Implementations of streaming DAQ on actual detector systems

Y. Igarashi, M. Dozono, R. Honda, N. Kobayashi, C. S. Lin, S. Ota, S. Y. Ryu, K. Shirotori, H. Sendai, T. N. Takahashi

***Abstract*—Detector systems in modern nuclear and particle experiments must handle a large number of channels and high data rates. In addition, the requirements for trigger systems have become more complex; thus faster decision making is required.**

Streaming readout and software-based triggering on distributed computers is a natural solution to these problems owing to recent technological advances. Therefore, we developed a widely usable streaming data acquisition (DAQ) system based on FairMQ and the Redis key value database. The DAQ system is relatively simple and can be developed and operated by a few people. It works with the cooperation of a multitude of elementary functional processes, and their connections are automatically set up based on the port name, service name, and connection information in a database.

The DAQ system was applied to actual detector systems, the RCNP Grand Raiden spectrometer, and the J-PARC E50 detector system beam test with streaming readout and online software trigger. The two detector systems, consisting of plastic scintillation counters and drift chambers, were read out by a field-programmable gate array based streaming readout time-to-digital converter and distributed synchronized clock, where the charge information was obtained from the time-over-threshold. The streaming DAQ worked well, including the online trigger, on 12 core server computers.

The software part of the DAQ was also tried to read a triggered DAQ system for a cylindrical drift chamber for the J-PARC COMET experiment. This application worked well with sufficient performance.

We further demonstrate the streaming-capable DAQ system and its actual application, which is mainly from the software side.

***Index Terms*—Data acquisition, System integration**

I. Introduction

The detector systems used in nuclear and particle physics must manage multiple channels and the high data rate of modern experiments. In addition, the trigger logic requirements have become complex and the decision should be faster. Software-based triggering on distributed computing is a natural solution to these issues in recent technological advances. Experiments to be performed in the near future at the Hadron Experimental Facility (HEF [1]) of Japan Proton Accelerator Research Complex (J-PARC), such as J-PARC E50 [2] require a system that is tens of times faster than the DAQ based on network event building [3] that has been available at HEF. On the other hand, experiments in the Research Center for Nuclear Physics (RCNP) of Osaka University should upgrade their DAQ system to be more flexible and significantly fast. Thus, we developed a widely usable streaming data acquisition (DAQ) system for near-future experiments. The DAQ system consists of three parts, the Field-Programmable Gate Array (FPGA) based streaming readout front-end electronics, FPGA-based clock synchronization and distribution system, and streaming-capable DAQ software. [4] We attempted to apply the developed streaming-capable DAQ system to three detector systems: a spectrometer called Grand Raiden, which is located in RCNP, the detector test system for the J-PARC E50 experiment, and the cylindrical drift chamber of the J-PARC E21 COMET experiment. In particular, we developed streaming DAQ system and reported its application on actual detector systems, especially on the software part.

Manuscript received May 26, 2024.

This work was supported by JSPS KAKENHI Grant Numbers 20K04005, 23H01206 and 22H04940.

This research was supported in part by the Yamada Science Foundation.

Y. Igarashi, R. Honda, C. S. Lin, H. Sendai are with Institute of Particle and Nuclear Studies, J-PARC Center, High Energy Accelerator Research Organization, Tokai-mura, Naka-gun, Ibaraki, Japan (e-mail: youichi.igarashi@kek.jp).

N. Kobayashi, S. Ota, S. Y. Ryu, K. Shirotori, T. N. Takahashi are with Research Center for Nuclear Physics, Osaka University, Ibaraki-shi, Osaka, Japan.

M. Dozono is with Graduate School of Science Division of Physics and Astronomy, Kyoto University, Kyoto-shi, Kyoto, Japan.

II. Readout electronics devices

To continuously acquire detector data, we used the following two types of front-end electronics(FEE): The first is an FPGA based clock and time synchronization system named MIKUMARI [5], and an FPGA based Time-to-Digital Converter (TDC) that can measure the Time-Over-Threshold (TOT). Both are running on a general purpose FPGA board AMANEQ [5]. The TDC can continuously read data in heartbeat frames (HBF), which are time frames divided into heartbeat delimiters without an absolute reference time, for continuous data acquisition. The heartbeat delimiter is sent every 524.288 us of a 16-bit counter driven by a 125 MHz clock. The TDC data of the detector signal coming in a heartbeat frame is timed with respect to this heartbeat delimiter. The TDC is synchronized with a 10 ps jitter by a clock distributed by a Clock-Duty-Cycle-Modulation (CDCM) based clock synchronization system. Data transfer from FEE is performed via TCP/IP implemented in the FPGA. There are two types of TDC. 1 ns TDC with multi-phase clock counting and High Precision TDC (HR-TDC), which has the intrinsic timing resolution approximately 20 ps in sigma using a tapped delay method. Both TDCs can measure TOT. The TDC was used for drift chamber readings and the HR-TDC was used for time-of-flight (TOF) measurements.

This readout system allows the DAQ server to read the FEE data using a common network switch. The FEE and the network switch are connected via 1 Gbps or 10 Gbps optical

fiber Ethernet, and the DAQ server and the network switch are connected via 10 Gbps optical fiber Ethernet. The FEE and

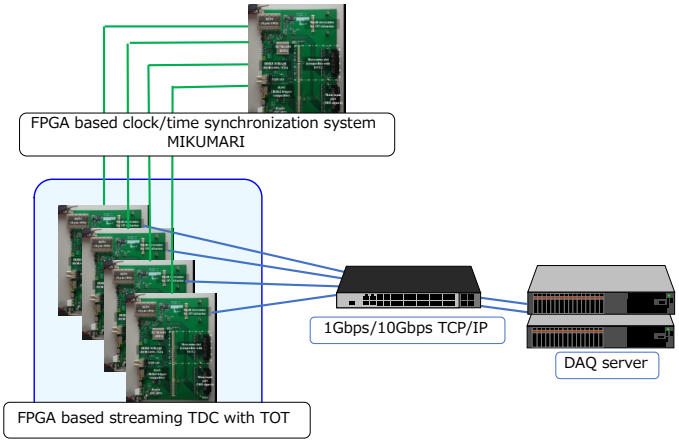

Fig. 1. Front-end readout electronics and their schematics; The front-end readout electronics consists of a TDC, a high-resolution TDC and a clock board to synchronize them, all built on a general-purpose FPGA board AMANEQ.

DAQ system used is shown in Fig.1. Data from the detectors could be read continuously with this system,

III. Streaming Capable DAQ Software

A. Used technologies

The DAQ system is designed to function in a coordinated manner with numerous simple process units, with the objective of handling the streaming data flow through multiple computers. The development of the system required the bottleneck-free mesh connection process communication and load distribution for the processor. Furthermore, the system required an overall process management and control. To achieve these objectives, we employed three distinct technologies.

1) ZeroMQ: We used ZeroMQ [6] for interprocess communication to realize one-to-many and many-to-one communication across computers. ZeroMQ is a universal messaging library that provides N-to-N communication as a message queue on TCP/IP and UNIX interprocess communication (IPC). It also supports many communication models useful for DAQ, such as fan-out and publisher-subscriber (PUB-SUB). In addition, the queue of the message queue acts as a data buffer to absorb the difference of processing speed among processes. These features were also advantageous for DAQ. ZeroMQ was well-suited to the communications on DAQ.

2) Redis: Redis [7], an open source key value database, was adopted to manage the state of many processes. Redis is one of the well-known key value databases and has a memory-oriented design and is fast. It has good real-time performance. Moreover, it has features desirable for DAQ, such as key-space notification, which notifies clients of changes in database values, and real-time PUB-SUB communication.

3) FairMQ: FairMQ [8] was used for the finite-state machine and control of the DAQ process. FairMQ is a software framework developed for experiments performed at GSI/FAIR, which includes a finite-state machine and its control mechanism. FairMQ can be extended with plug-ins that can be used to build DAQ software.

We developed a streaming DAQ software framework that combined these three technologies. We used ZeroMQ for mesh communication between processes and Redis for overall management. Each DAQ process has a finite state machine based on FairMQ, and they work cooperatively under the control of Redis.

B. Structure of the DAQ process

We developed three FairMQ plugins to communicate the Redis database: “DAQ Service”, “Metrics”, and “Parameter Config”. Figure 2 shows the structure of a DAQ process made up of FairMQ devices.

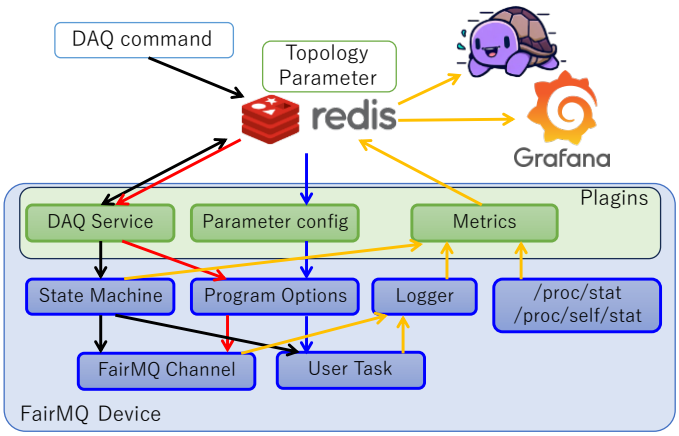

Fig. 2. The structure of the DAQ process; The round squares represent a single DAQ process. The arrows show how the communication with Redis is conveyed through the plugin.

1) DAQ Service plug-in: DAQ Service plug-in has two functions. The first is RUN control, and the second is service discovery. The DAQ Service plug-in receives PUB-SUB messages from Redis and transitions the state of its own processes. This allows DAQ processes to change to RUN state, IDLE state, and intermediate states in a unified manner. Service discovery is a mechanism that allows DAQ processes to automatically find other processes with which they need to connect. This is achieved by information in a database that tells it which process to connect to and on which port. The DAQ process searches for ports to connect to each other based on the endpoint and link information in the database and constructs the overall connection. Therefore, DAQ processes are controlled and managed by communicating with the database via the DAQ Service plugin.

The distinctive advantage of the DAQ system is the semi-automatic self-configuration of the communication connection by this DAQ Service. The database contains service name, port name, connection method, and information about connection service. This information is called topology.

Figure 3 shows an example of key values to configure the worker model. The system was organized and connections were made as shown in Fig. 4 from the key values.

This mechanism allows for the configuration of connections independent of the number of processes. Furthermore, it allows for highly flexible DAQ configuration using simple, fewer lines to describe.

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

Fig. 3. A sample of the port name and service name and their connection information in the database

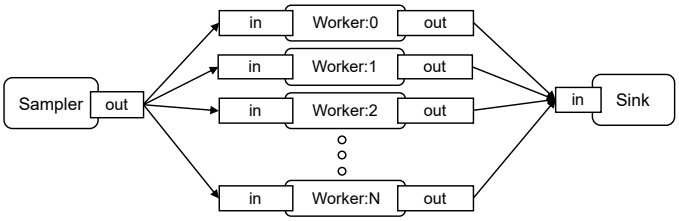

Fig. 4. This image shows a connection generated from the information in the database.

2) Metric plug-in: The Metric plug-in reports status within a process. The amount of data input to the port, the amount of data output, CPU usage, and memory usage are written to the database in Redis Time series every unit of time. The written data are displayed on the Web using a database visualization tool and is used to understand the status and detect system anomalies.

3) Parameter Config plug-in: The Parameter Config plug-in provides the ability to read arbitrary parameters from a database. The database contains initialization data for front-end electronics and experimental parameters that can be read and used by individual processes. The parameter input is integrated with the command line options; thus, the same parameters can be entered from the command line.

4) User interface: A Web based controller was developed as an interface to control DAQ. It works as a Web server, communicating with a Web browser via WebSocket and with the database to issue RUN control commands and monitor the status of each process. The status of the detector and various monitoring information are analyzed by the online data analysis process, etc., and recorded in the database. The recorded time series data and histograms are displayed in a browser using a web-based visualization tool, such as SlowDash [9]. SlowDash is a tool that displays data in the database in a Web browser in an easy-to-view format. It was developed for particle beam measurement experiments and to support the display of time series data as well as histograms and maps.

C. DAQ processes

A typical process configuration of the DAQ is shown in Fig. 5. Any number of processes can be executed, depending on the detector system, on any computer connected to the network. Each process has one input port and two output ports. One output port can be used for the main data stream in a push-pull pattern where all data is sent, and the other can be used for data branching. The second output port can also be used in a pub-sub pattern to send data to a data quality monitor, such as an online monitor, without disturbing the main stream of data. This structure enables any DAQ process to access the data for the purposes of debugging and monitoring.

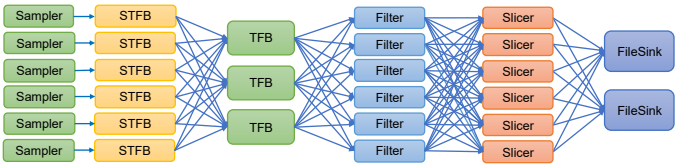

Fig. 5. The common DAQ software configuration; The circular squares represent each DAQ process, while the arrows indicate the ZeroMQ communication between them.

The functions of each DAQ process are as follows.

1) Sampler: The sampler reads data from the front-end electronics, converts them to ZeroMQ messages, and forwards them to the next process. This process is lightweight and also works as a data buffer for the next process.

2) Sub-Time Frame Builder(STFB): The Sub-Time Frame Builder separates the data sent from the sampler into heartbeat frames and combines several heartbeat frames to create a subtime frame. The created sub-time frames are transferred to the next process.

3) Time Frame Builder(TFB): The Time Frame Builder creates a time frame by combining sub-time frames sent from multiple STFBs into one time frame with the same time frame number. The time frame contains data from all detectors with the same time frame number. The created time frame is transferred to the next process.

One-to-many communication between STFB and TFB does not use ZeroMQ's PUB-SUB distribution for exact time frame matching, but point-to-point communication with strict round-robin according to the ZeroMQ time frame number.

4) Filter/Online software trigger: The filter process finds events of interest in the data in the incoming time frame and adds the time to the time frame as the trigger time. The data with the trigger time appended is transferred to the next process.

5) General logic online triggering filter: First, a widely available generic logical trigger is implemented as a filter process. It reads from the database a front-end ID, channel number, time offsets, trigger logic expression, and width of the trigger window as well as creates a look-up table (LUT) from the calculation results of the logic expressions in all cases. Mark the time when the signal arrives on the array and satisfy the trigger logic equation by scanning the array and determining true/false with the LUT. The position in the array where LUT returns true is recorded as the trigger time. With this mechanism, general-purpose logic triggers can be created from signals with arbitrary combination logic. The fields and values to be read from the database and their sample values are shown in TABLE I.

6) Time Frame Slicer (TFS): The Time Frame Slicer takes the data of all detectors near the trigger time added by

TABLE I
THE FIELD AND VALUES IN THE DATABASE FOR THE LOGIC FILTER; AS A SAMPLE, THIS SHOWS THE CASE OF TAKING COINCIDENCE FROM SIX SIGNALS.

Key: parameter:LogicFilter	
Field	Value
trigger-signals	(0xc0a802a9 8 0) (0xc0a802a9 10 0) (0xc0a802aa 32 0) (0xc0a802aa 33 0) (0xc0a802aa 34 0) (0xc0a802aa 35 0)
trigger-expression	(0 & 1 & 2 & 3 & 4 & 5)
trigger-width	3

the filter process and composes an event. It deletes TDC data except around the trigger time, but leaves in the data the parts containing flags, such as various headers and the heartbeat delimiters necessary for later analysis. The data is then transferred to the next process. The operation of the Time slice is shown in Fig. 6.

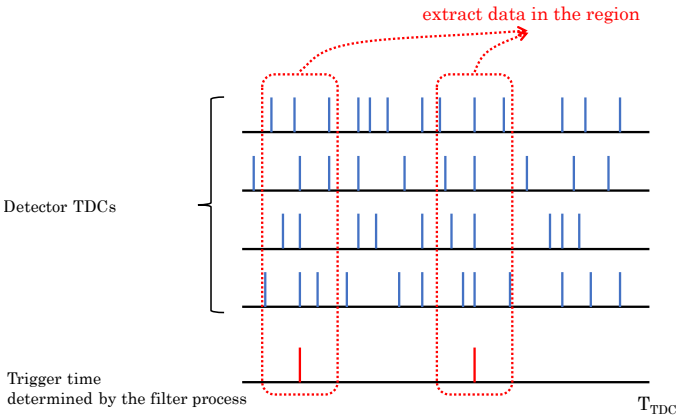

Fig. 6. The operation of Time Frame Slicer; The horizontal axis represents the TDC values. Short vertical lines indicate the position of hit values. The Time Frame Slicer extracts data closed to the trigger time.

7) FileSink: FileSink records the received data in a file, which can be compressed if necessary using the Zstandard (zstd) library. [10]

IV. IMPLEMENTATIONS

A. RCNP Grand RAIDEN

1) First trial to apply the detector system: The first implementation at the streaming DAQ was applied to the spectrometer named Grand Raiden at the Research Center for Nuclear Physics (RCNP), Osaka University. An overview of Grand Raiden and the location of the new FEE and DAQ equipment is shown in Fig. 7. Drift chambers for momentum analysis and a hodoscope counter to measure the time of particle arrival were installed in the focal plane of Grand Raiden. A trial was made to apply a streaming data acquisition system to this detector system.

The FEEs used are two 64-channel HR-TDCs with TOT and eight 128-channel TDCs with TOT. The FEEs are synchronized by the clock synchronization system "MIKUMARI". The DAQ software consists of Sampler 10 processes, STFB 10 processes, TFB 3 processes, Filter 16 processes, and FileSink 2 processes for the streaming DAQ. The streaming

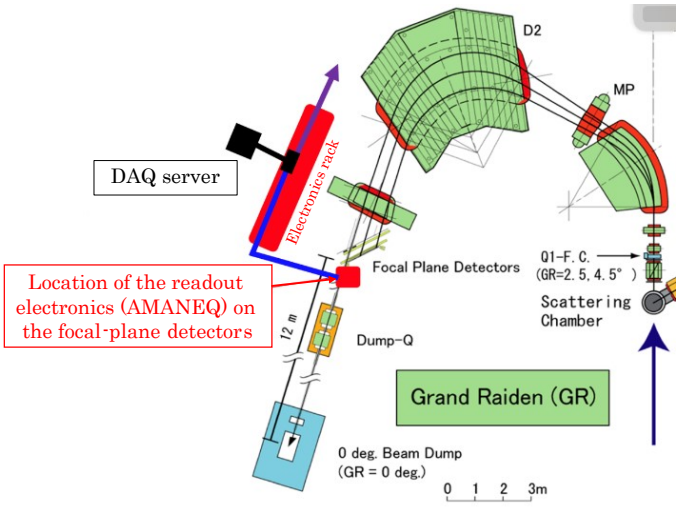

Fig. 7. Grand Raiden; This figure shows the top view of the Grand Raiden spectrometer and the locations of the installed FEE and DAQ servers.

DAQ was conducted using this configuration. The DAQ server has an AMD EPYC 74F3 24-Core Processor @ 3.5 GHz. The DAQ was performed on a single server computer. The behavior of the software trigger process, a feature of streaming DAQ, was examined in detail. This process involves complete software replacement of the trigger circuit configured using the Nuclear Instrumentation Module (NIM) standard modules, for example, in traditional triggered DAQ systems.

The trigger was done with a coincidence of four counter signals, which at this time were processed using simple hard-coding. We focused on the load balancing of the filter processes and the processing time for the triggering process. The load balancing of the 16 filter processes is shown in Fig. 8. The distribution of processes from TFB to Filter is performed

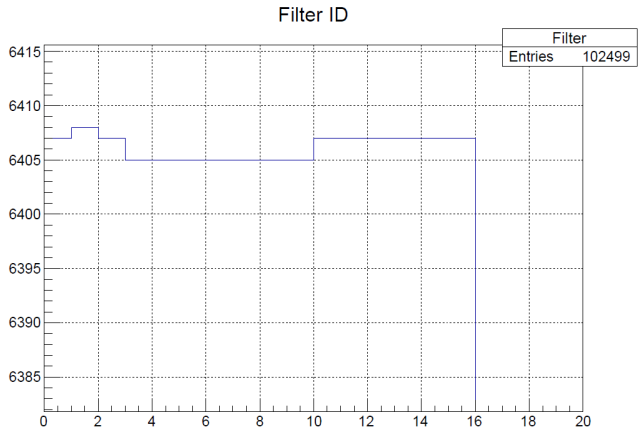

Fig. 8. The load balancing of filter processes; The horizontal axis represents the identical number of filter processes, while the vertical axis represents the number of tasks completed. Sixteen Filter processes were running at the data acquisition.

in a round-robin scheduling, but if the queue is full, the queue is skipped. Although it is a simple algorithm, the load is well distributed, as shown in Fig. 8.

Figure 9 shows the processing time of a sub-time frame consisting of five HBFs. The structure of the processing time depends on the number of HBFs, but on average the processing takes approximately 4.6 ms. The time for 5 heartbeat frames is 2.62 ms, indicating that processing can be conducted in more than two Filter processes.

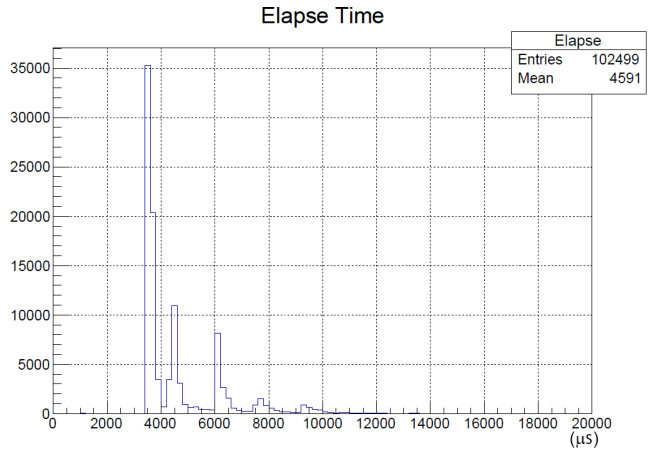

Fig. 9. The elapse time of the Filter process; The horizontal axis represents the elapsed time, which is expressed in microseconds. The vertical axis represents the degrees. The mean time for the software triggering process is 4.59 ms.

This trial confirmed that streaming DAQ works well in nuclear experiments.

2) RCNP E585 actual experimant data taking: As previous trials had shown that the streaming DAQ worked well, it was adopted by the RCNP E585 [11] nuclear experiment at Grand Raiden to take data from the experiment. The DAQ configuration became complex to achieve reliable physics measurements. Mainstream Sampler, STFB, 11 processes TFB, Filter process, and 8 processes, at this time, TFS was introduced to reduce event organization and recorded data. In addition to the main data stream, a software scaler from the STFB was connected to monitor the respective count rates. Unbiased data, prescaled to 1 / 1000, was recorded from TFB. A diagram of the configuration at this time is shown in Fig. 10

DAQ was performed using two computers: a DAQ server (AMD Ryzen 9 3099XT 12-Core 3.8 GHz) and a file server (Xeon E5-2967 v2 48-COre 2.7 GHz). SlowDash [9], a web-based visualisation system, was introduced at the time as a data online monitor. SlowDash displayed information written to the database in a web browser. It monitored data flow between each connection, FEE error flags, scaler values, etc.

The data acquisition of the experiment was successful, with the experimental data being collected 40 times faster than with the previous system.

B. J-PARC E50 detector test in J-PARC Hadron Hall

The next trial is the detector test bench for the J-PARC E50 experiment [2] planned for the J-PARC Hadron facility High-p beamline. This detector system was installed in the Hadron Hall K1.8 beamline. The setup of the detector test bench for the E50 experiment is shown in Fig. 11.

The TOF hodoscope counters TOF(U), LTOF(D) and TOF(D) and the drift chamber KLDC and BDC were installed at June 2023. FPGA boards with two 128-channel HR-TDCs, 15 1920-channel TDCs and three 96-channel readout clock synchronization systems were used to read out these detectors.

The DAQ configuration was typical, with 20 processes Sampler and STFB, 4 processes TFB, 16 processes Filter, and 2 processes FileSink. The Filter process was updated with arbitrary combinatorial logic triggering by look-up memory. A data quality monitor was developed to receive data from the Filter process and display detector hits, etc., to monitor the state of detection. The distribution of the processing time of the Filter process at this time is shown in Fig. 12.

The DAQ worked well with streaming readout, including a combination logic software trigger. In this measurement, the STF consists of 1 HBF. The trigger was applied by expression (1), which determines the passage of the beam particles in the detectors.

$$\begin{aligned} Trigger =& ((TOF(U)1_L \wedge TOF(U)1_R) \\ & \vee (TOF(U)2_L \wedge TOF(U)2_R)) \\ & \wedge (TOF(D)1_L \vee TOF(D)1_R \\ & \vee TOF(D)2_L \vee TOF(D)2_R \\ & \vee TOF(D)3_L \vee TOF(D)3_R) \end{aligned} \tag{1}$$

On average, 1 HBF is processed in 7.6 ms. Therefore, more than 15 Filter processes can be processed. Data are collected at an average rate of approximately 180 MB/s, which is close to the upper limit of SATA hard disks. Because the J-PARC accelerator beam had a spill structure (repetition interval 4.2 s, extraction 2 s), the data could be written at a rate of around 240 MB/s while the beam was extracted. This is believed to be because of the effect of the disk cache on the memory.

C. Cylindrical Drift Chamber of J-PARC COMET experiment

The software part of this streaming DAQ system should be able to be used for streaming DAQ as well as for conventional triggered DAQ by processing in event units instead of time frames. We have tried to take data with the streaming DAQ software on the readouts of the cylindrical drift chamber, which is currently being prepared and tested for the J-PARC E21 COMET experiment. [12] The CDC FEE, called RECBE, has a 48-channel TDC and 40-MHz FADCs and can send data out to the network in the same way as an FPGA TDC. It is triggered and driven by the trigger system of the COMET experiment and outputs TDC and charge data. The CDC is read out by 105 FEEs. The CDC readout setup is shown in Fig. 13.

The FEE is connected to the DAQ server (Intel Xeon E-2236 3.4 GHz 6 cores) via a 1G/10G network switch (FS.COM S3900 24F4S). Ninety-six of the 105 FEEs were used for the test, triggered by a cyclic pulser. The data read out used a mode with a fixed value of 6156B. The DAQ software was configured to develop a Sampler that processes data on an event-by-event basis, with the TFB used directly as the Event Builder. To observe the performance of the Event building, Sink did not write any files but only checked that the data

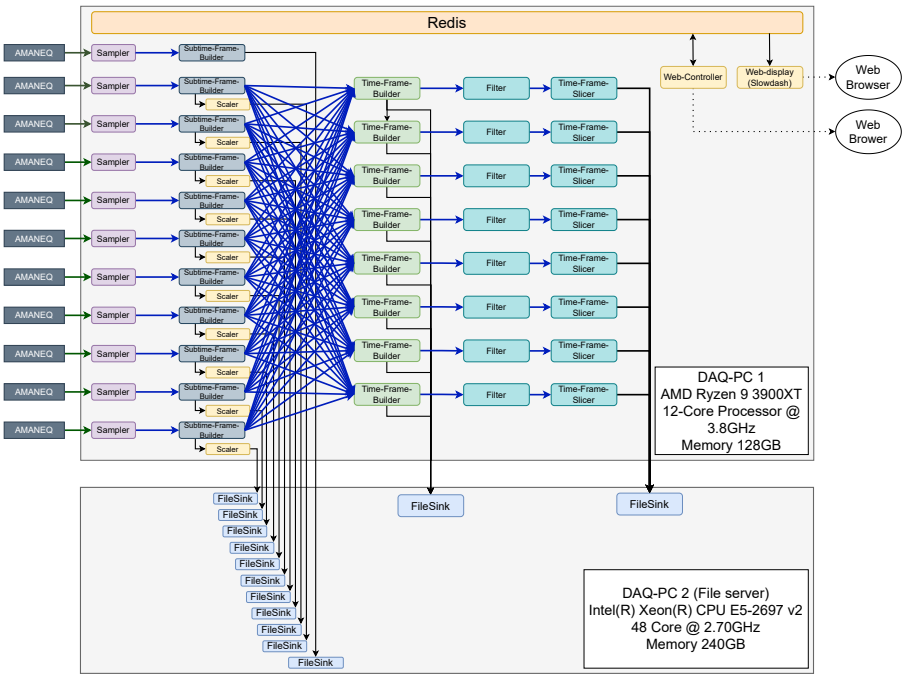

Fig. 10. DAQ configuration of RCNP E585 data acuisiton; The rectangular frame indicates the DAQ server. The circular square represents the DAQ processes that are executed on the server. The arrows illustrate the communication pathways.

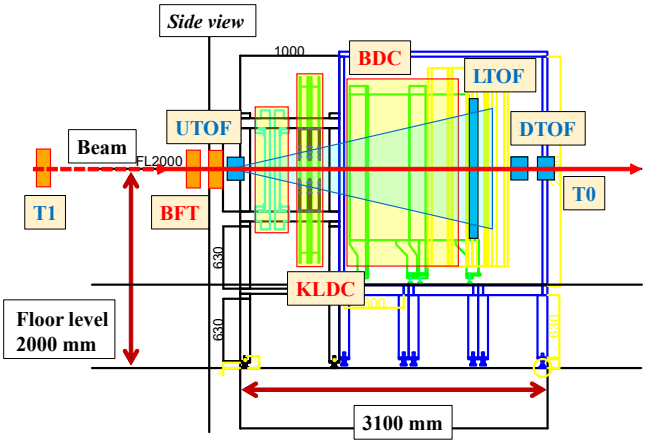

Fig. 11. The configuration of the J-PARC E50 detector test at the J-PARC Hadron Hall K1.8BR beamline; The TOF counters TOF(U), LTOF(D) and TOF(D) and the drift chamber KLDC and BDC were installed at June 2023.

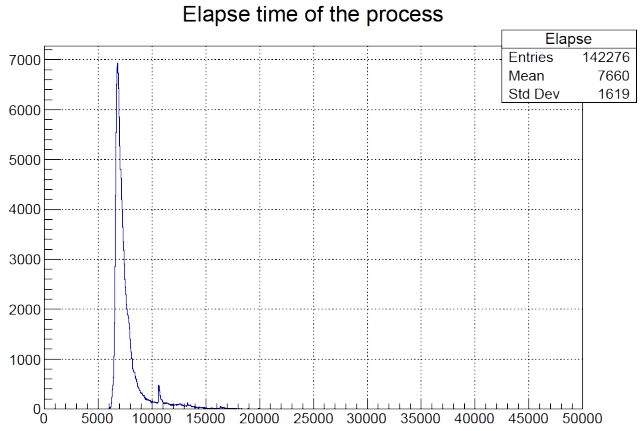

Fig. 12. The elapse time distribution of the combination logic filter; The horizontal axis represents the elapsed time, which is expressed in microseconds. The vertical axis represents the degrees. The mean time for the software triggering process by the combination logic is 7.66 ms.

were correct. The software configuration at this time is shown in Fig. 14.

The trial works well and it was established that the streaming DAQ software could also be applied to triggered DAQs. Figure 15 shows the throughput in event building compared to the number of TFB when collecting data on a single DAQ server. Event building was most efficient when using the TFB 3 process, reaching a throughput of 654 MiB/s.

V. Discussion

In conventional triggered DAQ, the complex triggering systems required for near-future experiments complicate the DAQ system. Their dead time makes efficient data collection difficult. To solve this problem, a distributed streaming DAQ system was developed running on multiple computers. By using ZeroMQ for interprocess communication, communication between a large number of processes could be achieved with high reliability and maintainability.

Furthermore, it was confirmed that ZeroMQ's simple round-robin load balancing, which skips queues when it is full, works well without central control in the case of triggered processes using combination logic.

In addition, by performing data acquisition on a real detector system, the following knowledge was obtained. We confirmed that software-triggered processes can operate on a practical

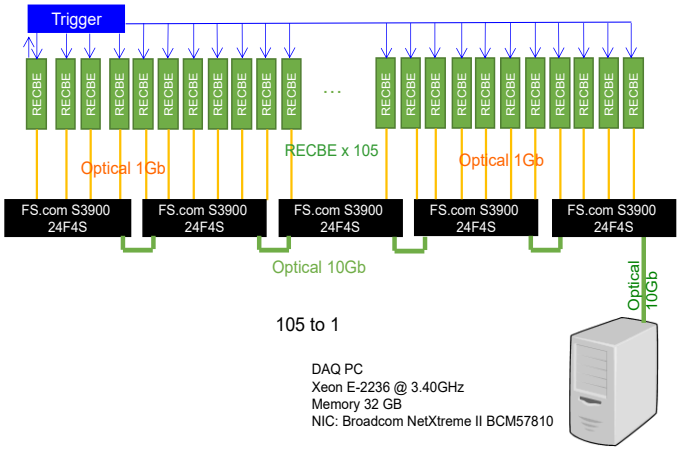

Fig. 13. The readout configuration of the COMET CDC; All readout electronics, designated as "Recbe," were connected to a DAQ server via 1 Gbps/10 Gbps network switches.

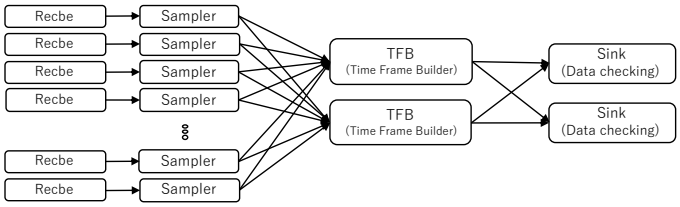

Fig. 14. The DAQ process configuration for the COMET CDC DAQ; When used with triggered DAQ, data is clearly separated into events by triggers. Therefore, STFB is not needed and TFB acts as an event builder.

number of server PC cores. It allows for high flexibility in triggering without accessing hardware. Even with higher beam rates, high throughput can be supported by distributing processing across multiple DAQ servers connected by a network with speeds of 10 Gbps or faster.

Currently, operations were conducted with the number of processes up to approximately 100 processes, but no major problems occurred within this range. A trouble that occurred was when sending small messages via ZeroMQ for each event, the excessive number of messages caused fluctuations in data throughput, resulting in reduced transfer performance. We resolved this issue by bundling the data for transmission.

The current TFB waits for 10 seconds, and if a Time Frame Build cannot be performed during that time, the Time Frame is discarded. If multiple discarded Time Frames exist owing to missing data caused by hardware problems, the performance of std::map in the C++ language became severely reduced owing to the excessive increase in data managed by std::map. Data acquisition can be more robust if the rejection of failed Time Frame data is done by data volume rather than by time.

In this study, we implemented and evaluated software triggers using combinatorial logic. The framework includes TFS for event separation and data reduction, enabling event discrimination at a higher-level analysis of events. Our future research will focus on further implementing this approach.

VI. Summary

We developed a streaming-capable DAQ system for nuclear and particle experiments. The system is based on FairMQ, and

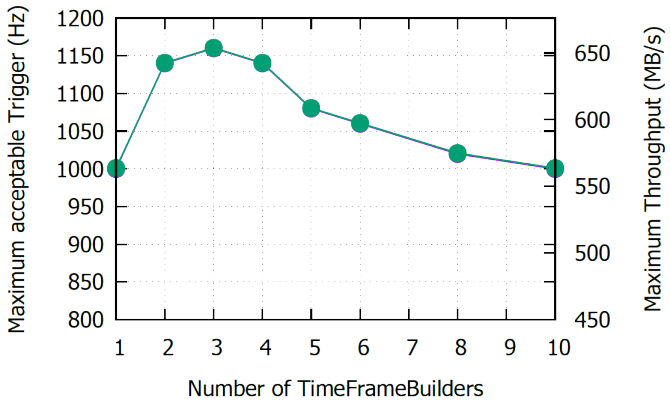

Fig. 15. The data flow throughput of the TFB as an event builder; The horizontal axis represents the number of TFB processes. The vertical axis represents the number of accepted triggers, which is corresponding to the throughput of the event building in megabytes per second.

is managed and controlled by Redis, a key value database. The system is decentralized and can be executed by many processes on various computers communicating with ZeroMQ. We applied this streaming DAQ system to the Grand Raiden RCNP spectrometer and the J-PARC E50 detector test bench in the J-PARC Hadron Hall and they performed well in streaming DAQ. The software part of the DAQ system is also applicable to triggered DAQ, which was demonstrated by applying it to the readout of the J-PARC COMET CDC.

VII. Acknowledgements

The authors express their gratitude to the members of the experimental groups of RCNP and J-PARC and the accelerator groups of RCNP and J-PARC. Furthermore, the authors acknowledge the financial support from the JSPS Kakenhi, which enabled us to conduct this research.

References

[1] "J-PARC Hadron Experimental Facility (HEF)", Accessed: May. 01, 2024. [Online]. Available: https://j-parc.jp/c/en/for-researchers/hadron.html
[2] H. Noumi et al., "Charmed Baryon Spectroscopy via the(π, D^{*-})reaction," [Online]. Available: https://j-parc.jp/researcher/Hadron/en/pac_1301/pdf/P50_2012-19.pdf
[3] Y. Igarashi et al. , *"An Integrated Data Acquisition System for J-PARC Hadron Experiments"*, IEEE Trans. Nucl. Sci., vol. 57, no. 2, pp. 618-624, 2010, DOI: 10.1109/TNS.2009.2037959
[4] T. N. Takahashi, Y. Igarashi, R. Honda, H. Sendai, *"Streaming DAQ Software Prototype at the J-PARC Hadron Experimental Facility"*, IEEE Trans. Nucl. Sci., vol. 70, no. 6, pp. 922-927, DOI: 10.1109/TNS.2023.3262061
[5] R. Honda, *"New Clock Distribution System Based on Clock-Duty-Cycle-Modulation for Distributed Data-Aquisition System"*, IEEE Trans. Nucl. Sci., vol. 70, no. 6, pp. 1102-1109, 2023, DOI: 10.1109/TNS.2023.3265698
[6] "ZeroMQ," [Online]. Available: https://zeromq.org/
[7] "Redis," [Online]. Available: https://redis.io/
[8] "FairMQ," [Online]. Available: https://github.com/FairRootGroup/FairMQ
[9] S. Enomoto, "SlowDash," [Online]. Available: https://github.com/slowproj/slowdash
[10] "Zstandard(zstd)," [Online]. Available: https://github.com/facebook/zstd
[11] M. Dozono and S. Ota et al. , "Searching for highly-excited pair vibration modes in ^{118}Sn using alpha-induced pair-transfer reactions," 2022, [Online]. Available: https://www.rcnp.osaka-u.ac.jp/Divisions/plan/b-pac/ex_appro/summary/pdf/E585.pdf

[12] COMET collaboration, “COMET Phase-I Technical Design Report,” 2020, PTEP 2020 3, 033C01, DOI: 10.1093/ptep/ptz125