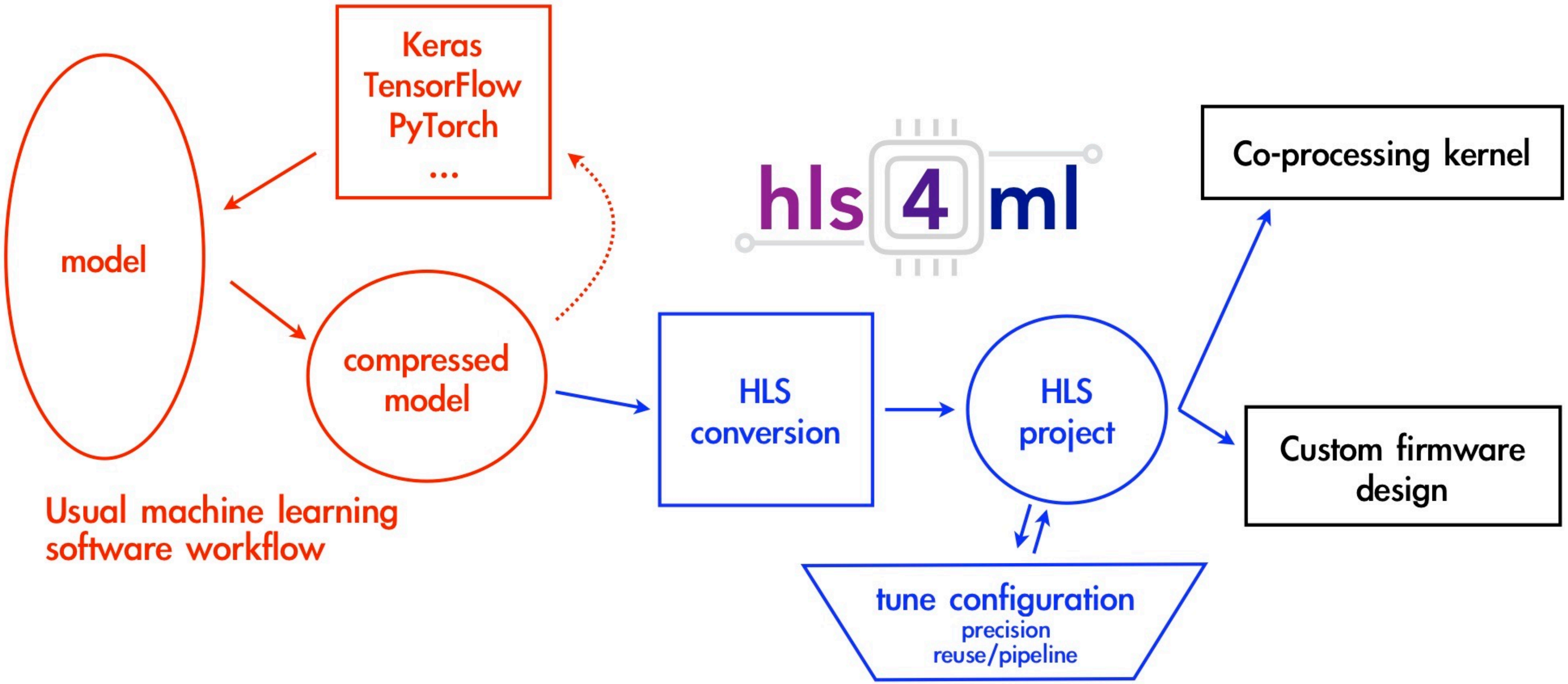

in_f7ic
data_f7ic
p
in_f7pl
data_f7pl
aoq
z
data_f3pl
in_f3pl
nchunk
in_f5ppac
data_f5ppac
data_f3ppac
in_f3ppac
data_f7ppac
in_f7ppac
load_ic_1 II(avg,max,min):N/A,N/A,N/A Latency(avg,max,min):N/A,N/A,N/A
IC
Block_entry177189_proc_U0 II(avg,max,min):N/A,N/A,N/A Latency(avg,max,min):N/A,N/A,N/A
load_pl_1_U0 II(avg,max,min):N/A,N/A,N/A Latency(avg,max,min):N/A,N/A,N/A
entry_proc_U0 II(avg,max,min):N/A,N/A,N/A Latency(avg,max,min):N/A,N/A,N/A
load_pl_618_1_U0 II(avg,max,min):N/A,N/A,N/A Latency(avg,max,min):N/A,N/A,N/A
load_ppac_617_1_U0 II(avg,max,min):N/A,N/A,N/A Latency(avg,max,min):N/A,N/A,N/A
load_ppac_616_1_U0 II(avg,max,min):N/A,N/A,N/A Latency(avg,max,min):N/A,N/A,N/A
load_p II(avg,ma Latency(N/A,N/A
PPAC
loop_ic_1_U0 II(avg,max,min):N/A,N/A,N/A Latency(avg,max,min):N/A,N/A,N/A
loop_pl_1_U0 II(avg,max,min):N/A,N/A,N/A Latency(avg,max,min):N/A,N/A,N/A
loop_pl_619_1_U0 II(avg,max,min):N/A,N/A,N/A Latency(avg,max,min):N/A,N/A,N/A
loop_ppac_xf_1_U0 II(avg,max,min):N/A,N/A,N/A Latency(avg,max,min):N/A,N/A,N/A
IC data analysis
ΔE
hit position in PPAC
Block_entry177191_proc_U0 II(avg,max,min):N/A,N/A,N/A Latency(avg,max,min):N/A,N/A,N/A
loop_ppac_op_1_U0 II(avg,max,min):N/A,N/A,N/A Latency(avg,max,min):N/A,N/A,N/A
position/direction of charged particle
loop_pid II(avg,max Latency(av N/A N/A,N/A,N/A
PID
write_output_1_U0 II(avg,max,min):N/A,N/A,N/A Latency(avg,max,min):N/A,N/A,N/A
write_output_620_1_U0 II(avg,max,min):N/A,N/A,N/A Latency(avg,max,min):N/A,N/A,N/A
Z
out_z
out_aoq
A/Q

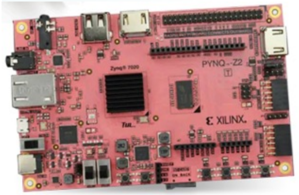
- Target = 300 MHz -> 220MHz
- Latency ~ 1000 clock = 4.3usec
- Pipeline process for every 5 clock (-> Maximum data input 40MHz)
- **Throughput 570k/13msec (cf. 570k/250msec by single CPU)**

X20 acceleration by FPGA

Further developments : hls4ml

hls4ml: A package for machine learning inference in FPGA





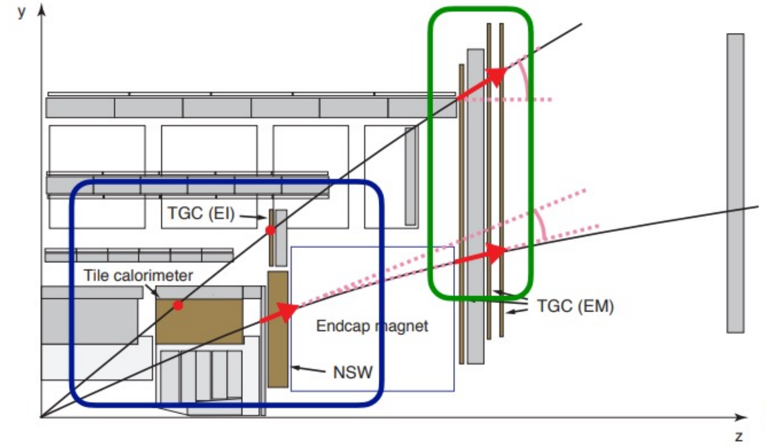
VivadoAccelerator for Xilinx accelerators / SoCs

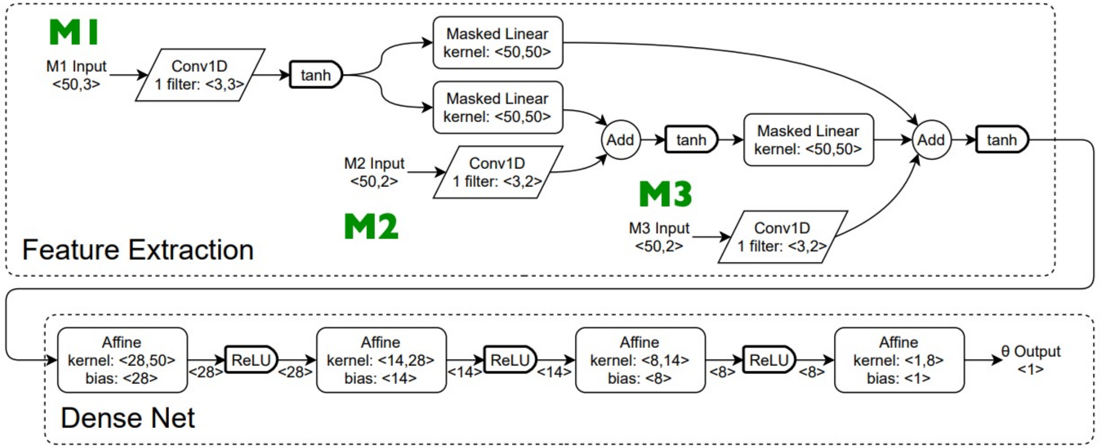
Further developments ： hls4ml

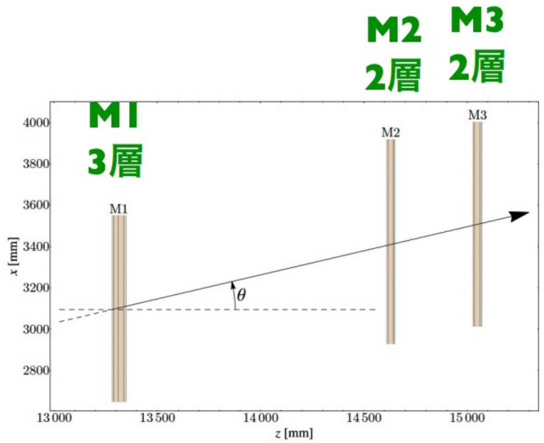
Examples of developments in Japan (ATLAS and Belle2)

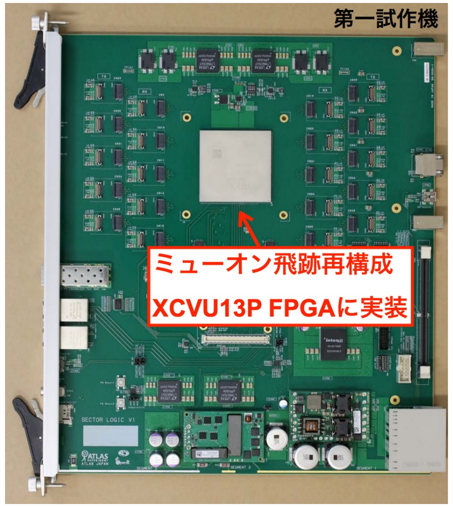
US also has a lot of activities with hls4ml for sPHENIX, J-lab, LHC experiments, EIC ….

Horii et al. (Nagoya)




M1
M1 Input <50,3>
Conv1D 1 filter: <3,3>
tanh
Masked Linear kernel: <50,50>
Masked Linear kernel: <50,50>
Add
tanh
Masked Linear kernel: <50,50>
Add
tanh
M2 Input <50,2>
Conv1D 1 filter: <3,2>
M3
M3 Input <50,2>
Conv1D 1 filter: <3,2>
M2
Feature Extraction
Affine kernel: <28,50> bias: <28>
<28>
ReLU
<28>
Affine kernel: <14,28> bias: <14>
<14>
ReLU
<14>
Affine kernel: <8,14> bias: <8>
<8>
ReLU
<8>
Affine kernel: <1,8> bias: <1>
θ Output <1>
Dense Net

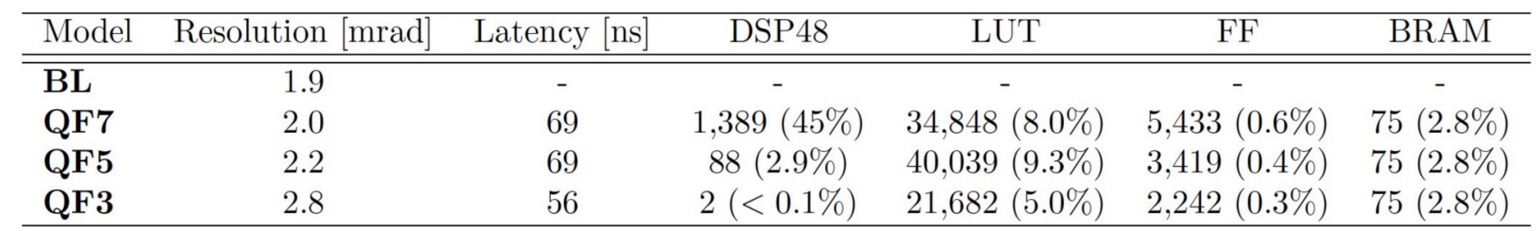




ATLAS Muon tracking using hls4ml

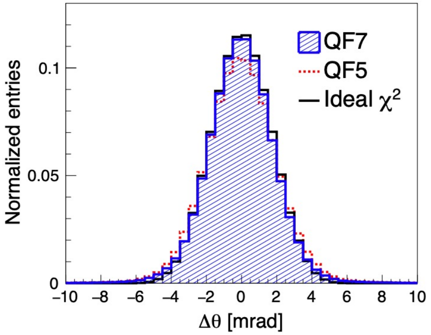
160 MHz clock, Latency = 100nsec

nima.2022.167546



Model	Resolution [mrad]	Latency [ns]	DSP48	LUT	FF	BRAM
BL	1.9	-	-	-	-	-
QF7	2.0	69	1,389 (45%)	34,848 (8.0%)	5,433 (0.6%)	75 (2.8%)
QF5	2.2	69	88 (2.9%)	40,039 (9.3%)	3,419 (0.4%)	75 (2.8%)
QF3	2.8	56	2 (< 0.1%)	21,682 (5.0%)	2,242 (0.3%)	75 (2.8%)

リソース使用率は、Super Logic Regionあたりの値

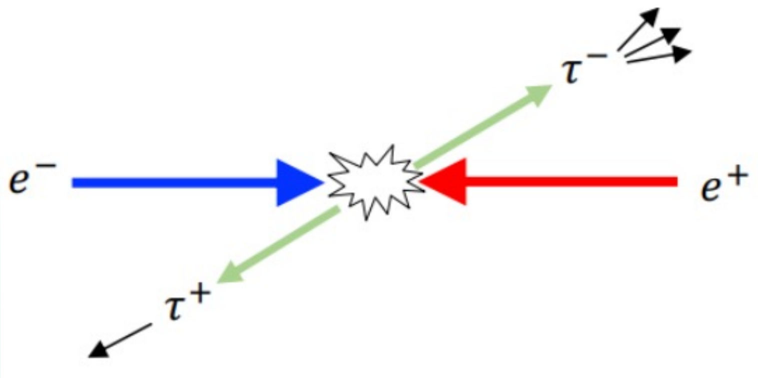


Further developments ： hls4ml

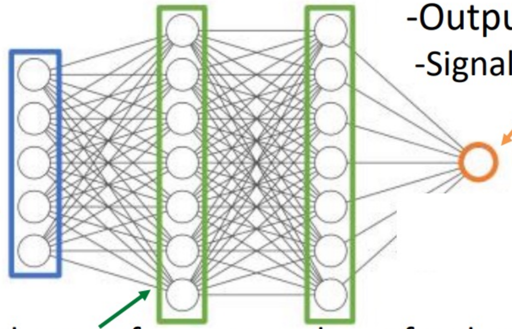
Examples of developments in Japan (ATLAS and Belle2)

US also has a lot of activities with hls4ml for sPHENIX, J-lab, LHC experiments, EIC ….

Koga et al. (KEK)

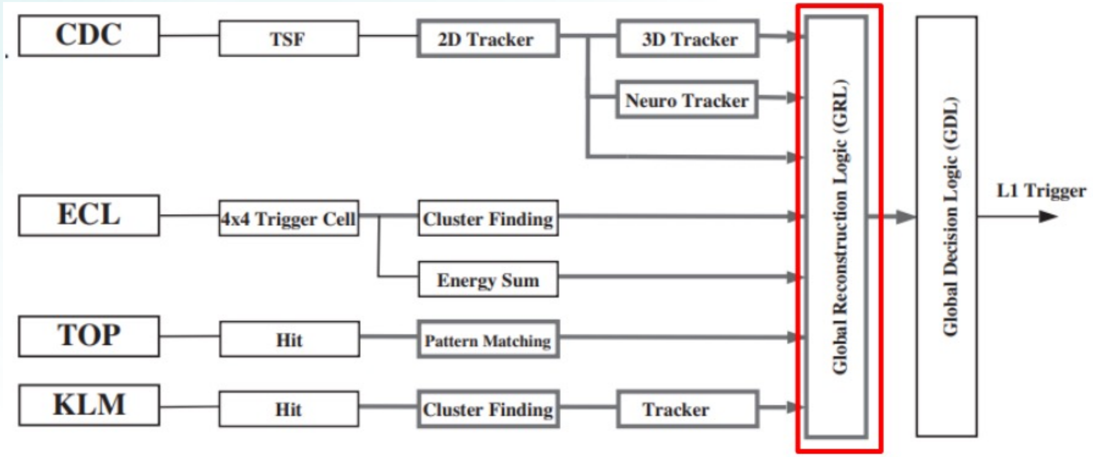


-Input a few tens nodes:
- -CDC charged track (ϕ,p_t)
- -ECL calorimeter cluster (θ,ϕ,E)
- -KLM muon track (θ,ϕ)

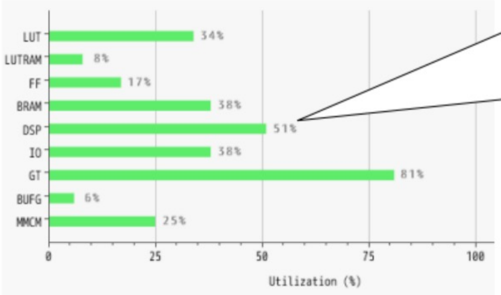


-Output 1 node:
-Signal probability

hidden layer: a few tens nodes, a few layers



-Successfully implemented to Xilinx Virtex Ultrascale (XCVU080)
- -include not only NN but also original logics on GRL
- -latency: 14clock × 127MHz = ~0.11μs



-DSP usage increased from 0% to 51% by implementation of NN

-Others are not so affected

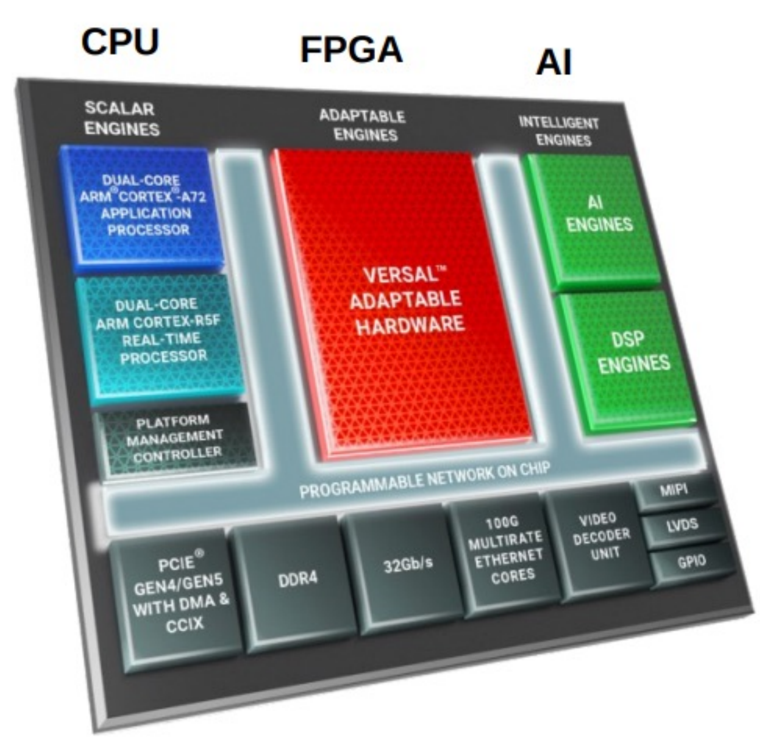
-Will be used for the next physics operation in 2024

Further developments : Versal

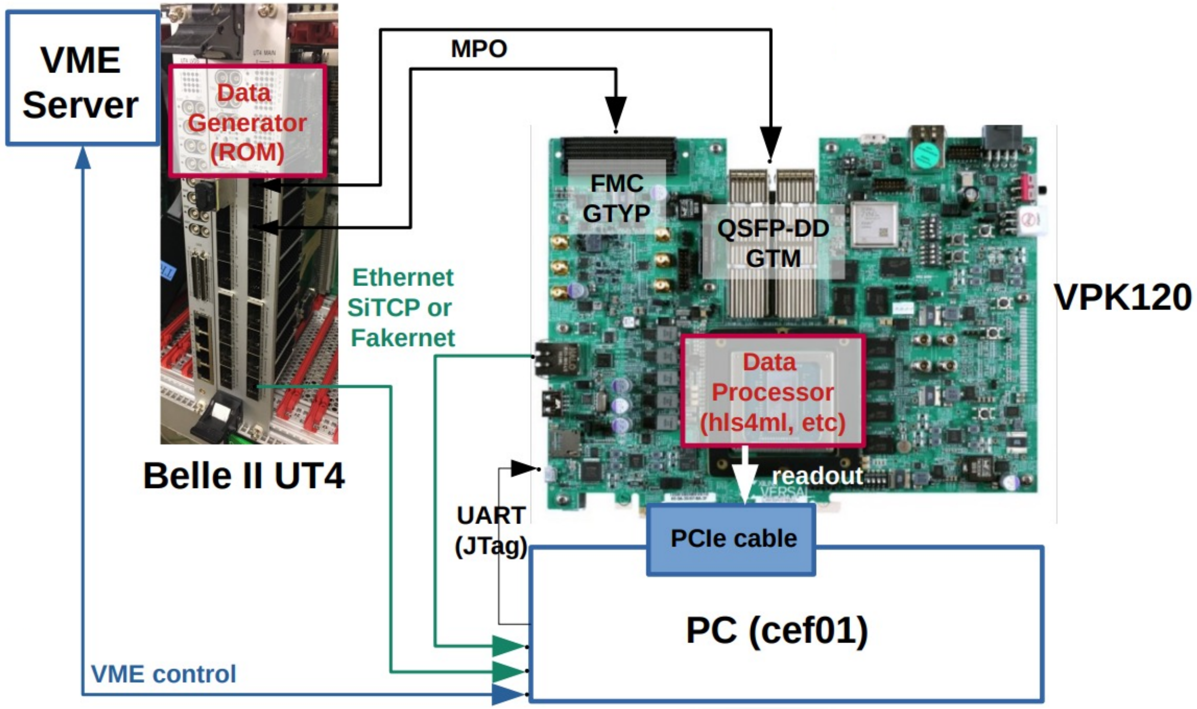
Versal Projects in Japan (Collider electronics Forum in Japan)

The features of Versal series:

- ACAP SoC.
- AI/DSP engine: interface to implement ML core into firmware.
- High Bandwidth Memory (HBM).
- Larger number of cells + High transmission bandwidth.


CPU
FPGA
AI
SCALAR ENGINES
ADAPTABLE ENGINES
INTELLIGENT ENGINES
DUAL-CORE ARM® CORTEX®-A72 APPLICATION PROCESSOR
DUAL-CORE ARM CORTEX-R5F REAL-TIME PROCESSOR
PLATFORM MANAGEMENT CONTROLLER
VERSAL™ ADAPTABLE HARDWARE
AI ENGINES
DSP ENGINES
PROGRAMMABLE NETWORK ON CHIP
PCIE® GEN4/GEN5 WITH DMA & CCIX
DDR4
32Gb/s
100G MULTIRATE ETHERNET CORES
VIDEO DECODER UNIT
MIPI
LVDS
GPIO

Test bench in KEK (lead by HEP community) and extension plans



Further developments : FELIX Card

Front-End Link eXchange (FELIX)

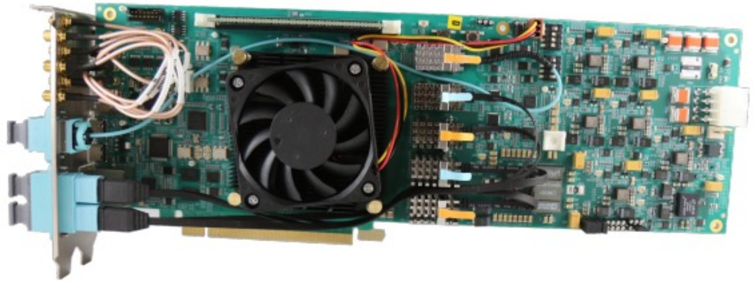
https://atlas-project-felix.web.cern.ch/atlas-project-felix/

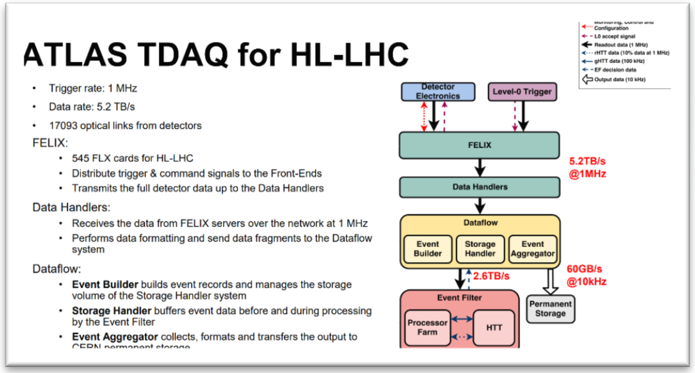
- Custom FPGA based PCIe cards
- Readout, trigger, clock distribution, Slow Control, BUSY
- Router between FE serial links and commercial network

FELIX

The FLX-182 card ATLAS Phase-2 Upgrade (Run4)

- FPGA: AMD Versal Prime VM1802
- 4 Firefly transceivers to support 24 bidirectional optical links
  - Up to 25 Gb/s per link
- 1 Firefly for LTI/TTC interface
  - New protocol for Timing, Trigger and Control
  - 100Gb/s Ethernet or White Rabbit are optional
- 16-lane PCIe Gen4 interface (240 Gb/s)
  - 2x 8 lanes bifurcated





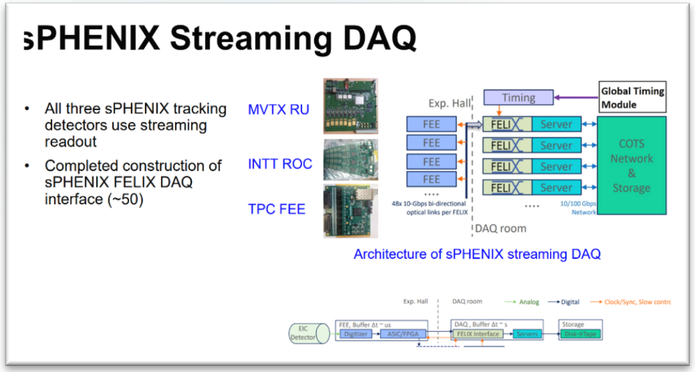
ATLAS TDAQ for HL-LHC

- Trigger rate: 1 MHz
- Data rate: 5.2 TB/s
- 17093 optical links from detectors

FELIX:
- 545 FLX cards for HL-LHC
- Distribute trigger & command signals to the Front-Ends
- Transmits the full detector data up to the Data Handlers

Data Handlers:
- Receives the data from FELIX servers over the network at 1 MHz
- Performs data formatting and send data fragments to the Dataflow system

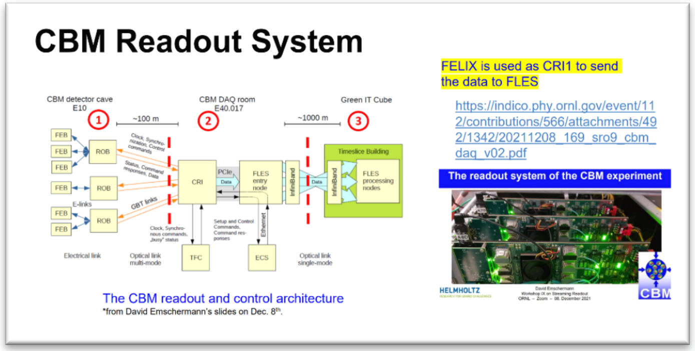
Dataflow:
- **Event Builder** builds event records and manages the storage volume of the Storage Handler system
- **Storage Handler** buffers event data before and during processing by the Event Filter
- **Event Aggregator** collects, formats and transfers the output to CERN permanent storage



sPHENIX Streaming DAQ

- All three sPHENIX tracking detectors use streaming readout
- Completed construction of sPHENIX FELIX DAQ interface (~50)

Architecture of sPHENIX streaming DAQ



CBM Readout System

The CBM readout and control architecture

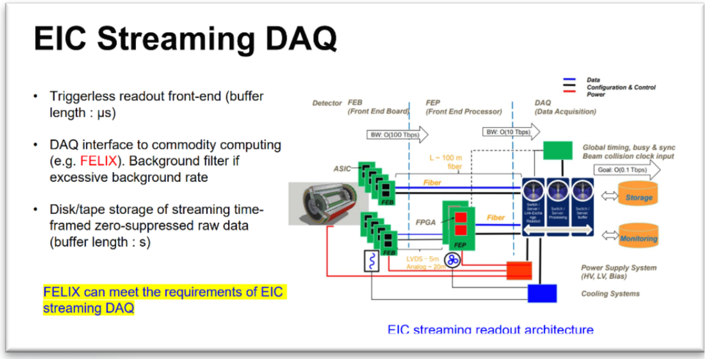
*from David Emschermann's slides on Dec. 8th.

FELIX is used as CRI1 to send the data to FLES

https://indico.phy.ornl.gov/event/112/contributions/566/attachments/492/1342/20211208_169_sro9_cbm_daq_v02.pdf

The readout system of the CBM experiment

HELMHOLTZ
David Emschermann
Workshop IX on Streaming Readout
ORNL – Zoom – 08. December 2021
CBM



EIC Streaming DAQ

- Triggerless readout front-end (buffer length : μs)
- DAQ interface to commodity computing (e.g. FELIX). Background filter if excessive background rate
- Disk/tape storage of streaming time-framed zero-suppressed raw data (buffer length : s)

FELIX can meet the requirements of EIC streaming DAQ

EIC streaming readout architecture

FELIX becoming more widely used in various experiments