Commissioning and Early Run 2 Experience of the Upgraded DAQ Backend System for the Belle II Experiment

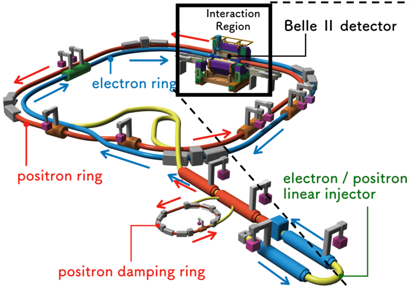
Dhiraj Kalita, Dmytro Levit, Ikuo Ueda, Matthew Barrett, Mikihiko Nakao, Ryosuke Itoh, Satoru Yamada, **Seokhee Park**, Soh Suzuki, Takanori Hara, Takuto Kunigo, Tristan Bloomfield

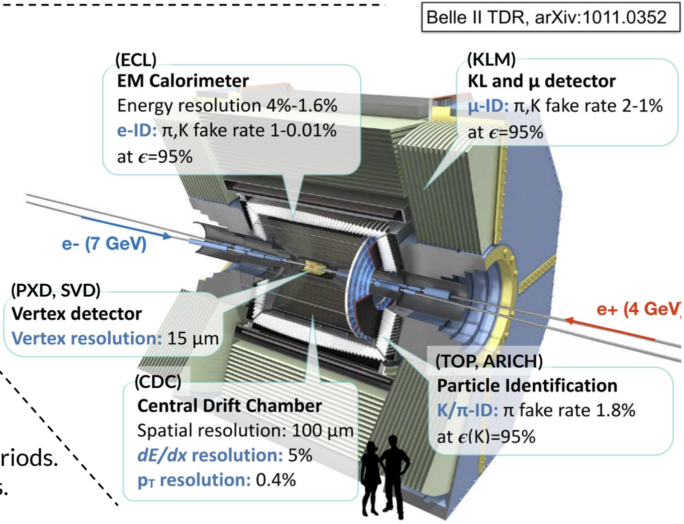
KEK

2024 Sep. 18th, Hokkaido University

79th JPS Meeting

Belle II and SuperKEKB



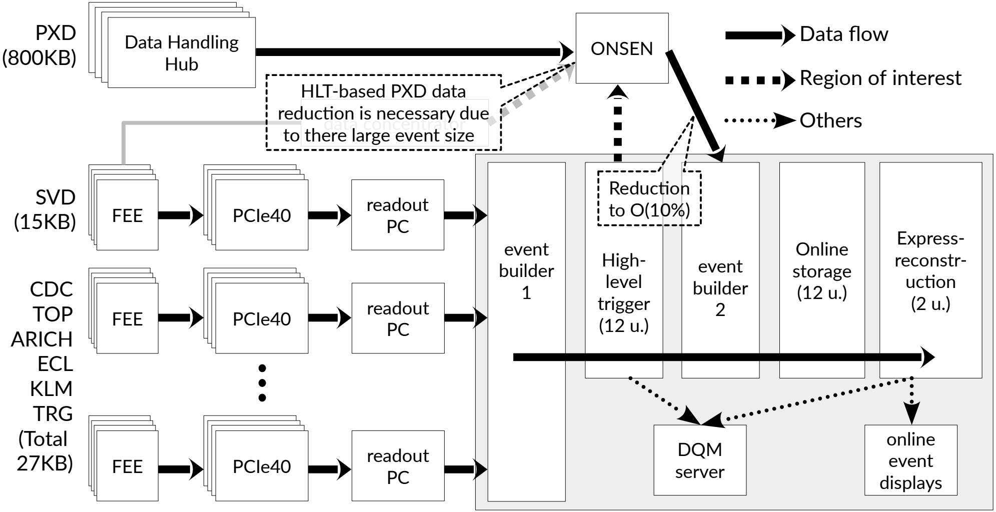


- Run1 (-2022)

  → Long shutdown 1 (2022-2024)

  → Run 2 (Feb. 2024-)
- After the LS1, we had 2024a/b run periods. 2024c run will be in a couple of weeks.

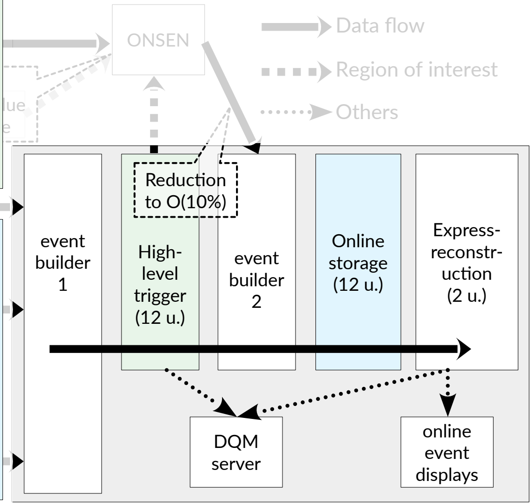
DAQ data flow



Introduction

- **High-level trigger (HLT)**
  - Making event selection and accepting / rejecting the event.
  - Framework update to support faulty event tolerance
  - Covered at the last year JPS by R. Itoh.

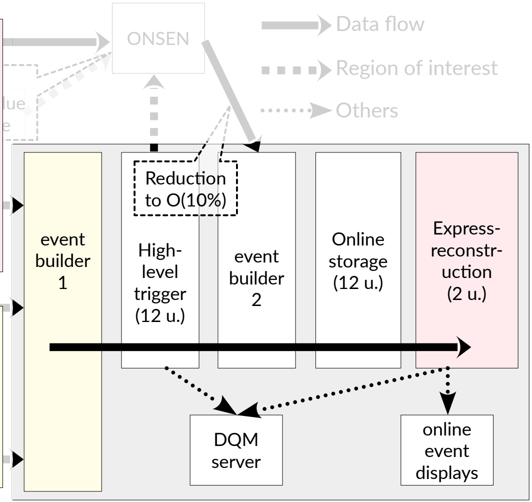
- **Onlise Storage (STORE)**
  - Save events from the event builder 2 (HLT and PXD)
  - Update the framework based on ZMQ library
  - Direct ROOT format support
  - Update the file transfer handshake between online to offline computing center



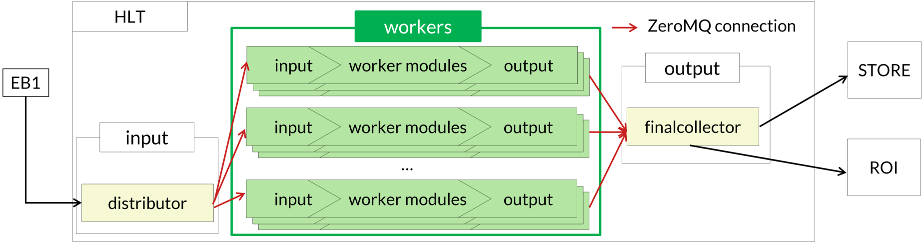
Introduction

- **Express-reconstruction (ERECO)**
  - Accept the sampled events (low performance than HLT) and provide additional monitoring information
  - Update the framework based on ZMQ library
  - Introduce a dedicated ERECO only for physics-tagged events from HLT

- **Event builder 1 (EB1)**
  - Performing event building before the HLT for all subdetectors except PXD
  - By updating the RX app structure, we allow to restart run except HLT and PXD to reduce the downtime



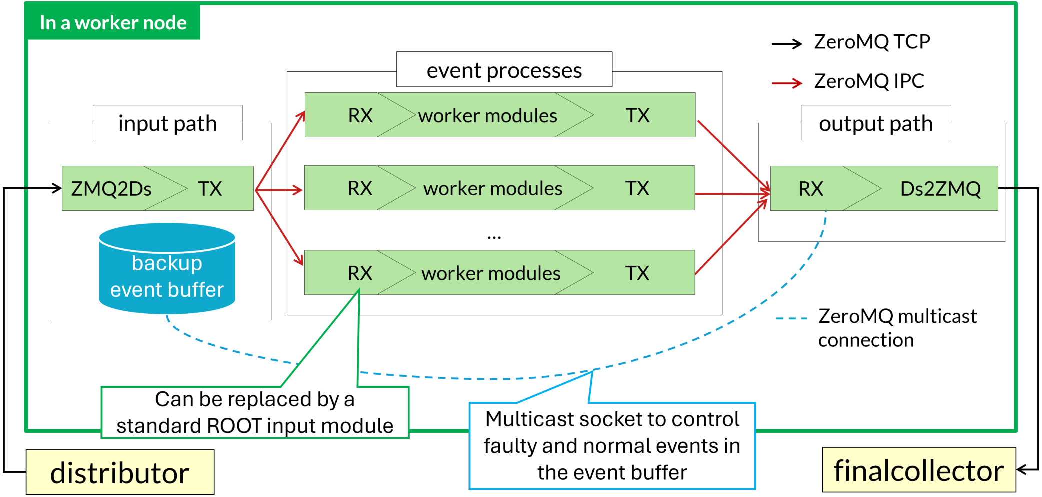
Key updates: HLT



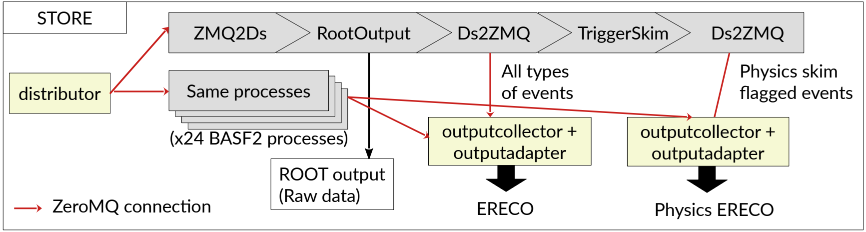
■ The worker section is upgraded.

- ▶ Before LS1: The worker processes in each work node are directly connected to the distributor and collector apps via ZMQ TCP/IP connection.
- ▶ Limitations
  - If one of the processes dies due to some reason, the event is lost and creates missing ROI for PXD → This causes permanent performance degradation until the run is restarted.
  - Hard to start the process individually for debugging

New HLT worker structure



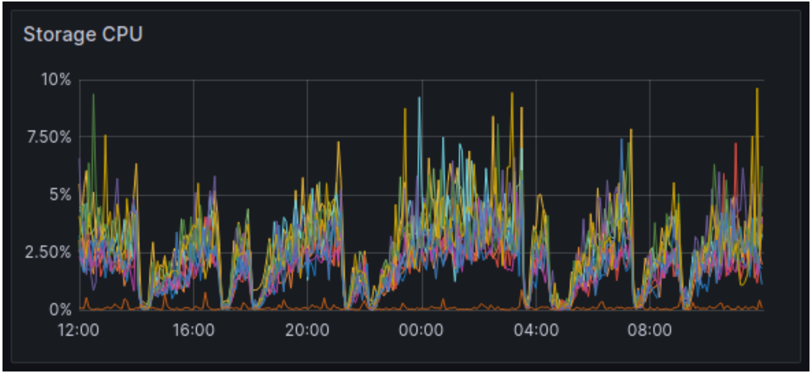
Key updates: STORE



- **Data distribution using the ring buffer + TCP/IP socket → ZeroMQ connections**
- **Single SROOT (custom Belle II format) → Standard ROOT format with compression**
  - Multiprocessing to achieve the online compression and multiple output files at the same time
  - Pros: Small file size, no additional offline processing
  - Cons: Large CPU usage for compression, requiring online side small-sized file merging
- **(New) Events categorization by the HLT results for ERECO**
- **Database-based online-offline file transfer handshake**

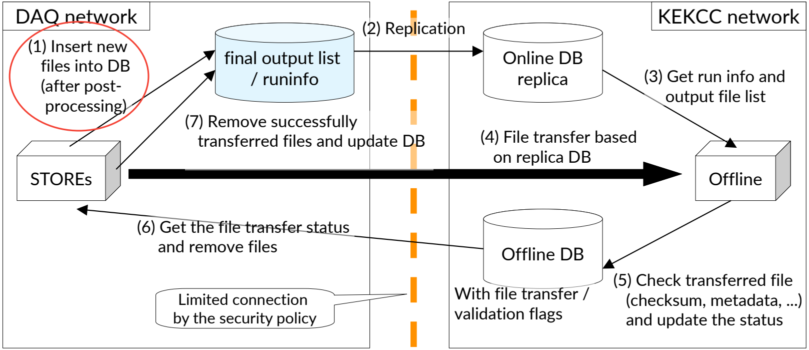
CPU usage in the operation

- **We are now early phase of the Run 2 operation, so the input rate is not so high.**
  - Maximum designed input rate: 30 kHz → 6-8 kHz during the early Run 2 opeartion
  - Roughly, 15-20% of the maximum design
- **With the condition, CPU usage is ~ 5% level → acceptable range.**



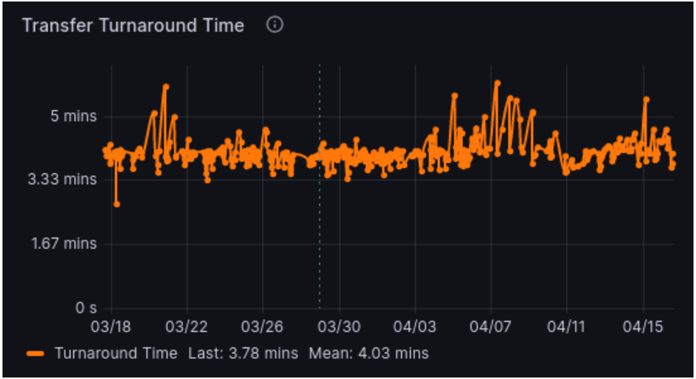
Handshake of file transfer from online to offline computing site

- **In Run 1, for the handshake of online to offline file transfer, we rely on the text file.**
  - New files, the status of file transfer, checksum kinds of information is listed in the text file.
  - We often get missing list of files or some troubles.
- **We finally decided to do not use the text file, and use database for the handshake.**
  - After a few early stage trouble, the handshake is running without any serious crash.

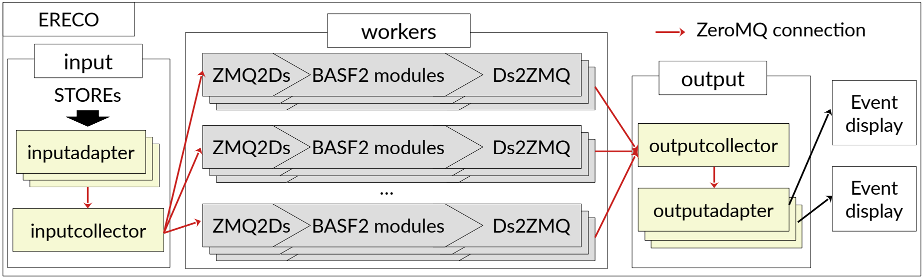


File transfer to offline computing site

- **The file transfers are done almost within 5 minutes**
  - Done by xrootd (Run 1: rsync)
  - Much faster than the previous text-based file listing & transfer
  - No additional format conversion and compression is needed from the offline computing site



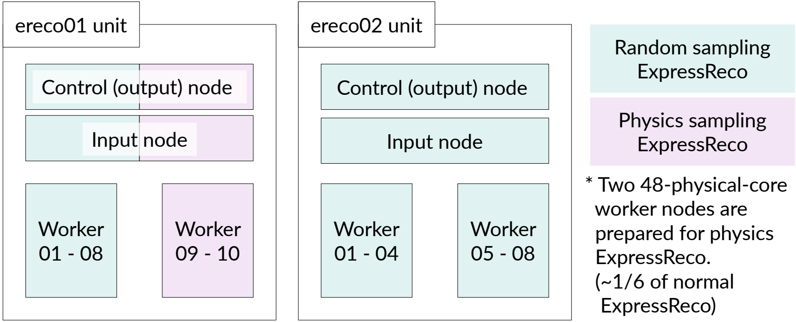
Key updates: ERECO



- **Data distribution using the ring buffer + TCP/IP socket → ZeroMQ connections**
- **Better maintainability and stability**
  - The operation is very stable.
  - Some old bugs in the previous system are gone
    - Slow DQM histogram update, run number mixing, silence crash, shard memory issue, ...
- **DQM and online event display for physics-tagged events**

Physics ERECO

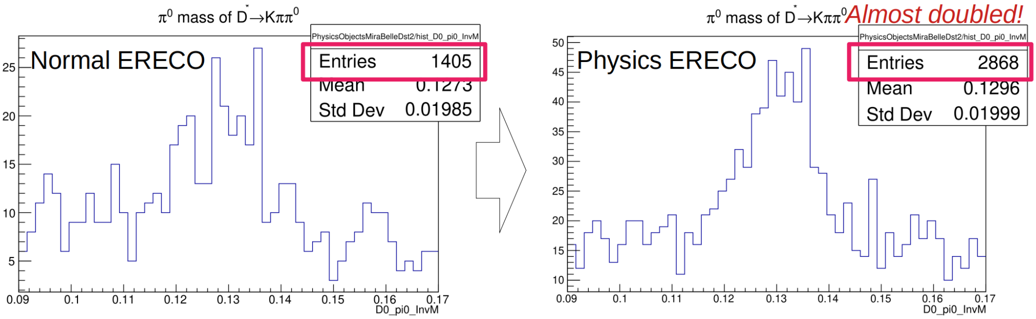
- **The ERECO performance is $O(10\%)$ of HLT → many events are randomly discarded.**
  - Prepare dedicated ERECO only for physics-tagged event for more statistics of DQM
- **The physics ERECO and one of normal ERECO share the same farm.**
  - Both ERECO share input and output (control) nodes.
  - Two worker nodes (~100-core) are prepared only for physics ERECO.



* Two 48-physical-core worker nodes are prepared for physics ExpressReco. (~1/6 of normal ExpressReco)

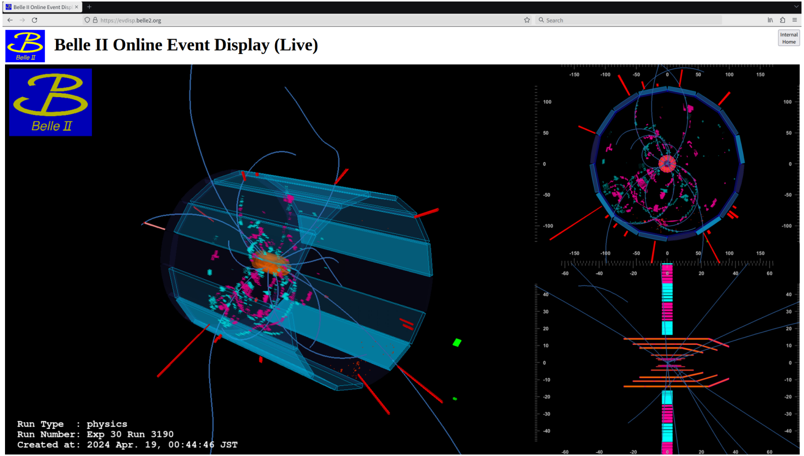
Physics ERECO DQM

- **The trigger lines for physics ERECO is now studied**
  - In the early phase of Run 2, input rate is small. So, we include as many as possible trigger lines.
  - The statistical enhancement is depending on the input rate and trigger line selection
- **Both normal and physics ERECO DQM files are stored**
  - Even with the low lumi, # event of physics info in the physics ERECO are double of normal one



Online event display

- **Public online event display is now running with the physics ERECO output**
  - We can provide only physics live events (available on **https://evdisp.belle2.org**)

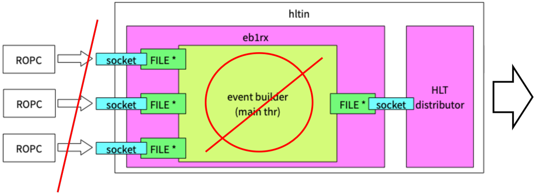


Partial run restart (Partial SALS)

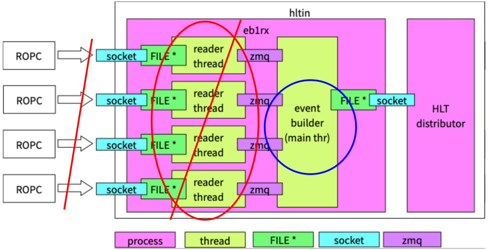
- STOP-ABORT-LOAD-START (SALS): Belle2 run restart procedure
  - The time of SALS is important to increase the DAQ running time.
- Currently, the most time-consuming parts are LOAD and STOP.
  - Loading takes 30 - 40 seconds → **except HLT, it takes less than 10s.**
    - Due to HLT worker process start up (performing geometry initialization)
  - Starting takes 15 - 20 seconds
  - Stopping takes 30 - 40 seconds
    - Due to DQM aggregation (especially for ERECO)
  - Aborting takes 20 - 30 seconds → **except HLT, it takes less than 10s.**
- Both time-consuming sources are under the HLT, so we implement "partial" ABORT and LOAD except HLT and PXD.
  - we can save ABORT and LOAD time → roughly 1 min.

Partial run restart (Partial SALS)

- To achieve the goal, we need to update the EB1 RX
  - the receiver app of the first event builder, which is in the HLT input node.
- We segmented the event builder into several threads connected via ZMQ to not close the connection to HLT when the readout system is restarted.
  - Better throughput by ZMQ is just a by-product in here.



If the connections to ROPC (detector) are dead, the main thread is also dead for reconnection.



If the connections to ROPC (detector) are dead, main thread is alive, and the program can restart the reader threads and recover the connections.

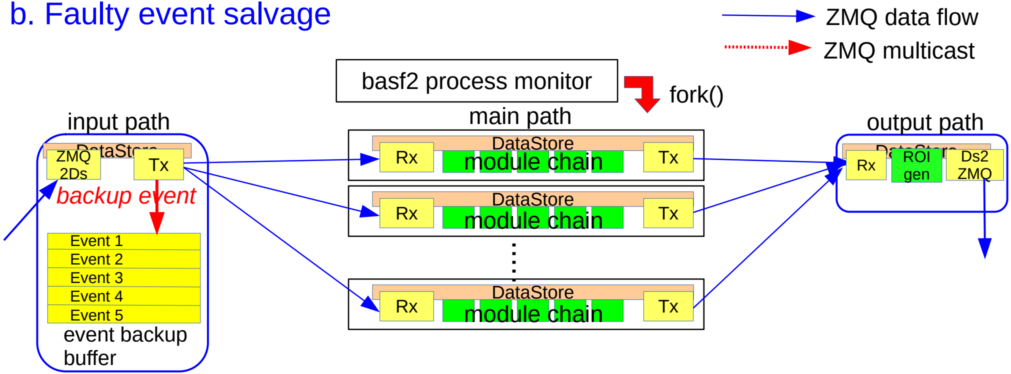
Conclusion

- **High-level trigger framework is upgraded to solve issues like missing ROI problems.**
- **Online storage and the Express-Reconstruction system is upgraded to use the ZeroMQ library already used in the High-level trigger.**
- **Online storage can create general ROOT output directly.**
- **A dedicated Express-Reconsturction system for physics is ready and creates more statistics of physics-tagged data quality monitoring histograms.**
- **The partial run restart is implemented and this can save around 1 minute per run restart.**
- **All the systems are running smoothly during the early Run 2 operation.**

Backup

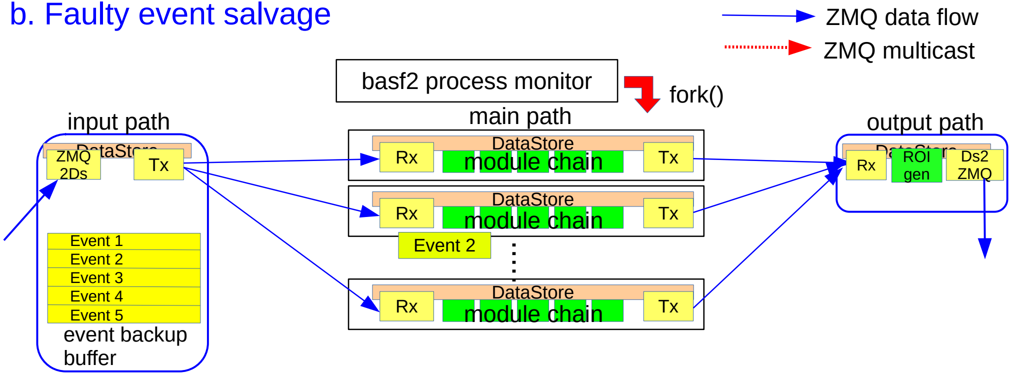
Faulty event salvage in the new HLT framework

b. Faulty event salvage



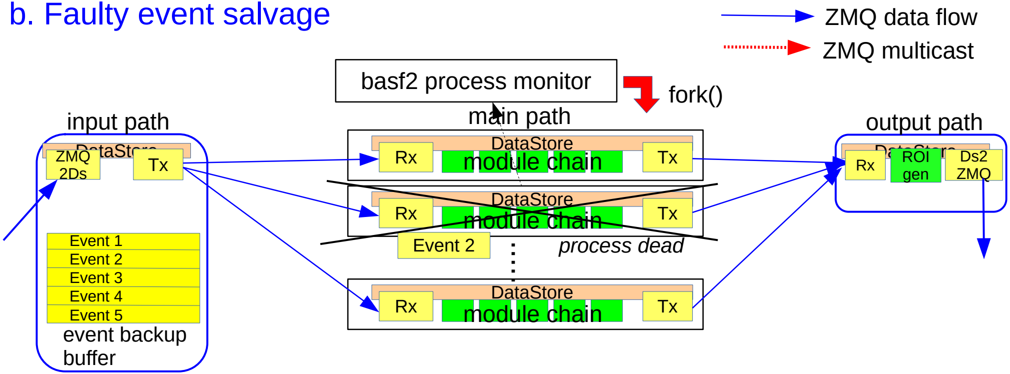
Faulty event salvage in the new HLT framework

b. Faulty event salvage



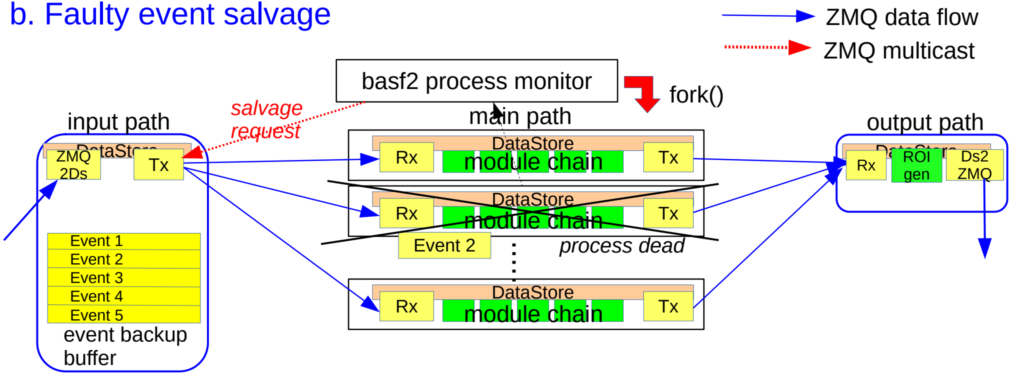
Faulty event salvage in the new HLT framework

b. Faulty event salvage



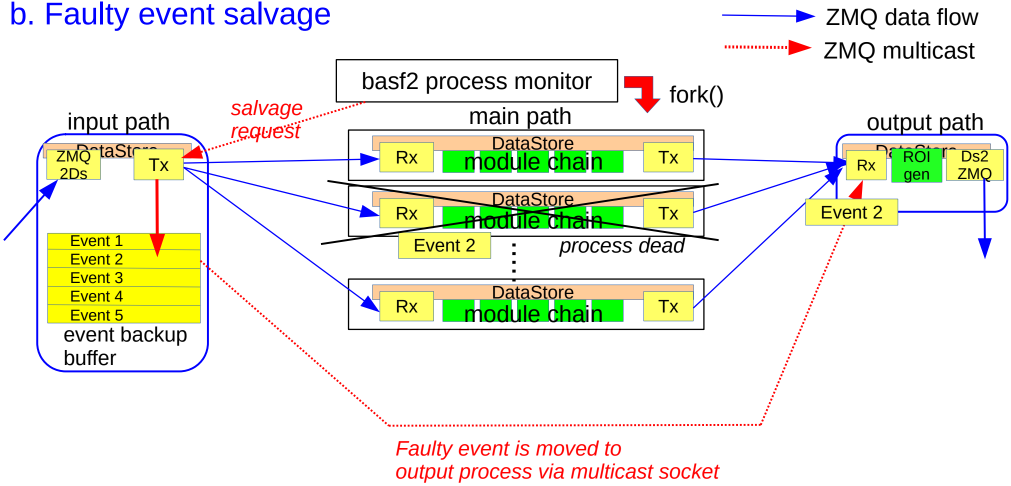
Faulty event salvage in the new HLT framework

b. Faulty event salvage



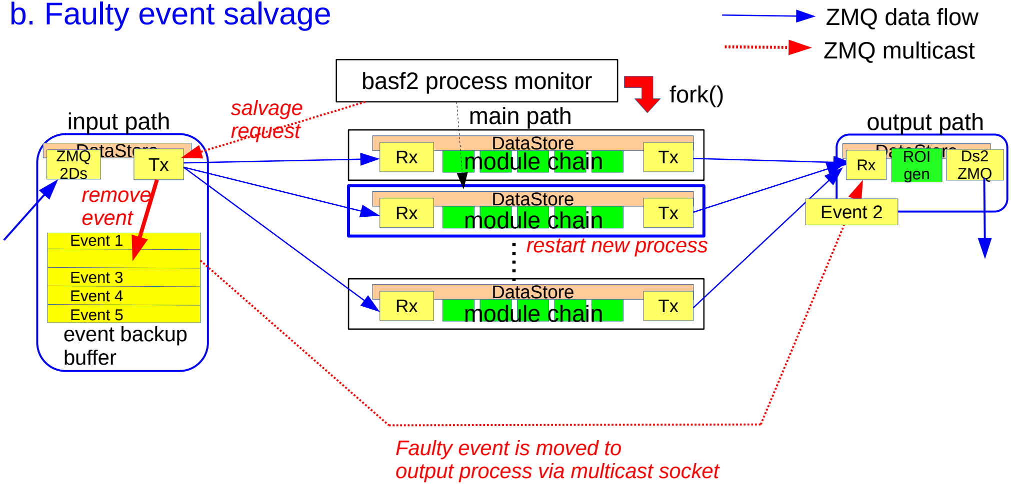
Faulty event salvage in the new HLT framework

b. Faulty event salvage

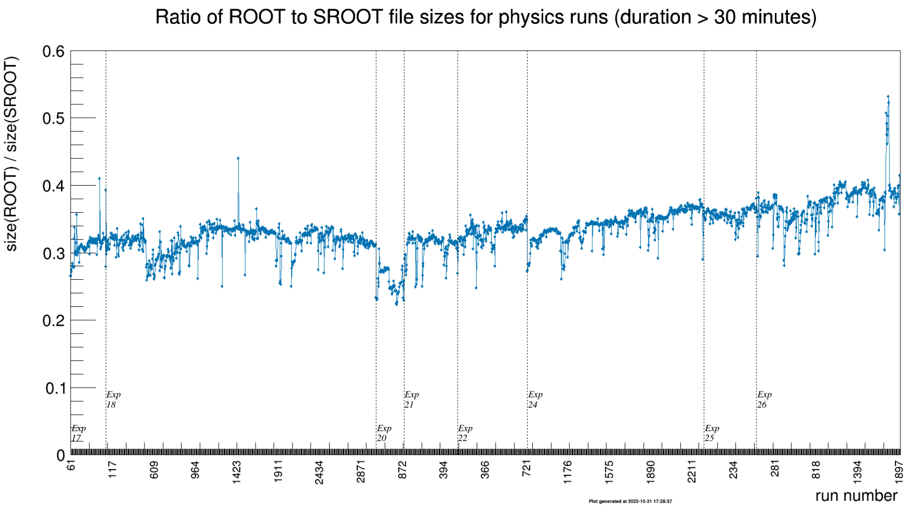


Faulty event salvage in the new HLT framework

b. Faulty event salvage



Size comparison: SROOT vs. ROOT



Write cache disks

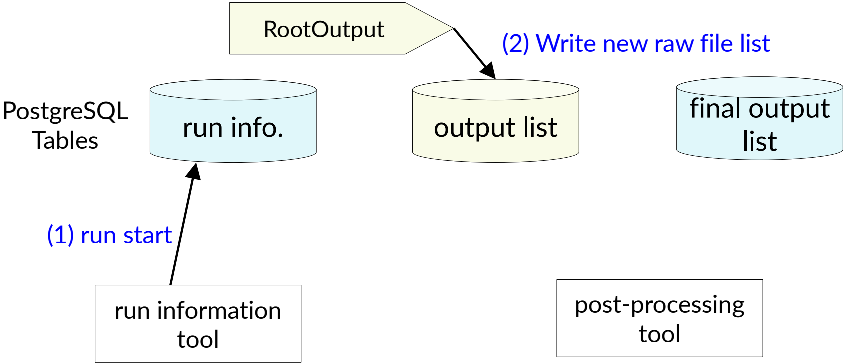
- **During the test, we faced some troubles on creating and closing ROOT files.**
  - It's because 8 files are trying to be created at the same time in an HDD array.
- **To solve the issue, write cache disks are installed.**
  - 2TB SATA SSD per RAID disk → 6TB buffer space
  - Once a file is correctly closed, the file immediately moved to the corresponding RAID disk.
    - Since the buffer disk is large enough, we can use it as temporary space in case of RAID disk issue.
  - The buffer space also prevents performance degradation of output writing which can be caused by reading files simultaneously
- **All the SSDs are hot swappable and monitored by zabbix smartmon.**

```
/dev/sdh1       1.9T  1.1G  1.9T   1% /buffer/rawdata/disk03
/dev/sdf1       1.9T  1.1G  1.9T   1% /buffer/rawdata/disk01
/dev/sdg1       1.9T  1.1G  1.9T   1% /buffer/rawdata/disk02
/dev/sda1        33T  2.5T   31T   8% /rawdata/disk02
/dev/sdb1        33T  2.7T   31T   9% /rawdata/disk01
/dev/sdc1        33T  2.6T   31T   8% /rawdata/disk03
```

Post-processing and database-based file listing

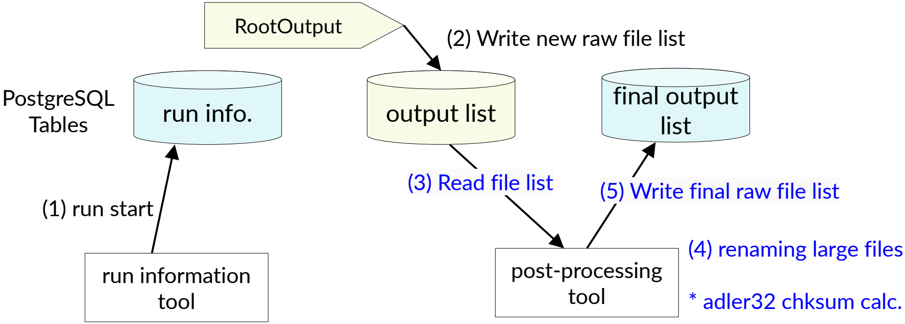
- **After the new output files are placed in the RAID disks, further processing is necessary.**
  - Renaming large enough files
  - Merging small files
  - Checksum calculation
  - Making the final file list to be sent
- **For the file listing, three PostgreSQL tables are used.**
  - run info table: recording run information, exp/run number, run type, global flags, ...
  - output list table: file list before the post processing, recorded by the RootOutput module
  - final output list table: file list after the post processing, used for the online-offline file transfer

Post-processing and database-based file listing



- **Beginning of the run**
  1. run info table: New run is recorded
  2. output list table: New files are listed by RootOutput modules

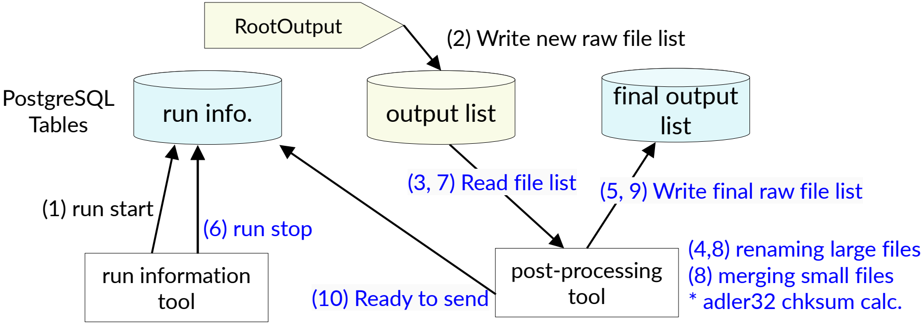
Post-processing and database-based file listing



■ Middle of the run

1. output list table: once file is reached at the size limit, close the files and update "closed" flag
2. Once file closing is confirmed, move files from buffer disk to RAID disk
3. final output list table: rename, calculate checksum, and update the entry

Post-processing and database-based file listing

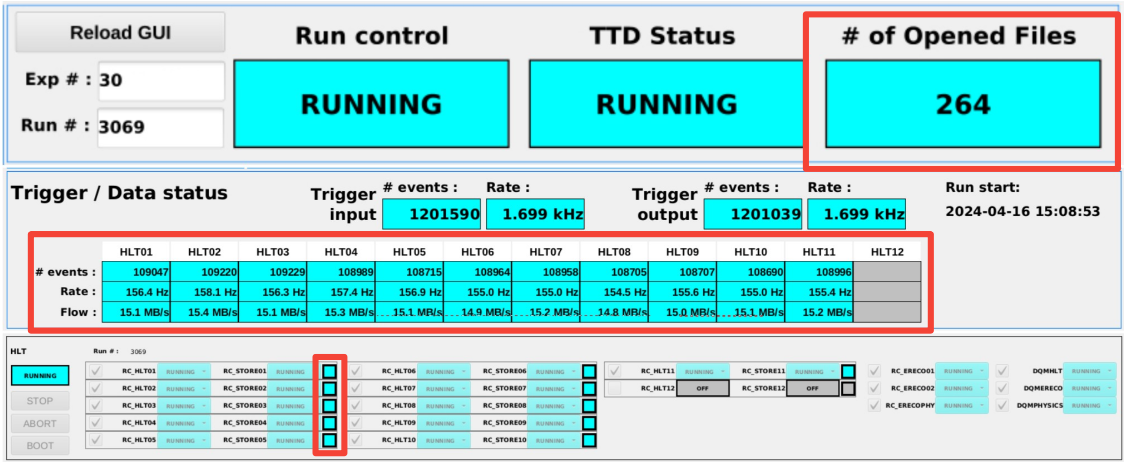


■ After the run end

1. run info table: flag the run end
2. output list table: close the files and update a "closed" flag
3. final output list table: rename or merge files, calculate checksum, and update the entry
4. Once everything is ready, set "ready to send" flag

Monitoring

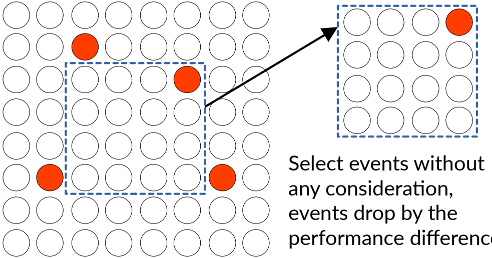
- Run control GUI provides useful information and CR shift can check the STORE status
  - The color becomes red or orange if the state is wrong



HLT result based selection for ERECO

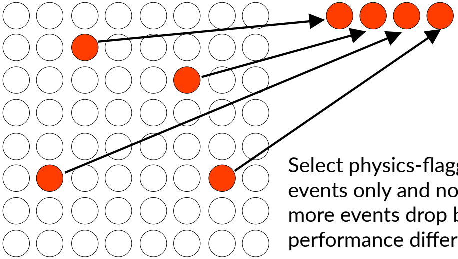
- **# of ERECO is smaller than HLT, therefore only a part of events can be processed.**
- **The less performance ERECO occurs random event selection caused by event drops.**
- **We want more statistics of physics features while keeping the random sampling.**
  - The random sampling is also important, especially for the pixel detector, since the pixel detector information is not in HLT.

< Random sampling >



Select events without any consideration, events drop by the performance difference

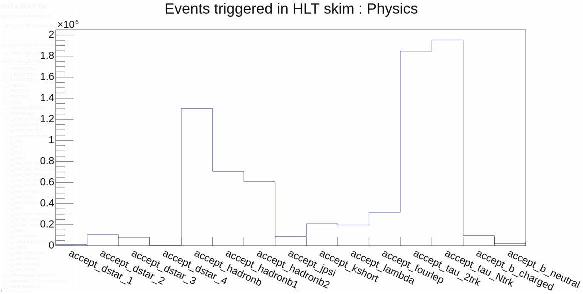
< Physics sampling >



Select physics-flagged events only and no more events drop by the performance difference

HLT result based selection for ERECO

- **# of ERECO is smaller than HLT, therefore only a part of events can be processed.**
- **The less performance ERECO occurs random event selection caused by event drops.**
- **We want more statistics of physics features while keeping the random sampling.**
  - The random sampling is also important, especially for the pixel detector, since the pixel detector information is not in HLT.



The number of events for each physics skims from 4.7M events.

accept_dstar_1 trigger rate