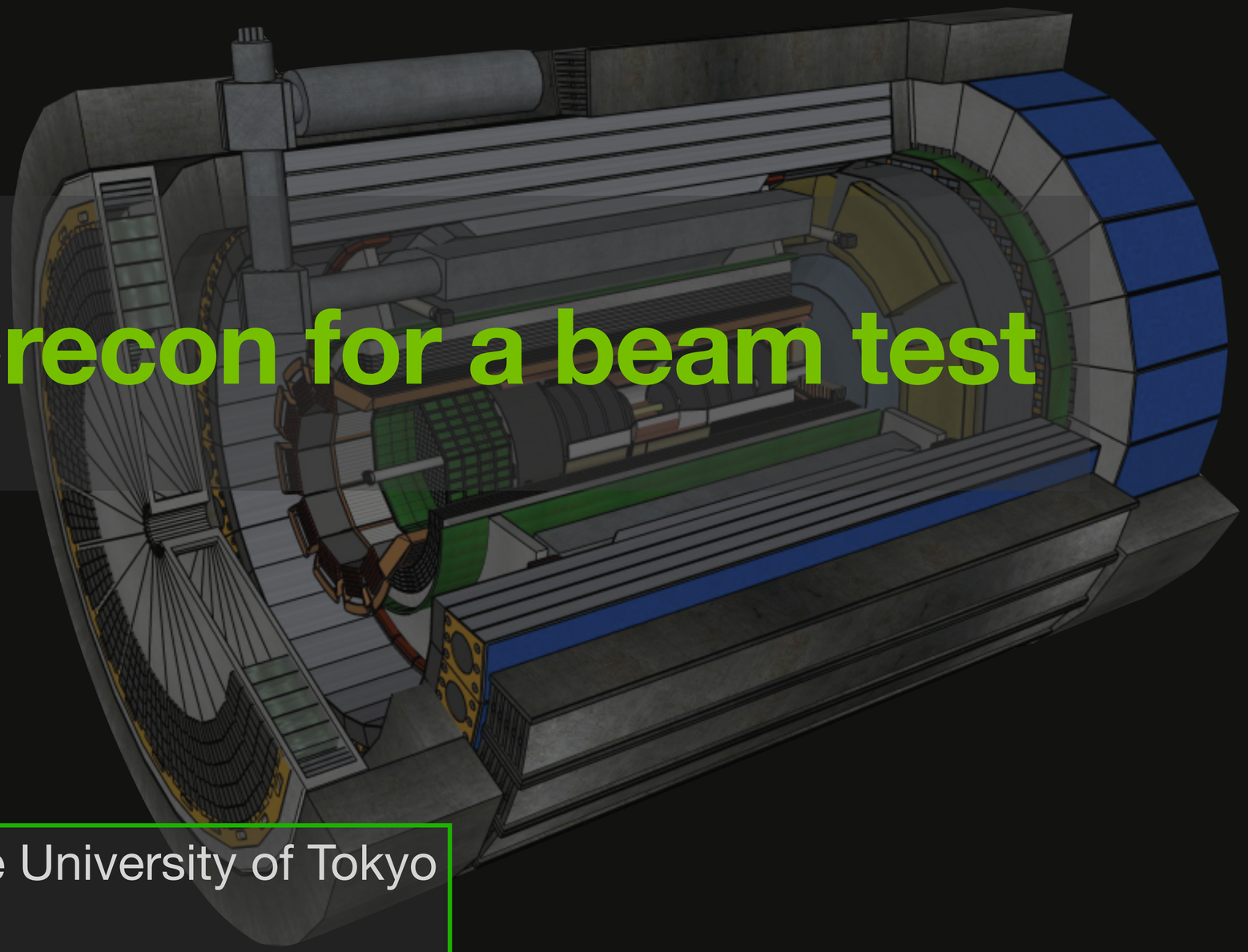
SPADI-EICrecon for a beam test

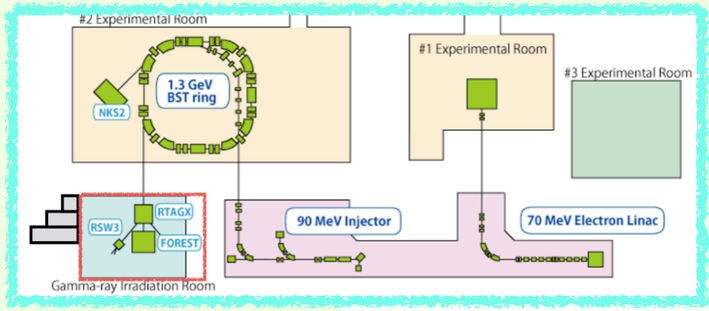


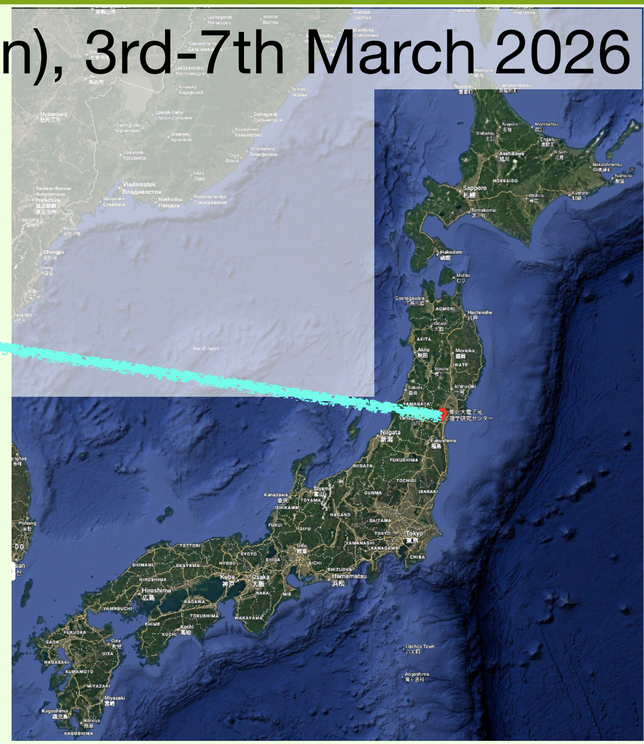
Quark Nuclear Science Institute, The University of Tokyo
Takuya Kumaoka

ePIC Streaming DAQ Test with a Local Beam Test

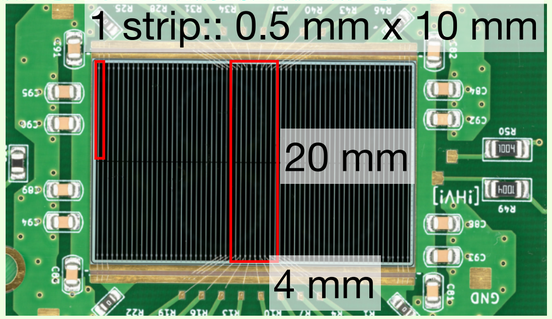
Japan barrel TOF R&D team held a beam test at RARiS (Tohoku, Japan), 3rd-7th March 2026

GeV-γ beam line:
positron: < 1 GeV/*c*, < 800 MeV/*c* (~30 Hz)
electron: < 1 GeV/*c* (~30 Hz)



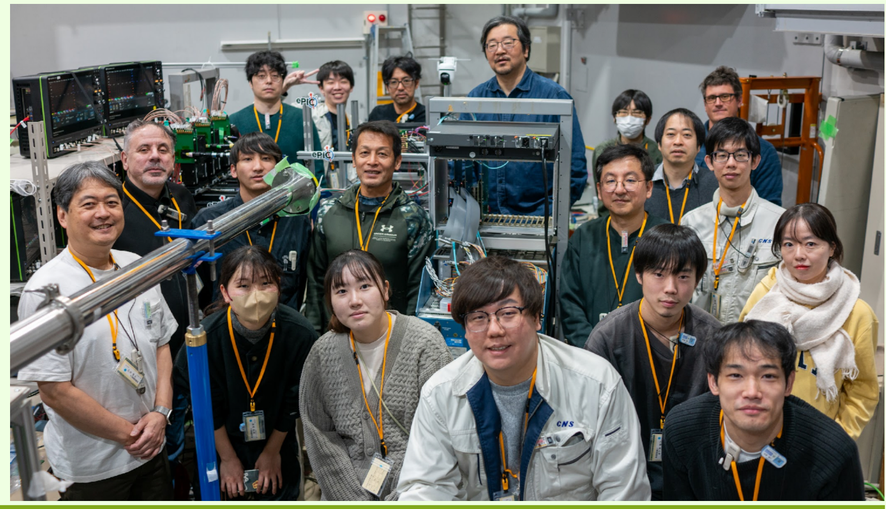


Detector:
AC-LGAD (AC-coupled Low-Gain Avalanche Diode):
- High time resolution ~30 ps
- High granularity



Beam test
- This beam test was conducted by 23 people.
- We had three parallel lines.

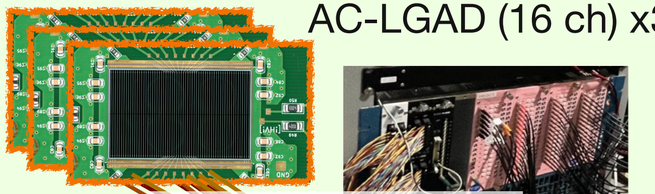
- **line1: mini ePIC streaming test with SPADI using AC-LGAD**
  → N.Kobayashi (RCNP, Osaka Univ), T.Takahashi (RCNP, , Osaka Univ), S. Y. Ryu (RCNP, Osaka Univ), T.Gunji (QNSI, Univ Tokyo), M.Kaneta (Tohoku Univ), K.Ono (Shinshu Univ)
- line2: AC-LGAD performance test using Waveform (Ochiroscope)
- line3: RFSoC DAQ test for AC-LGAD performance estimation



Developed Two Streaming DAQ Lines

Tried mini ePIC streaming orchestration (Echelon0-Echelon2)

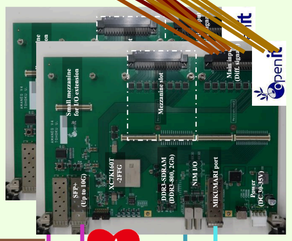
DAQ Flow



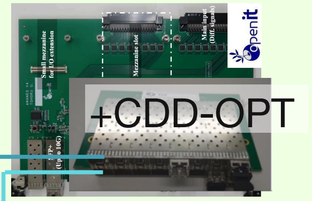
AC-LGAD (16 ch) x3

QTC / ECL to LVDS

AMANEQ
(Data read out)
x2



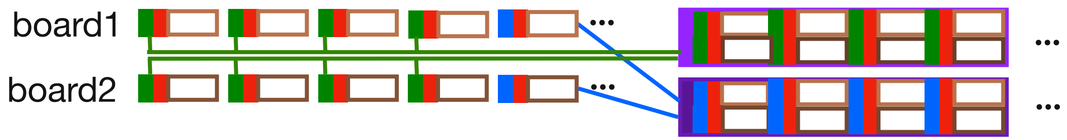
AMANEQ (Clock) x1



Series of data

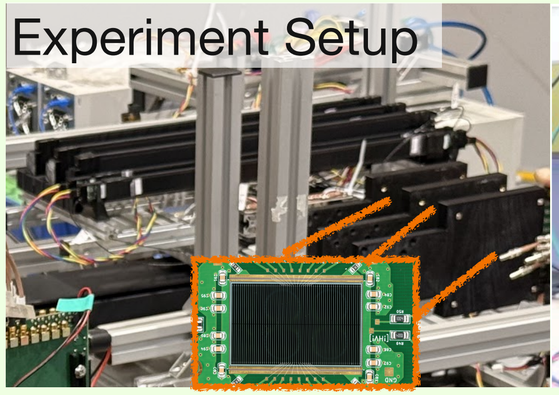
Series of data

Desktop PC (NestDAQ) 524 μs
- Sampler
- SubTimeFrameBuilder - TimeFrameBuilder

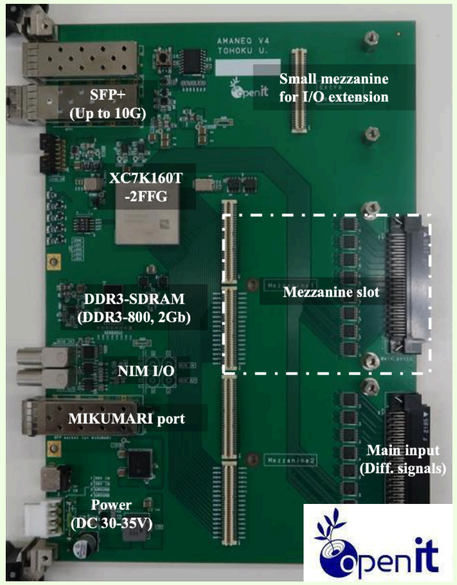


- **Output**
  1. **File sink (EDM4eic) → XRootD →Univ Osaka**
  2. **Send a data as message queue**

Experiment Setup



AMANEQ/MIKUMARI Setup



AMANEQ [ref]

(A main electronics for network oriented trigger-less data acquisition system)

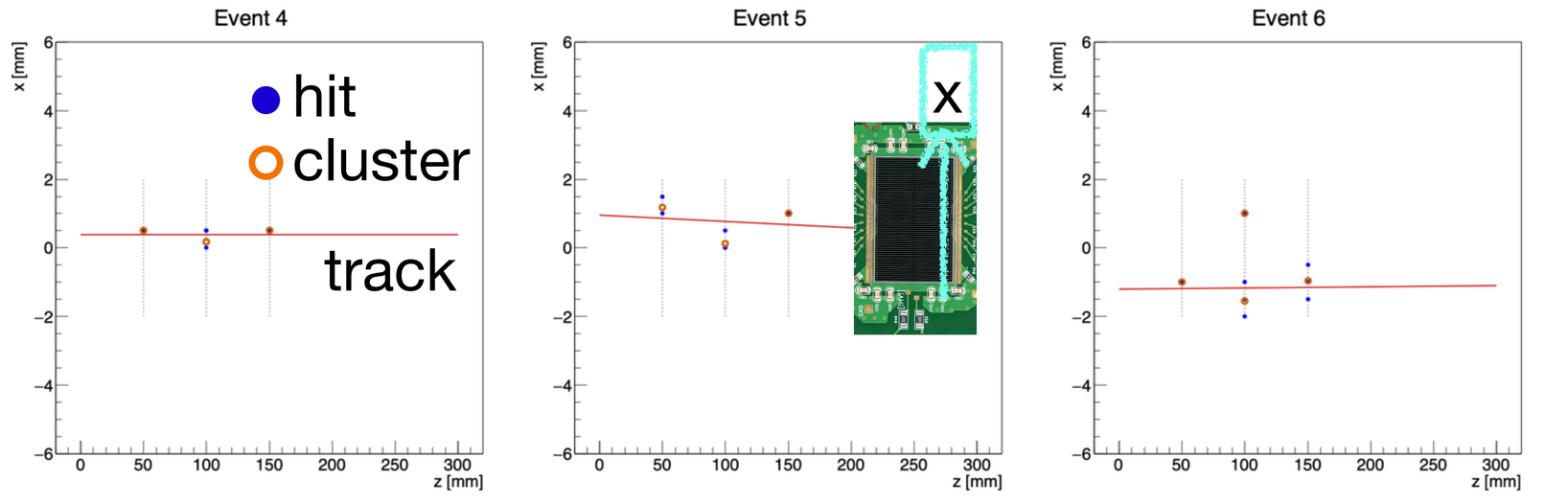
- Kintex7 with speed grade -2
  - Transceiver bandwidth up to 10 Gbps (10 GbE)
- Two mezzanine slot for functionality extension
  - HR-TDC
  - Clock distribution
  - Input-channel extension
  - etc…
- A jitter cleaner (CDCE62002)
- DDR3-SDRAM as a de-randomizer

AMANEQ→EDM4eic→EICrecon

We had not use EICrecon for actual data not a simulation file produced by npsim.
→ Establish to use EICrecon for a beam test and actual data with SPADI-A DAQ system.

- AMANEQ (TDC) → Sink a file written as edm4eic::RawTrackerHit (N.Kobayashi: RCNP Univ of Osaka) [git:EDM4eicSink.cxx]

- EDM4eic file → EICrecon (K.Ono: Shinshu Univ, T.Kumaoka: QNSI Univ of Tokyo)
  Run the EICrecon (based on BTOF Plugin):
  Original EventSourse (edm4eic::RawTrackerHit)→ LGADHitCalibration_factory (edm4eic::TrackerHit)→ LGADHitClustering_factory(edm4eic::Measurement2D)



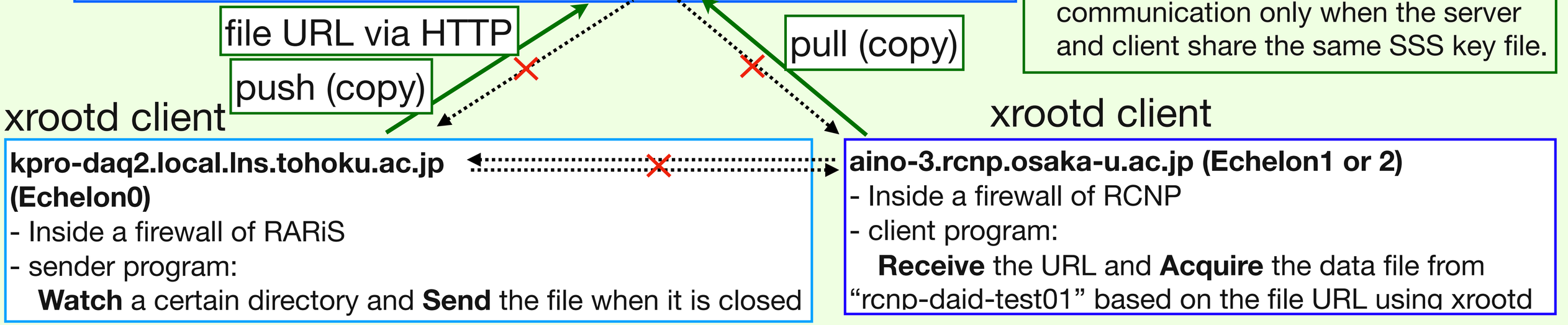
Outlook - Implement the timeframe splitter in this stream.

AMANEQ→EDM4eic→XRootD→Univ Osaka

Goal: Establish streaming data communication between servers at different locations, assuming the ePIC DAQ orchestration. (Echelon0 - Echelon1 - Echelon2).
by T.Takahashi, RCNP Univ Osaka, referred [swf-fastmon-agent] (No reusable code/New)
The data transfer is performed using XRootD

virtual machine on mdx-ii (rcnp-daid-test01)
(Echelon0 or 1)- xrootd server with standalone mode
- receiver program:
  Receive the file URL and **broadcast** the URL via SSE to client program

Security
- OpenStack security groups on mdx-ii control access by CIDR and port.
- xrootd SSS key authentication allows communication only when the server and client share the same SSS key file.



Outlook: - Communicate with ActiveMQ, Django, Rucio, and PanDA
- Split the file to be sent.

AMANEQ → ZeroMQ →EICrecon

We wanted to realize a streaming DAQ by a message queue. (N.Kobayashi, T.Kumaoka)
Nathan already develop a ZeroMQ framework based on JANA2
https://github.com/JeffersonLab/JANA2/tree/ae032455df191f0ce21a15e423572c0332514510/src/examples/misc/InteractiveStreamingExample

- It was difficult to run it directly, and it needs to be modified to be based on EICrecon.
→ The basic receiver seems to work. (T. Kumaoka)

Outlook

- Create a data sender in the INDRA message format.
- Establish an EICrecon ZeroMQ receiver.
- Enable communication between different servers.
- Develop an online monitoring tool using Jupyter (a base has already been developed by Nathan).

Summary and Outlook

Summary

- Try to establish two kinds of streaming DAQ with the SPADI system and EICrecon
  1. AMANEQ → NestDAQ → EDM4eic file sink → XRootD → EICrecon
  2. AMANEQ → NestDAQ → ZeroMQ → EICrecon
- Achievements

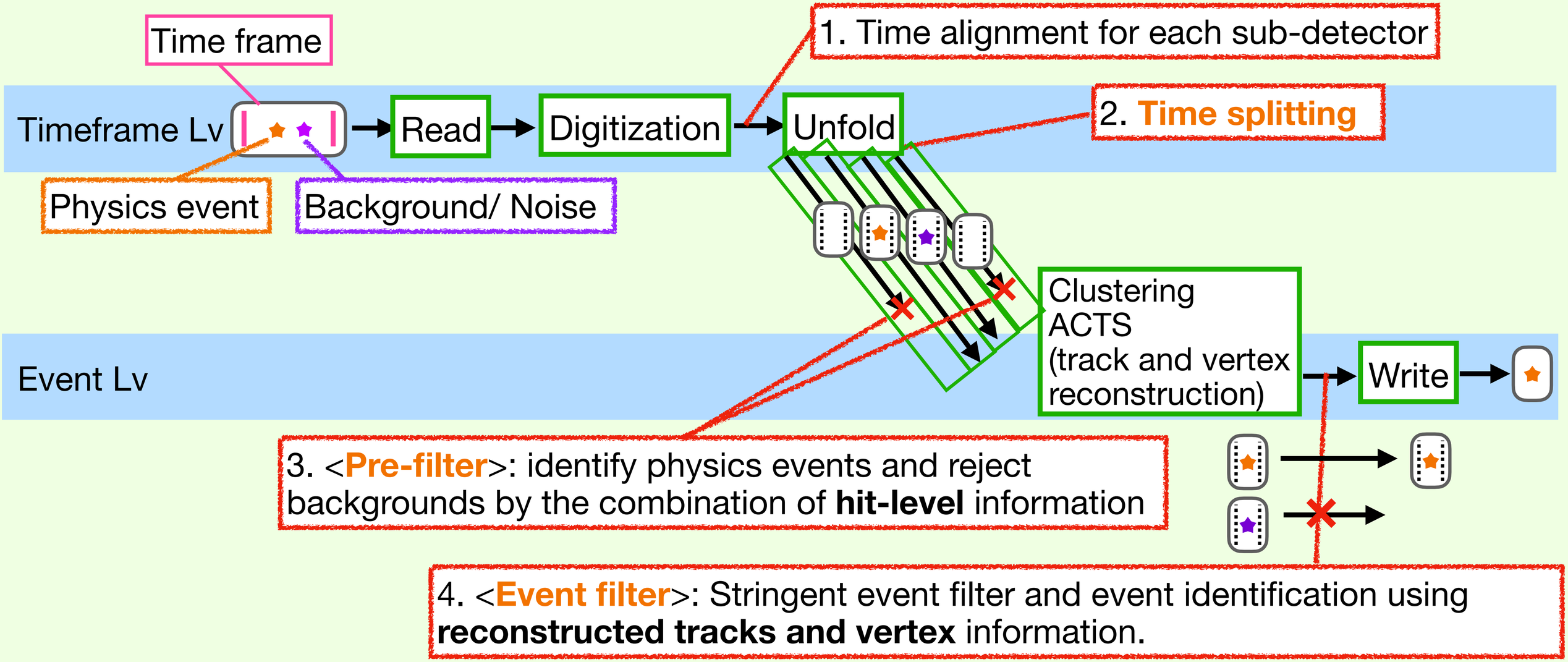
- Data sink as a file written as EDM4eic format.
- Read the file written by EICrecon.
- Send data files from the experimental hall to University of Osaka via XRootD.
- Prepare some codes for ZeroMQ receiver of EICrecon

Outlook

It is possible to reproduce the situation using a replayer.
→ Enable to test the remaining tasks

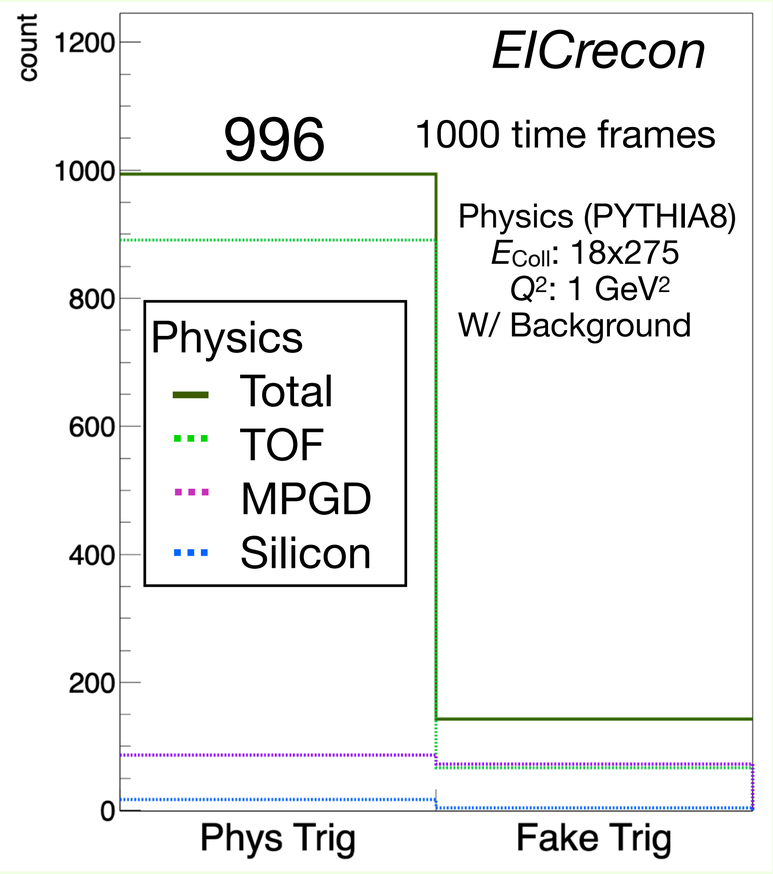
- Apply timeframe splitter and reconstruct tracks from data in a EDM4eic file sent from AMANEQ.
- Run the full pass DAQ using XRootD (NestDAQ-XRootD-EICrecon).
- Establish the full streaming DAQ using ZeroMQ.

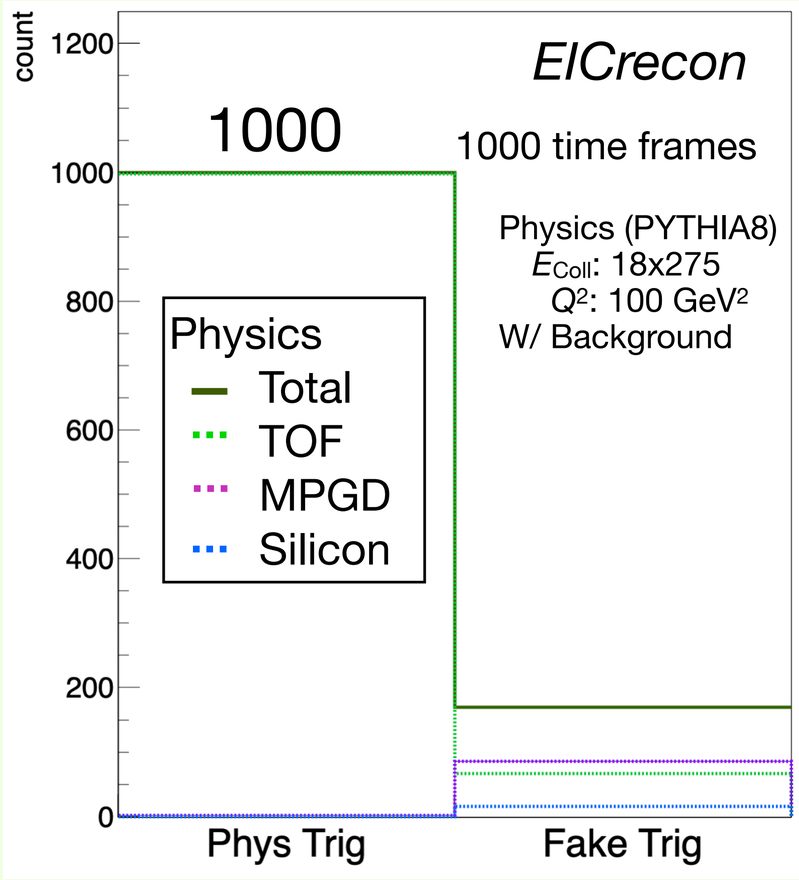
Online Event Filtering System

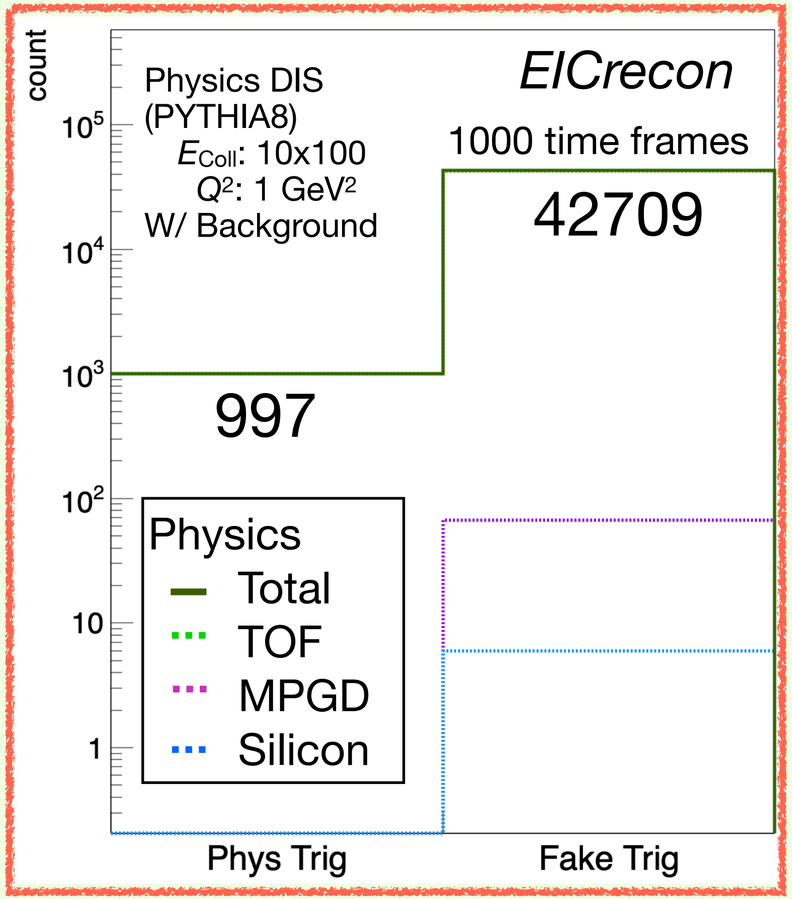


Pre-filter Efficiency

- There were problem to count the triggered physics event.
→ Fixed them, but there are still some concerns.







- Found the pre-filter can triggered most low Q^2 (> 1 GeV^2) DIS events.
- In the new bkg simulation, large number of fake triggers were created.
→ I am checking the details now.

Sub-detectors used for the pre-filter

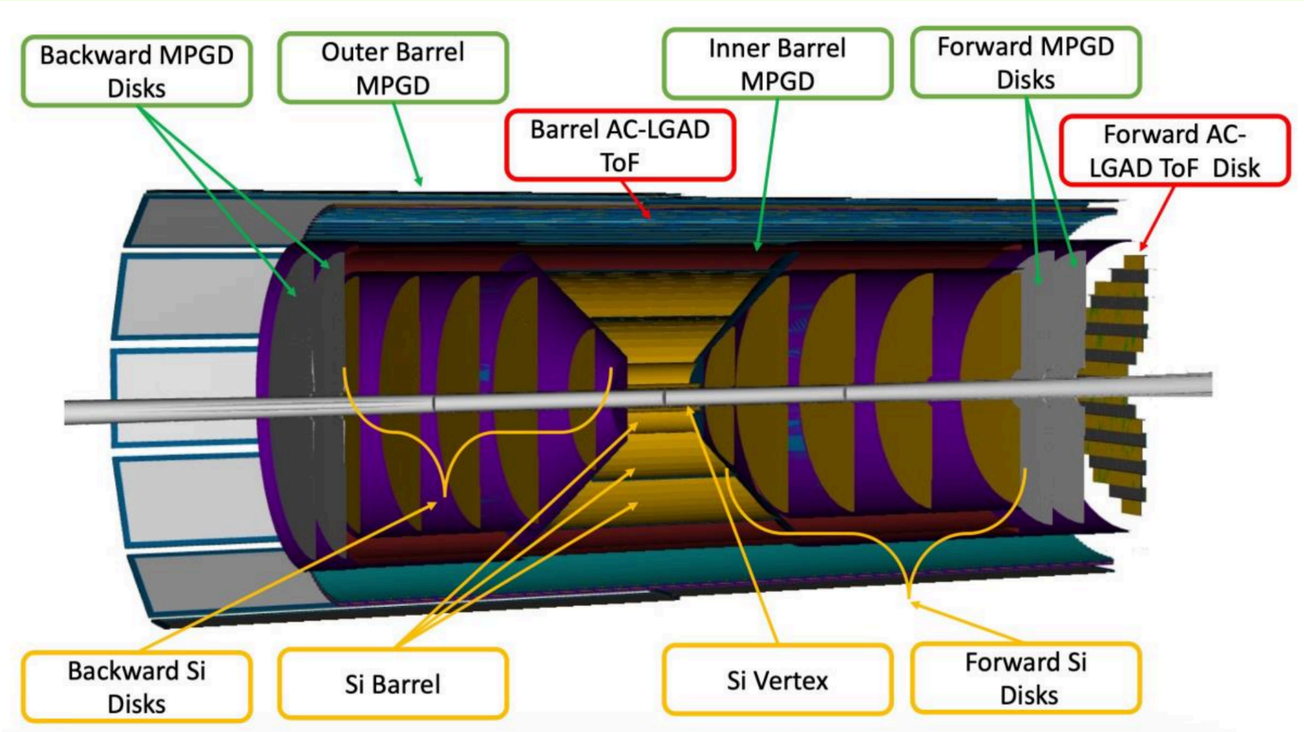
Need some sub-detectors's hit coincidence to extract a physics event from a time frame

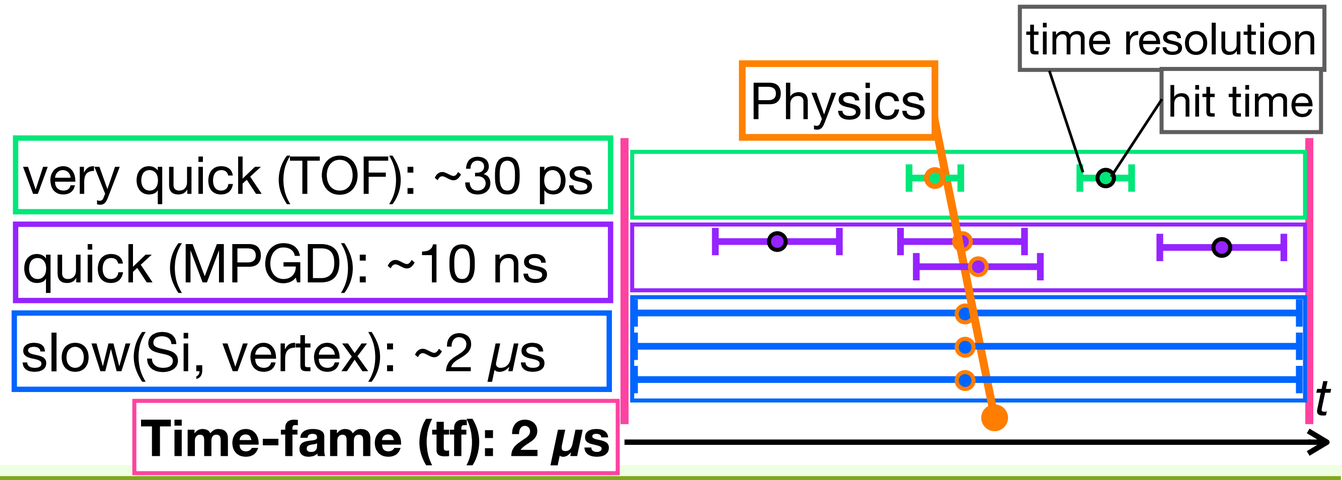
→ Use tracker detectors (**ToF**, **MPGD** and **Silicon (MAPS)**) for the event filter

ToF
- Very high time resolution (~ 30 ps)
- One layer / There are uncovered region

MPGD
- High time resolution (~ 10 ns)
- Wide covered region
- Two layers

Silicon (MAPS)
- 2 μs time window (~ time frame length)
- Wide covered region
- Multiple layers





Outlook map

5. Test online chain (E0 - E1 orchestraton::connect detector/rucio) (**- Sep?)**

1. Test for **various** physics events (**low Q^2**) (**- May)**

Time frame

Timeframe Lv

Read → Digitization → Unfold

Physics event

Background/ Noise

3. Apply **multi-threads** (or GPU) (**- Jun**)

6. Apply detector response (???)

Event Lv

Clustering
ACTS
(track and vertex reconstruction)

Write

2. Up date the pre-filter using **all sub-detectors** (**- May)**

4. Implement "**Event filter**": Stringent event filter and event identification using **reconstructed tracks and vertex** information. (**- July)**

7. Add calibration stream line (**???)**