Streaming DAQ software prototype at J-PARC hadron experimental facility (HEF)

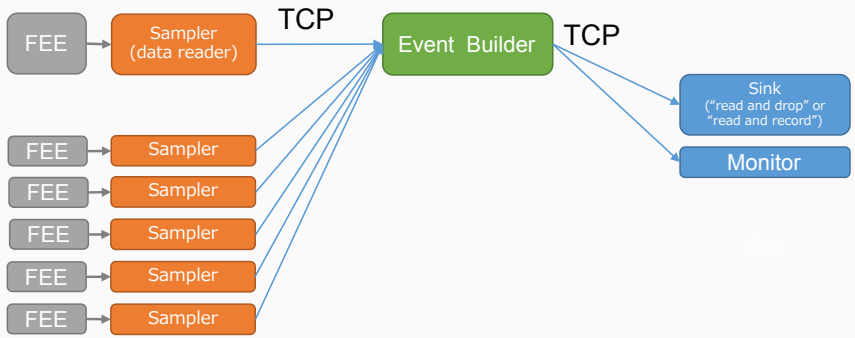
Tomonori TAKAHASHI (RIKEN)
Ryotaro Honda, Youichi Igarashi, Hiroshi Sendai

August 3rd, 2022
23rd Virtual IEEE Real Time Conference

Motivation

The DAQ system used in nuclear/hadron physics experiments at J-PARC HEF

- Triggered DAQ with network-distributed readout and event building
- Developed 10+ years ago for small- to medium-scale experiments.

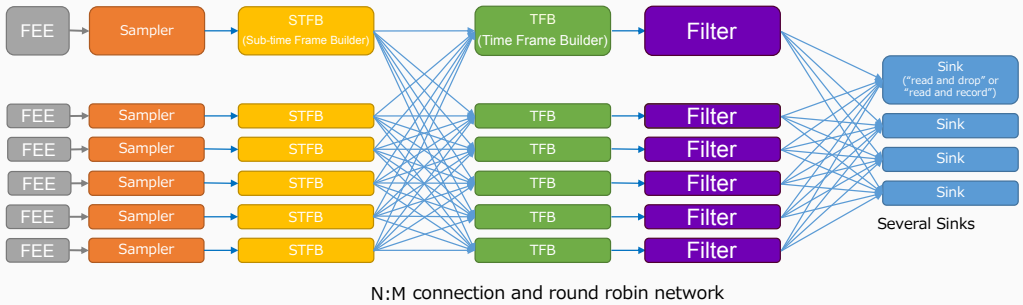


Limitations for new experiments with high event and data rates

- The single endpoint event builder is the bottleneck.
- It is difficult to a develop complex trigger system (hardware and logic) with only a few people in each group.

New DAQ software is needed.

- Load balance across multiple endpoints to cope with high event/data rates
- Flexible, used commonly in several experiments
- Simple and low learning costs



DAQ software

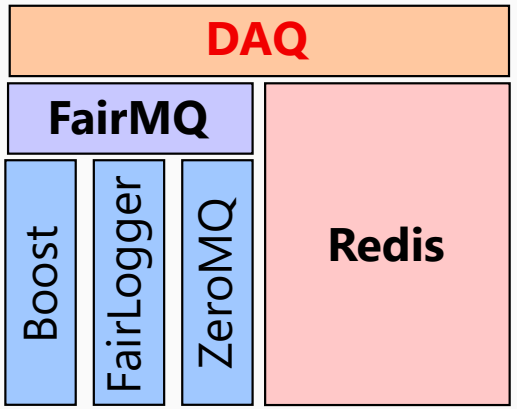
FairMQ

- **State machine** to execute a user task
- Data transport through **ZeroMQ** (, shared memory, InfiniBand)
- Few library dependencies

→ suitable for task processing with multiple endpoints
However, the official user interface of FairMQ is designed for a batch job system, not a web-friendly UI suitable for DAQ.

Redis

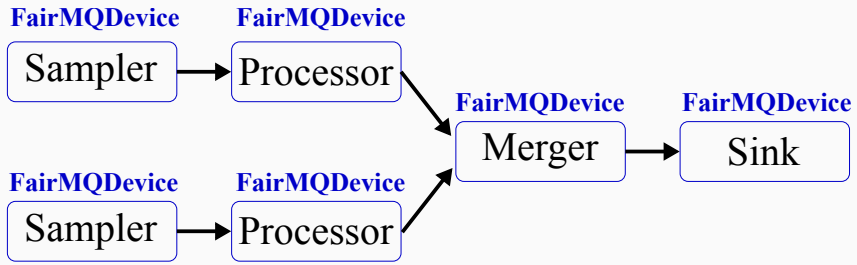
- In-memory type **Key-Value store**
  - Fast
  - Various value types: strings, lists, hashes, ...
  - Client libraries in many programming languages
- **Message queue broker**: controlling and monitoring state machine
- **Time series database**: monitoring task/data flow

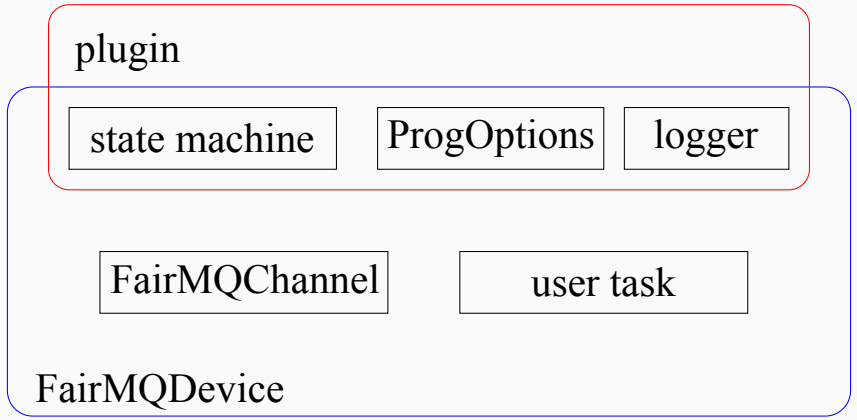


We have developed a DAQ software prototype on top of FairMQ and Redis.

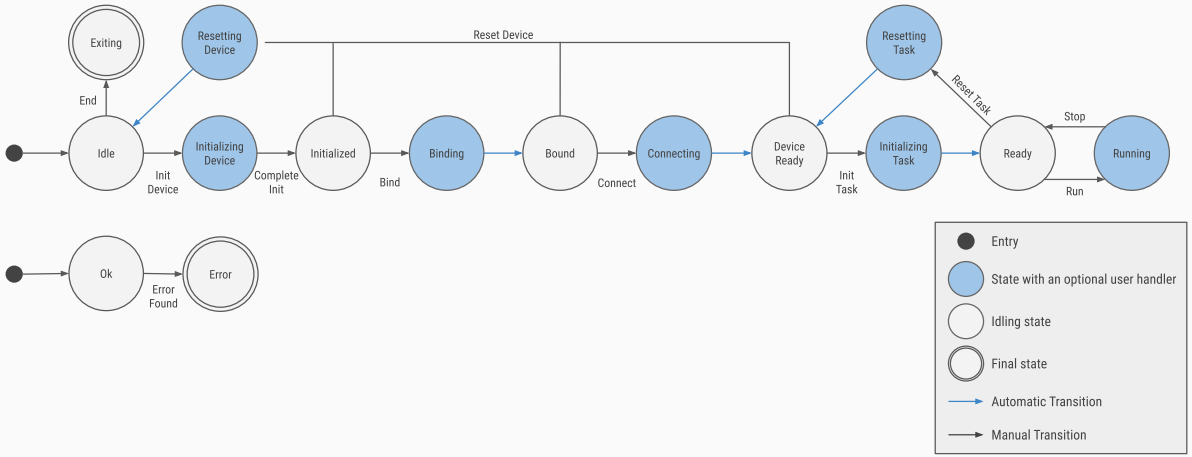
FairMQ

- **FairMQDevice**
  - A process to execute user task
- **FairMQChannel (FairMQSocket)**
  - Data transportation between FairMQDevices
- **FairLogger**
  - Log output levels, and destinations (consol, file, ...)
- **Plugin**
  - Dynamically loadable library at process launch to **add a custom user interface**
  - Controls the state machine
  - Configures parameters (program options)





State machine



https://github.com/FairRootGroup/FairMQ

FairMQ custom plugins using Redis client library

DAQ service

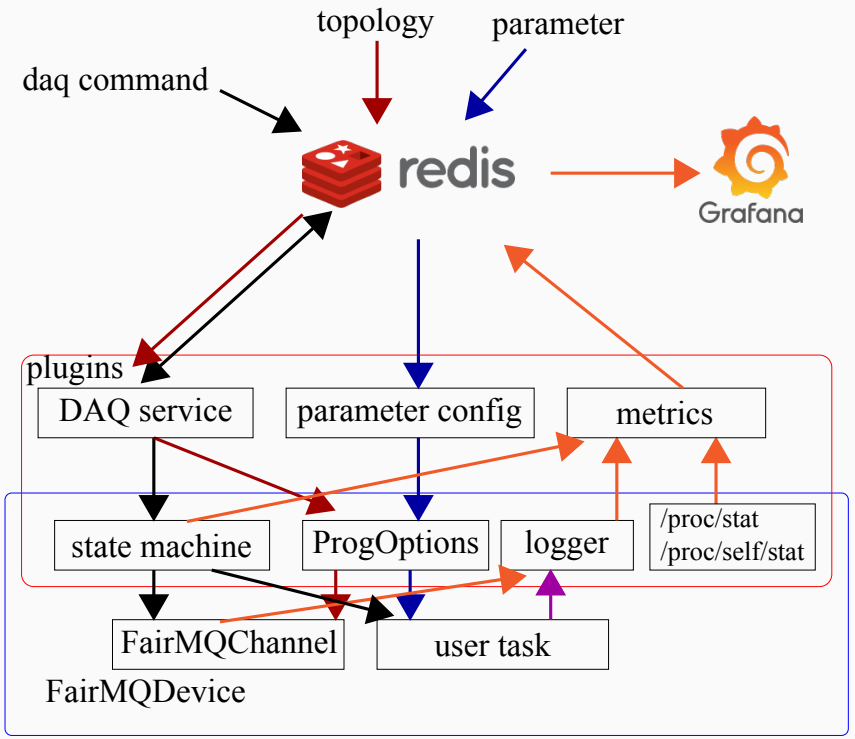
- Control and monitor state machines (Redis message queue)
- **Service discovery** to easily configure topology (Redis key-valu store) (→ next slide)

Parameter config

- Alternative to command line input parameters (Redis key-value store)

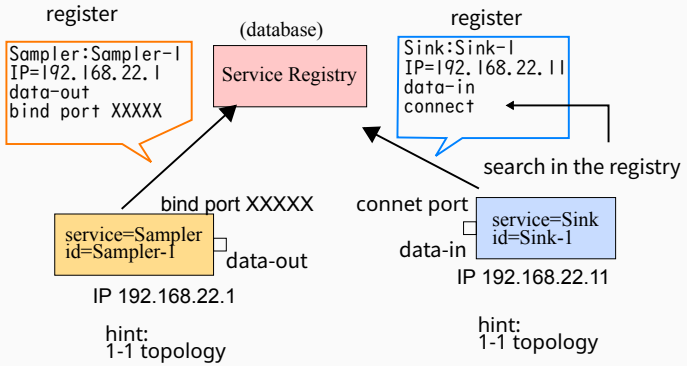
Metrics

- Monitoring states and data flow metrics with Grafana dashboard (Redis time series database)



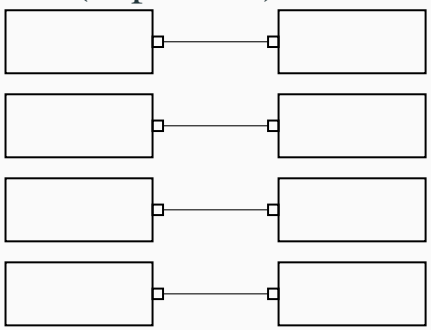
Service discovery

- We want to easily set up the topology configuration for multiple endpoints even if the number of endpoints grows.
- The following features are introduced:
  - **Redis as a service registry**
    - Register **process presence** with TTL (Time-To-Live), information on connection (**address, port**
    - The expiration of TTL is used to detect the abnormal process termination.
  - Simple rules to configure topology.
    - ***service*** = a process group
    - configure connection between ***services***
      → generate configuration parameters of connections between processes by using **topology hint** (1-1, 1-N, N-1, N-M)

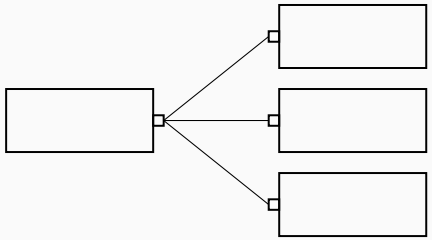


Typical topology in DAQ

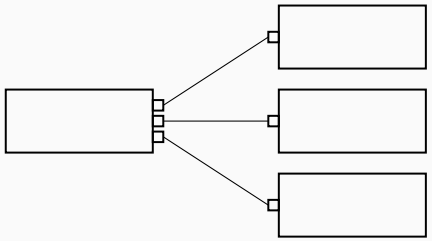
1-1 (N parallel)



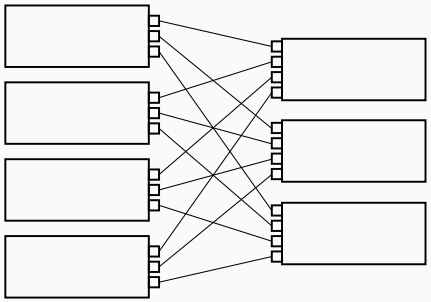
1-N
(broadcast, PUB-SUB)



1-N (static round robin)

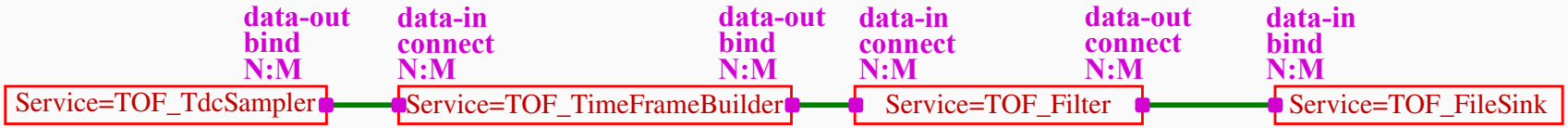


N-M
(static round robin)



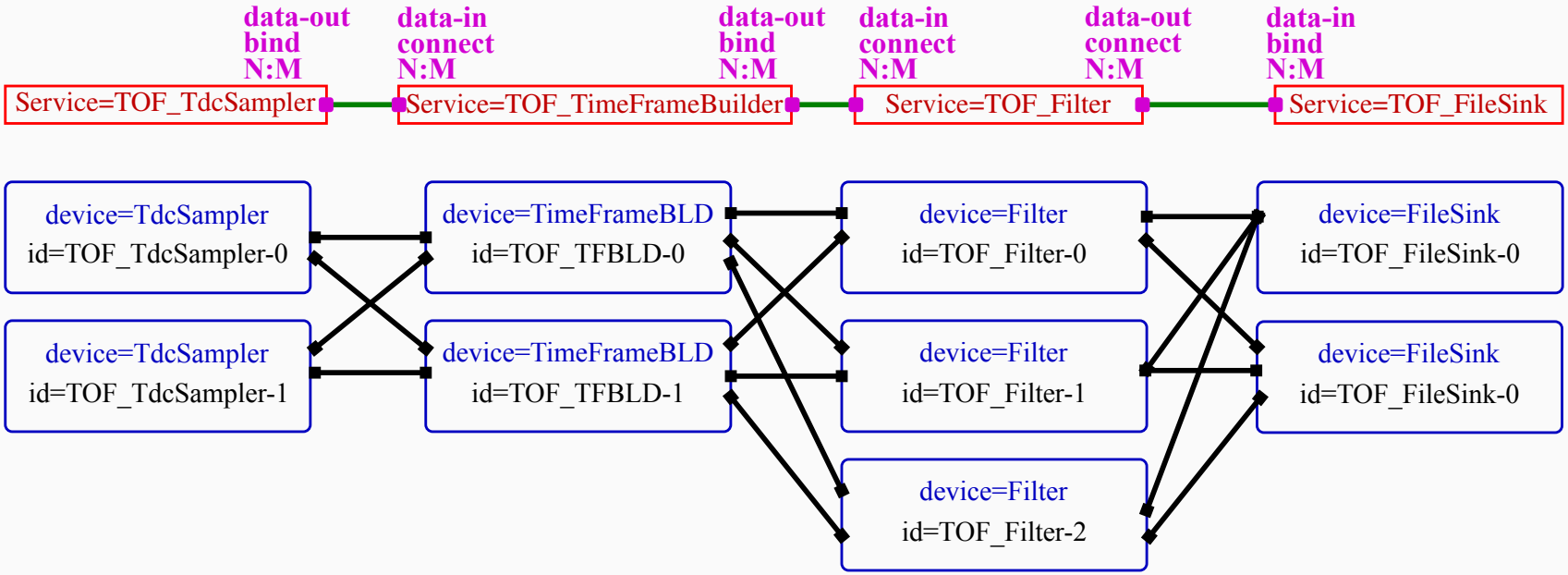
Example of topology configuration

- 4 ***services***: `TOF_TdcSampler`, `TOF_TimeFrameBuilder`, `TOF_Filter`, `TOF_FileSink`
- Input parameters uploaded to Redis (= 9 lines of shell command of `redis-cli`)
  - 6 ***endpoints*** between ***services***
  - 3 ***links*** between ***services***



Example of topology configuration

- 4 ***services***: `TOF_TdcSampler`, `TOF_TimeFrameBuilder`, `TOF_Filter`, `TOF_FileSink`
- Input parameters uploaded to Redis (= 9 lines of shell command of `redis-cli`)
  - 6 ***endpoints*** between ***services***
  - 3 ***links*** between ***services***

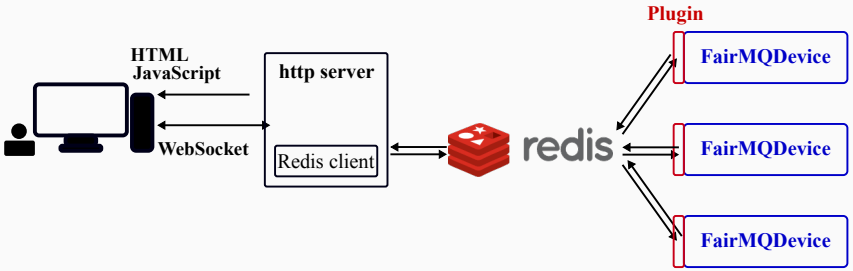


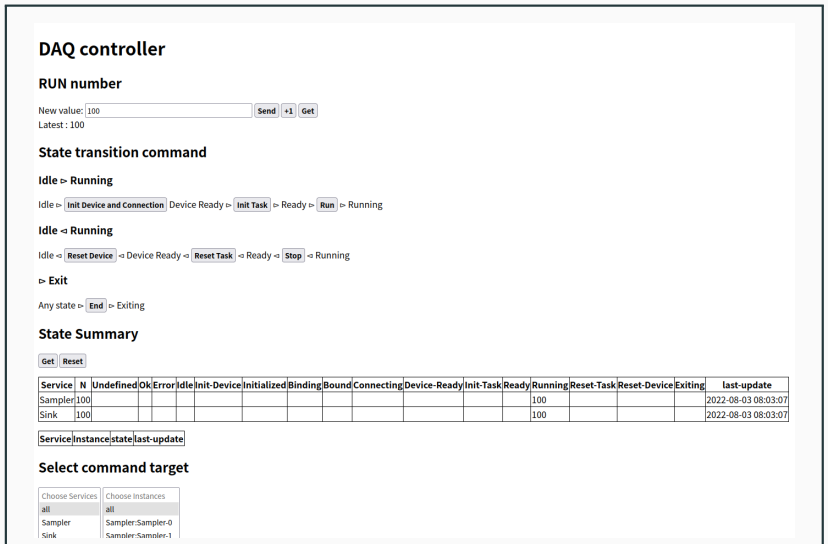
The topology parameters can remain the same even if the number of processes changes.

User interface

Controller UI

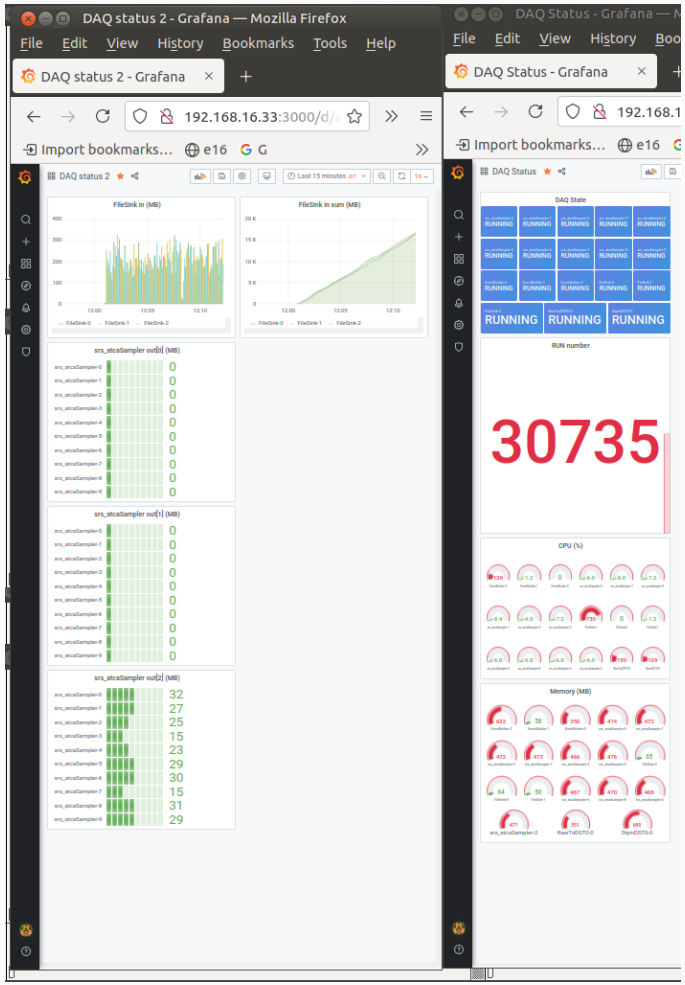
- Redis message queue (pub-sub)
  - Simple JSON message
    - Distribute commands to change state
    - Collect responses
  - Receive notifications of key expiration
- Direct `redis-cli` from terminal
- Web (HTML + Javascript + WebSocket)



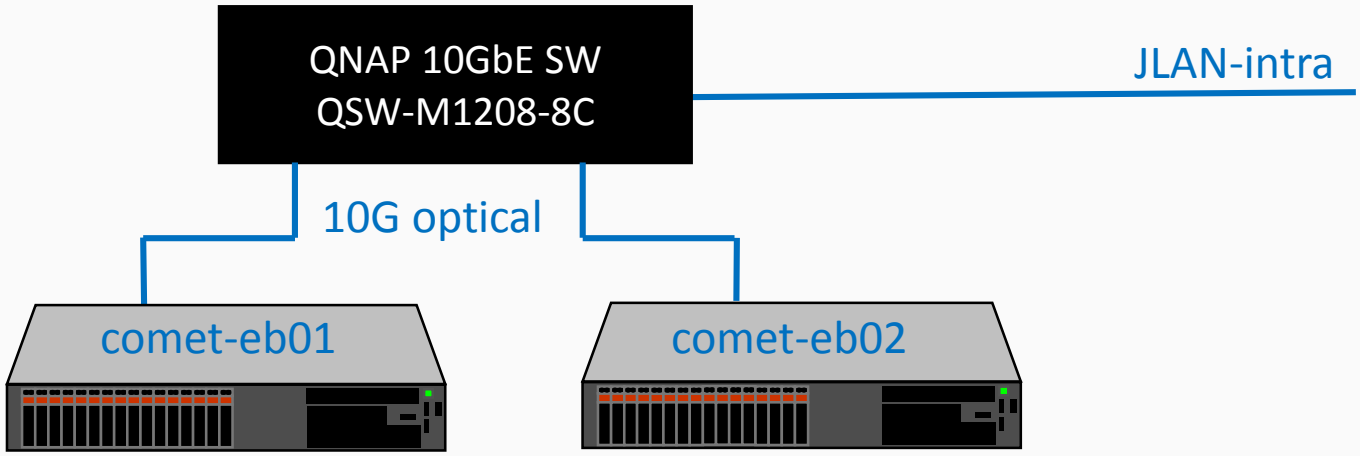


Metrics monitoring

- Redis TimeSeries + Grafana
- Process state, CPU load, RAM usage, message rate



Performance evaluation (1) : Setup

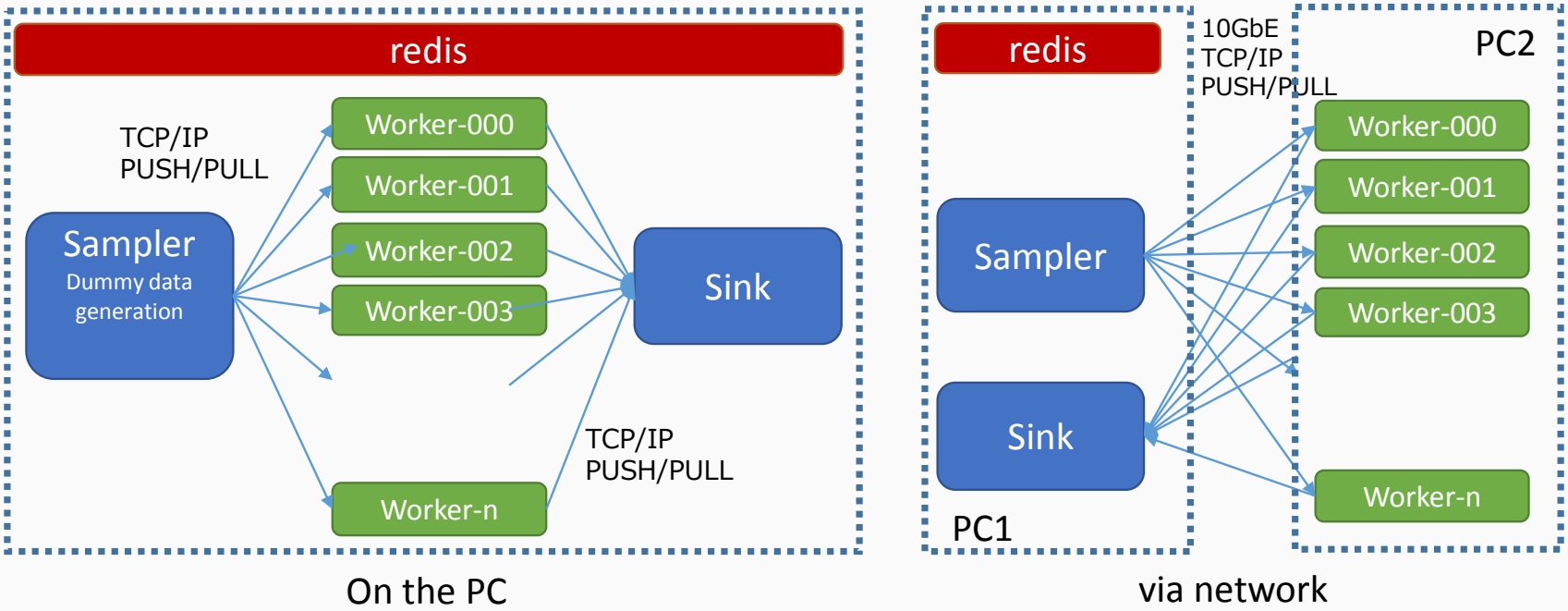


- PC: Dell PowerEdge R740
  - Xeon Gold 6126 @ 2.6 GHz, 12 Cores ×2
  - RAM: 64GB RAM
  - NIC: Intel X710-DA4
- OS: Scientific Linux 7.9
- Switch: QNAP 10GbE QSW-M1208-8C

Performance evaluation (2) : Test environments

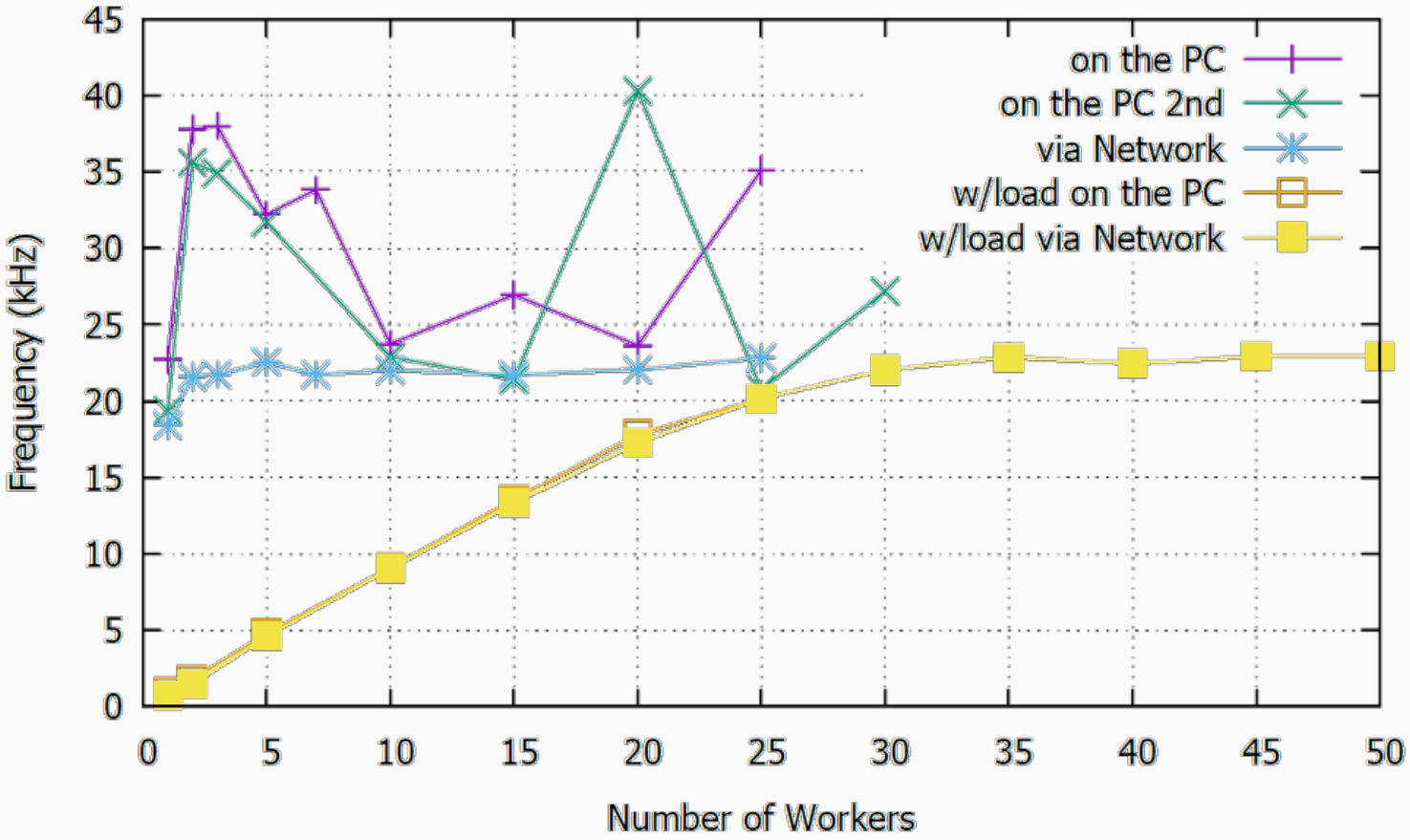
We measured the performance of:

- 2 different process deployments of ”1-n-1” (1-sampler ↔ N-workers ↔ 1-sink)
  - ”On the PC”: All processes on one PC
  - ”via network”: All worker processes on PC2, and the others on PC1
- 2 types of workloads
  - ”w/o load”: Workers do nothing with the received data, act like FIFO.
  - ”w/ load”: Workers run dummy workloads, shuffling the received payload for 1 msec.



Performance evaluation (3) : Results for 1-n-1 data transfer

- In the measurements, event size was fixed to 50 KiB.



- The system was able to handle the input data rate up to the upper limit of the 10 GbE.
- Also confirmed: the controller software could manage at least 700 processes.

Summary

- A prototype of streaming DAQ software has been developed for high event and/or data rate experiments at J-PARC HEF.
  - Easy to handle multiple endpoints
  - Simple and low learning costs
- **FairMQ** and **Redis** were employed as the technical basis.
  - Service discovery, controller
    - Currently, static round robin load balancing is supported.
  - Metrics monitoring
  - Parameter configuration
- The performance of the software prototype was evaluated.
  - It has scalability up to the network bandwidth of at least 10 GbE
  - Also, the Redis-based controller can manage at least 700 processes.

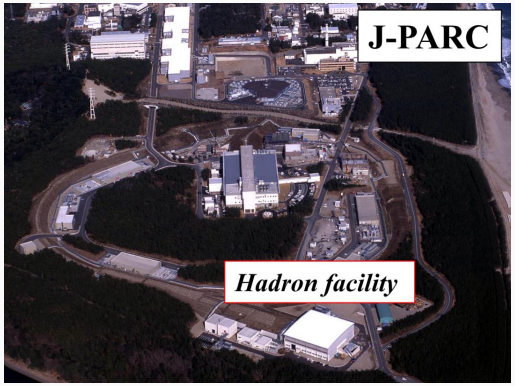
Future work

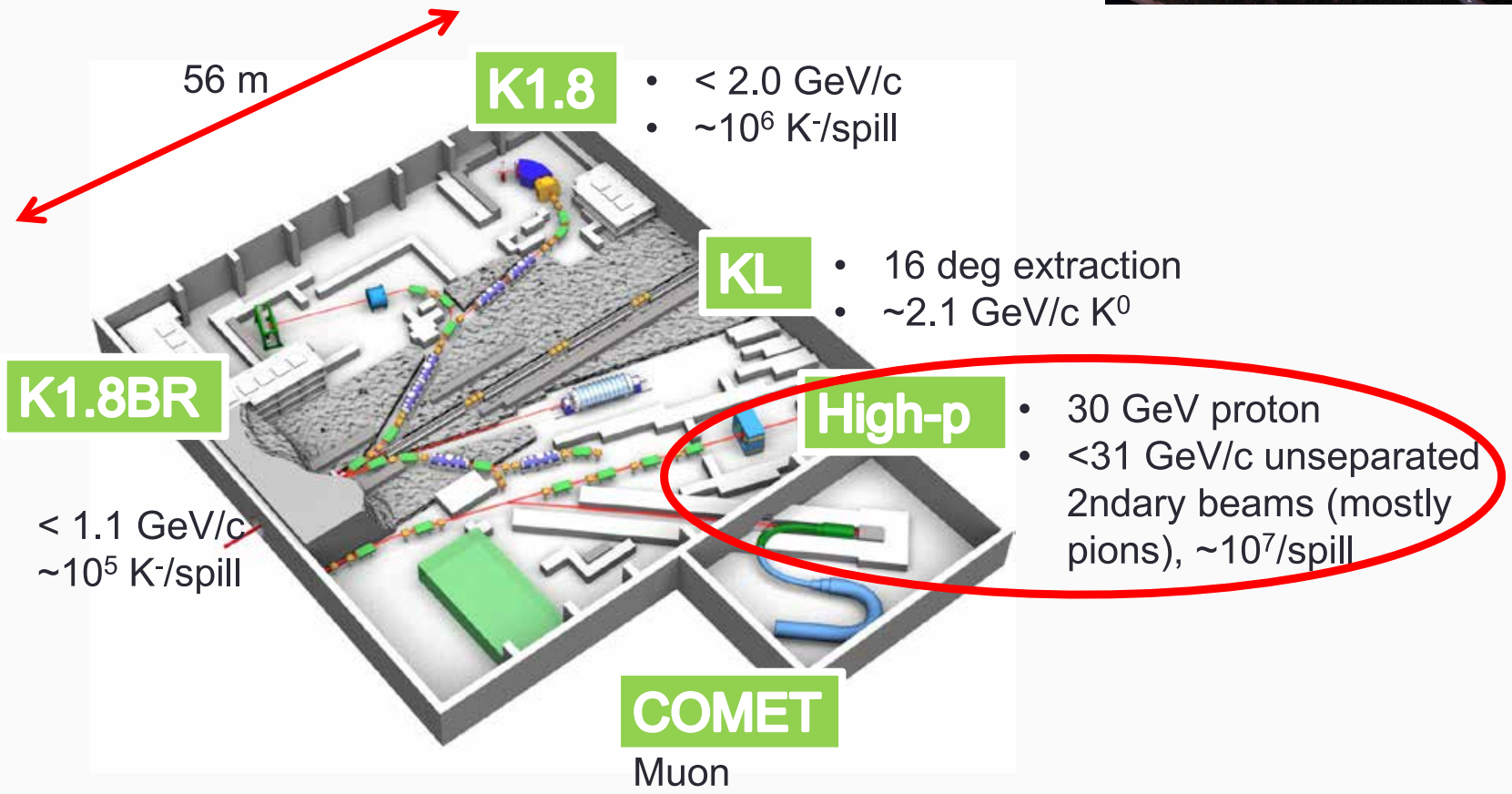
- Performance evaluation with many processing nodes and realistic workloads is in progress.
- Development of applications for physics experiments
  - J-PARC E16: triggered DAQ from 2020 → **combined DAQ** from 2023
  - J-PARC E50: commissioning of detectors and **streaming DAQ** with a small setup in 2023

Backup

J-PARC hadron experimental facility (HEF)

- Particle and nuclear physics
- Intense beam of p, K, π, μ





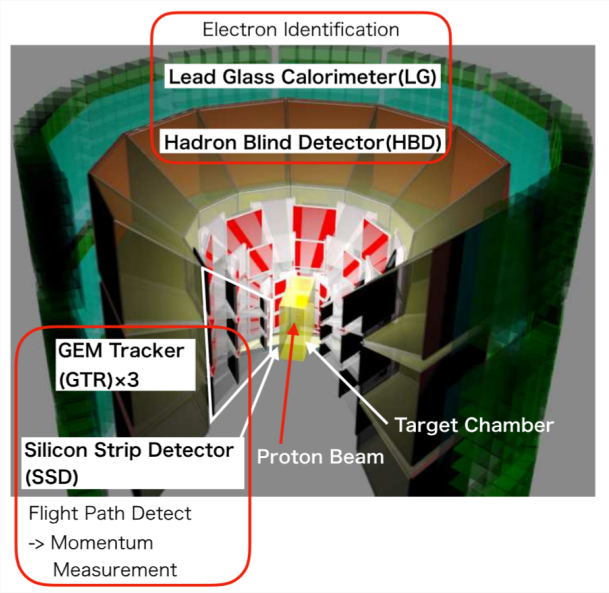
Hadron physics experiments at J-PARC high-p beam line

J-PARC E16 (ongoing experiment)

- **Physics** Hadron mass in medium
- **Beam** 30 GeV 5×10^9 protons/sec
- **DAQ** Triggered (+ streaming)
  - Detector channels: 150,000
  - Trigger channels: 2,600
  - 10 MHz reaction → trigger rate: 1–2 kHz
  - Data rate 1–3 GB/spill

As of Aug. 2022, ~1/3 of full detectors are installed.

Only SSD electronics (XYTER2 of GSI-CBM) supports self-triggered readout.



J-PARC E50 (future experiment)

- **Physics** Diquark correlation through charmed baryon spectroscopy
- **Beam** 20 GeV/c, 30 M π^-/sec
- **DAQ** Streaming
  - Detector channels: 25,000
  - Trigger channels: 5,000–25,000
  - 1.5 MHz reaction → trigger rate: 10–20 kHz
  - Data rate: 10–20 GB/spill (w/o trigger)
  - Reduction by software filtering: 100-200 MB/spill