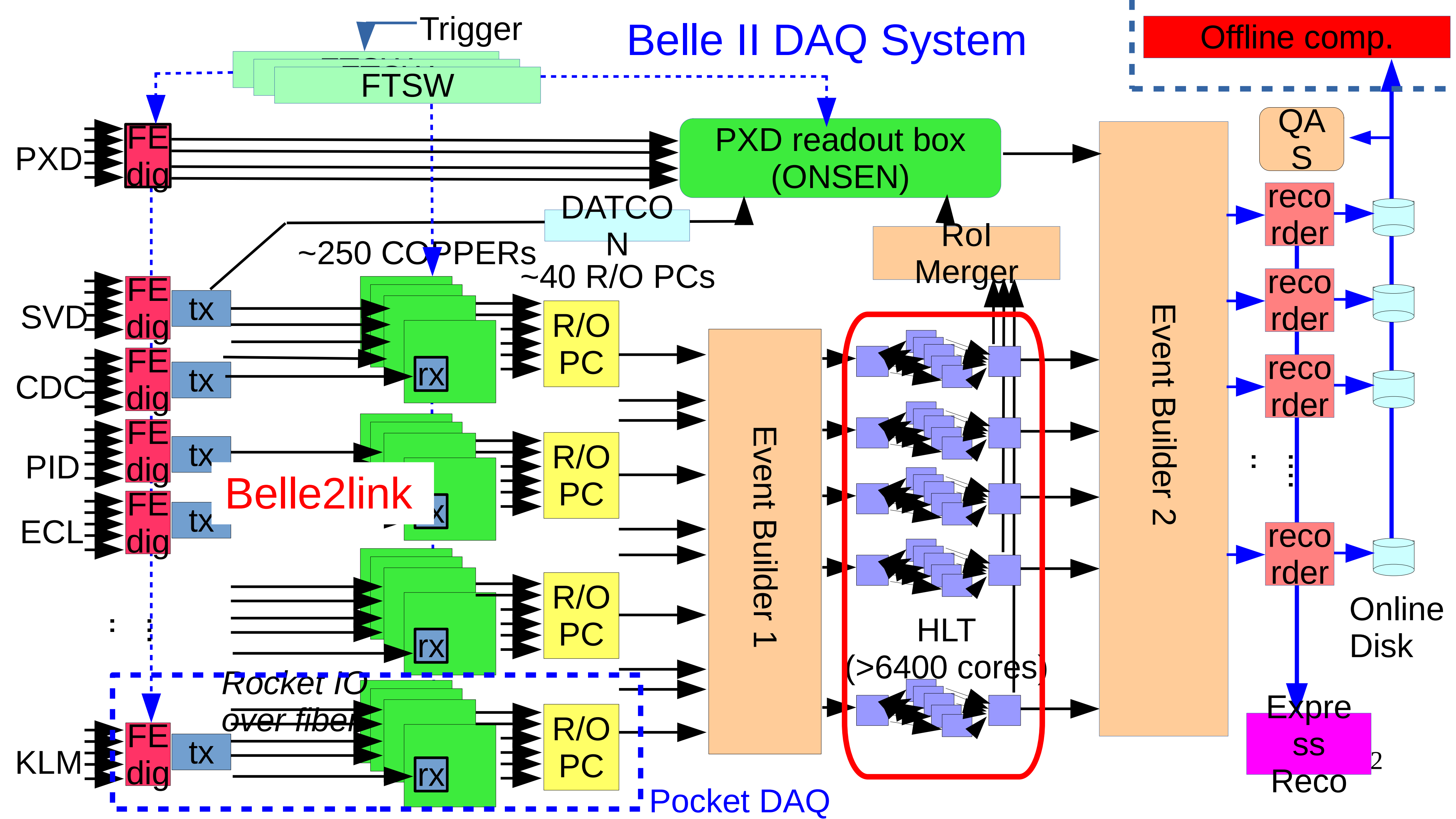
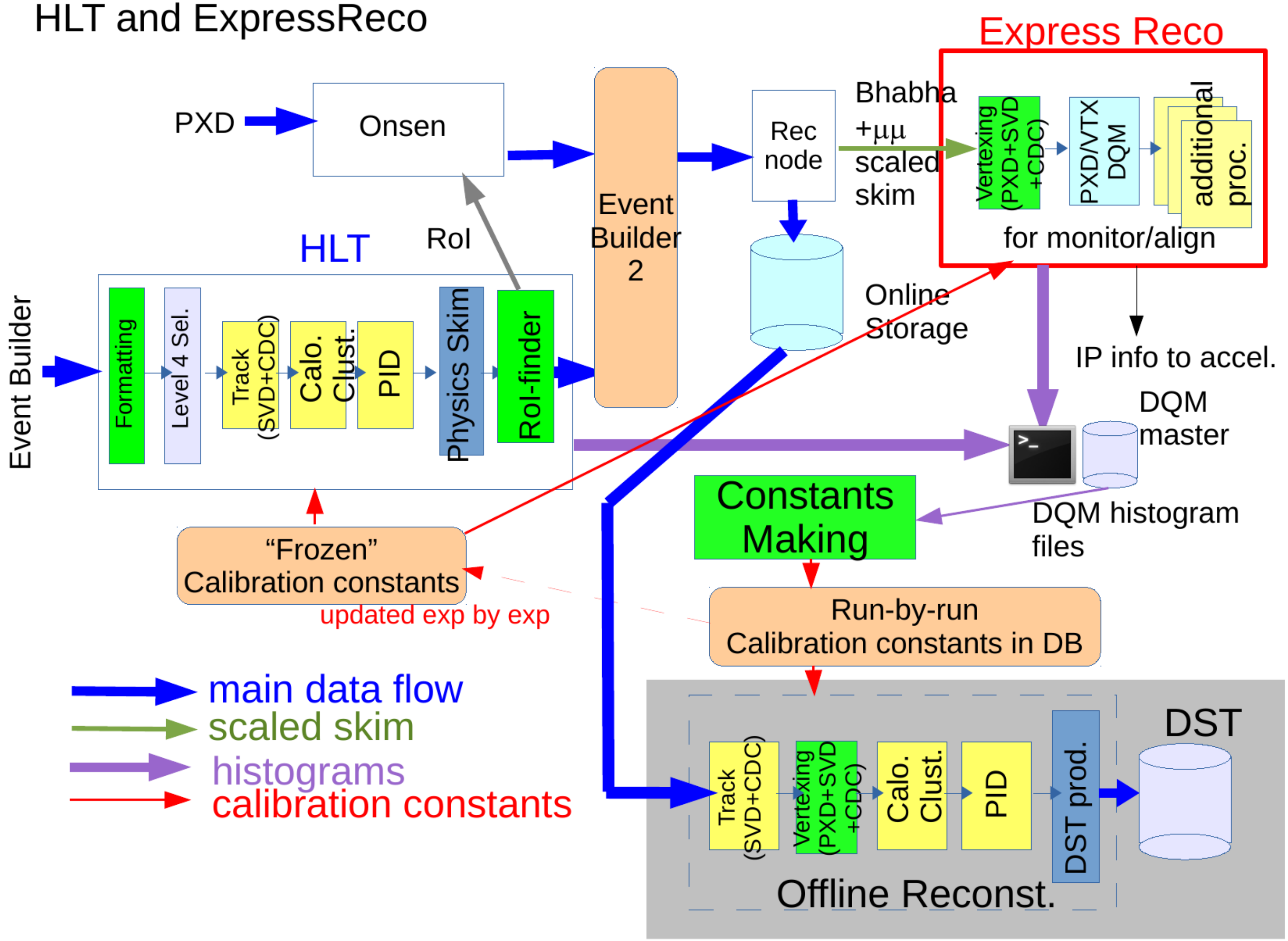
Operation History of Belle II HLT and New Framework

Ryosuke Itoh
IPNS, KEK

Belle II DAQ System



HLT and ExpressReco



1. Requirements to Belle II HLT

Functions of Belle II HLT

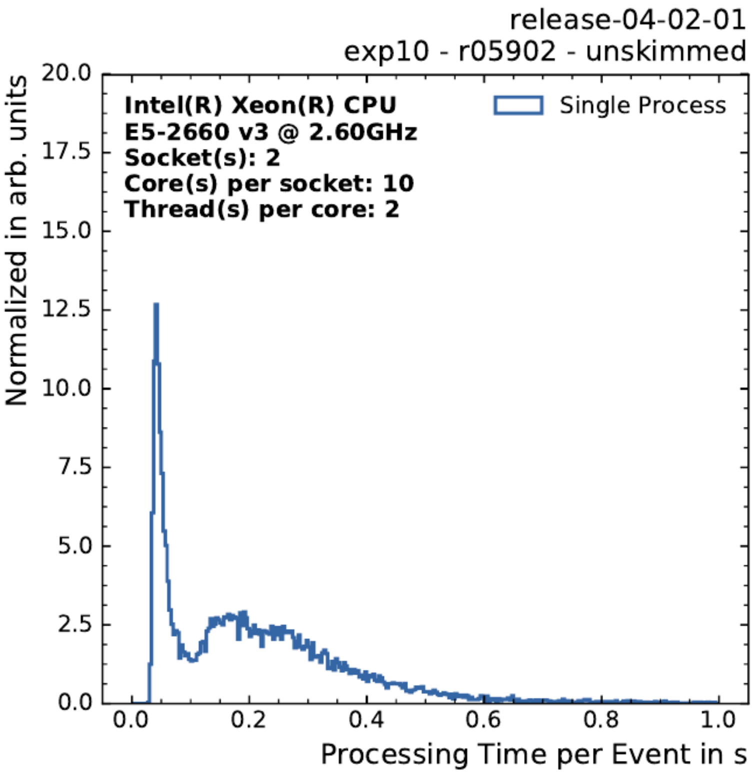
0. Event data transport to storage (except PXD) at the rate of up to 30kHz. Event size is around 100kB.
   -> >3GB/sec throughput must be ensured.

1. Discarding background events using full event reconstruction results.

2. Real time monitoring of data quality including physics level monitoring

3. RoI generation and transport to PXD readout

- The processing for 1 to 3 is based on the full event reconstruction.
  * The same offline software is assumed to be used.
  * The processing time / event is critical.



- Ave. time is > 0.3 sec./event/core.

- To manage 30kHz, one event must be processed in 0.000033 sec.

  => Needs large scale parallel processing with a granularity of O(10000).

Design of HLT based on Belle's RFARM experience

- The required number of cores in HLT is estimated based on the experience of RFARM in Belle where a full event reconstruction was performed using the same offline software for all the events in real time.

- The estimated number of cores required at t=0 (L=2x10^{35}; 1/4 of full L) is 1400. One HLT unit is equipped with 320 and we will prepare 5 units for t=0, 1600 cores.

- We will gradually add HLT units to keep up with the luminosity improvement. -> 6400 cores in total for the full luminosity.

- The requirement to the average processing time per core per event is about 0.33 sec(3Hz/core). It is the average time for all event types.

- Belle's experience shows the processing time for a hadronic event is around 1 sec. while 1/10 for other types of events.
  - -> Average processing time was less than 0.3 sec / core considering the cross section and L1 trigger selection.

2. Hardware of Belle II HLT

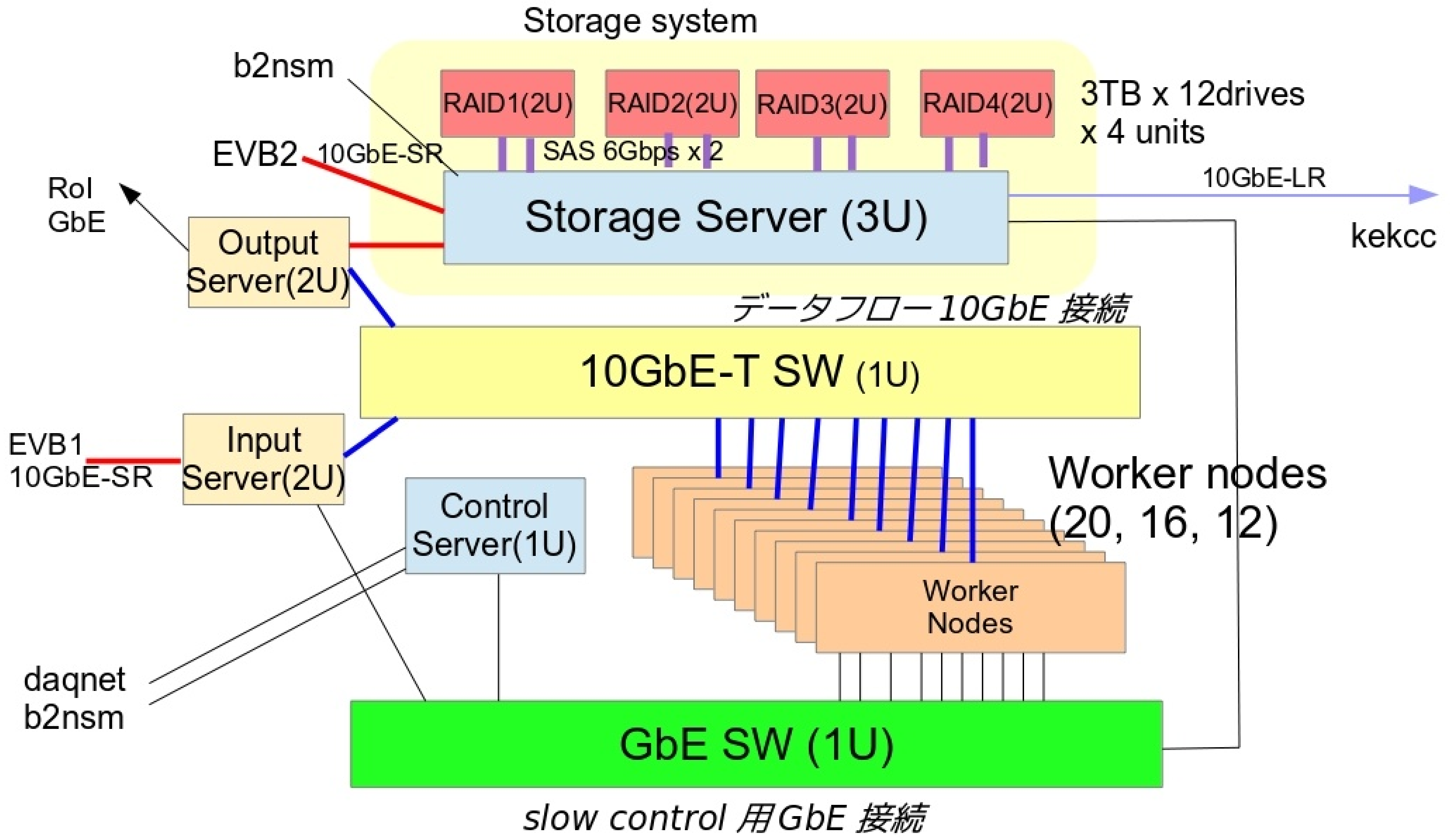
- Unit structure: Coarse parallelism is implemented by the unit structure. Event builder distributes the events to each unit following the modulo of event number.

- One unit consists of an input server, an output server, and up to 20 worker nodes with a control server.

- Each worker is equipped with muticore CPU(s) providing 16-40 physical cores / server.

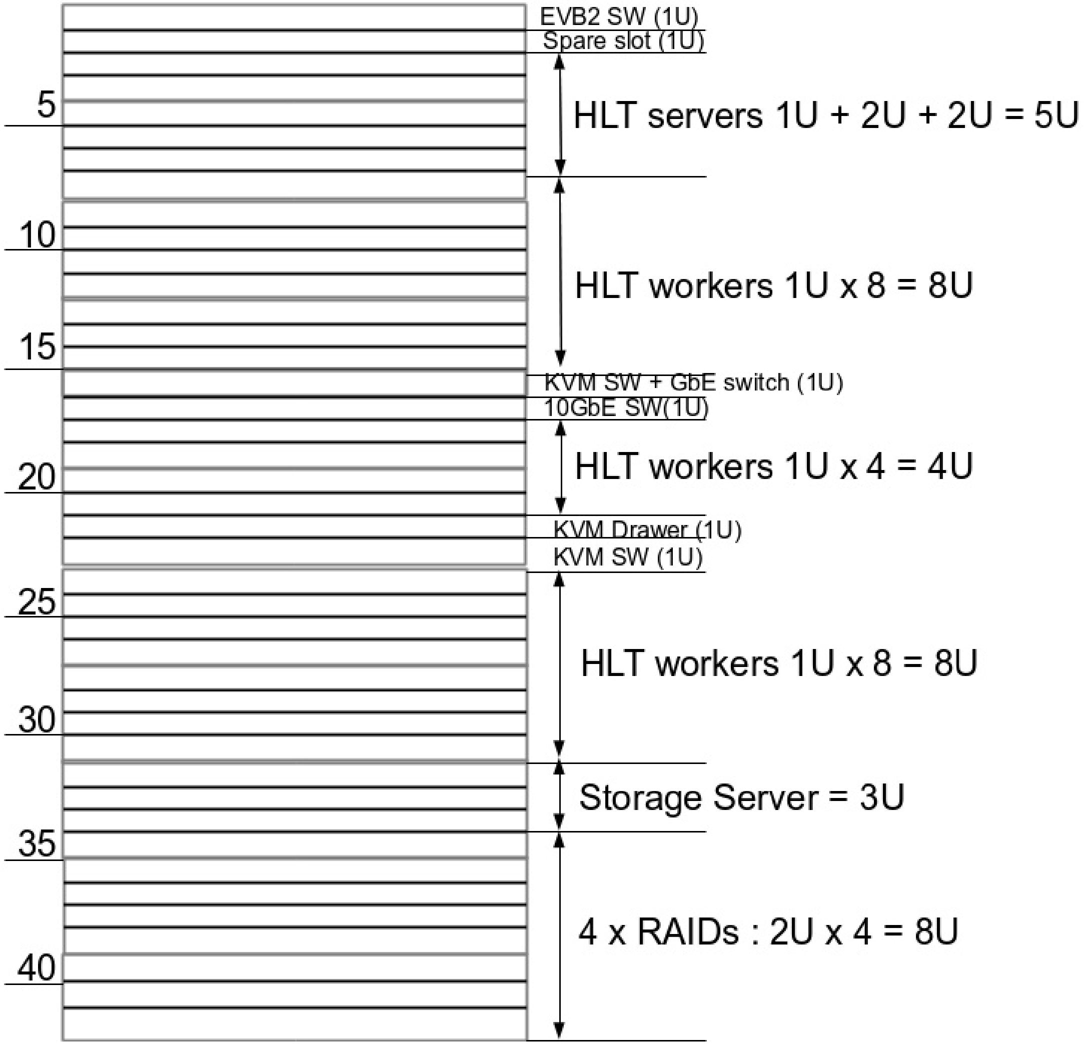
- Data flow nodes are connected via 10GbE network while all nodes are via GbE network for the system control.

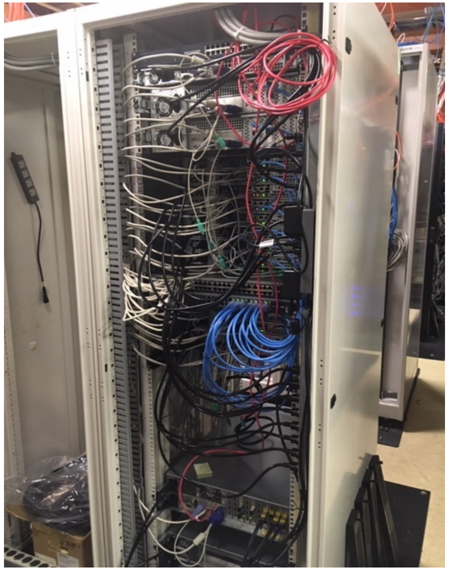
Actual Implementation of HLT/Storage unit

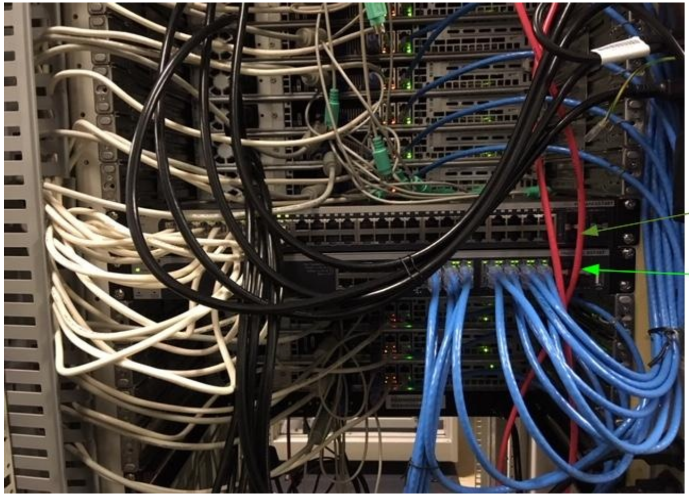


Equipped in a single rack

Layout of a rack for HLT/STORE unit







See following link for the detail of the construction
https://confluence.desy.de/display/BI/Hardware+preparation+of+an+HLT+unit

Number of physical cores after 2021 HLT Reinforcement

HLT01:
16 cores * 9 + 20 * (2+2) + 28 * 2 + 36 * 2 + 40 * 3 = 472 cores
(replaced 11 of 16 core servers with new ones).
HLT02-05
20 cores * 16 + 36 cores * 2 + 40 cores * 2 = 472 cores
HLT06-09
28 cores * 12 + 36 cores * 2 + 40 cores * 2 = 488 cores
HLT10
28 cores * 12 + 36 cores * 2 + 40 cores * 2 = 488 cores

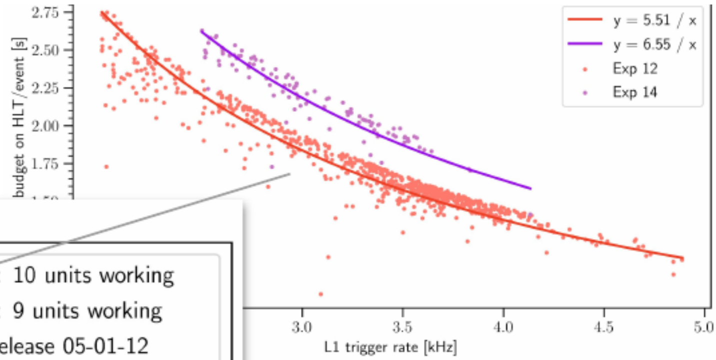
4800 cores

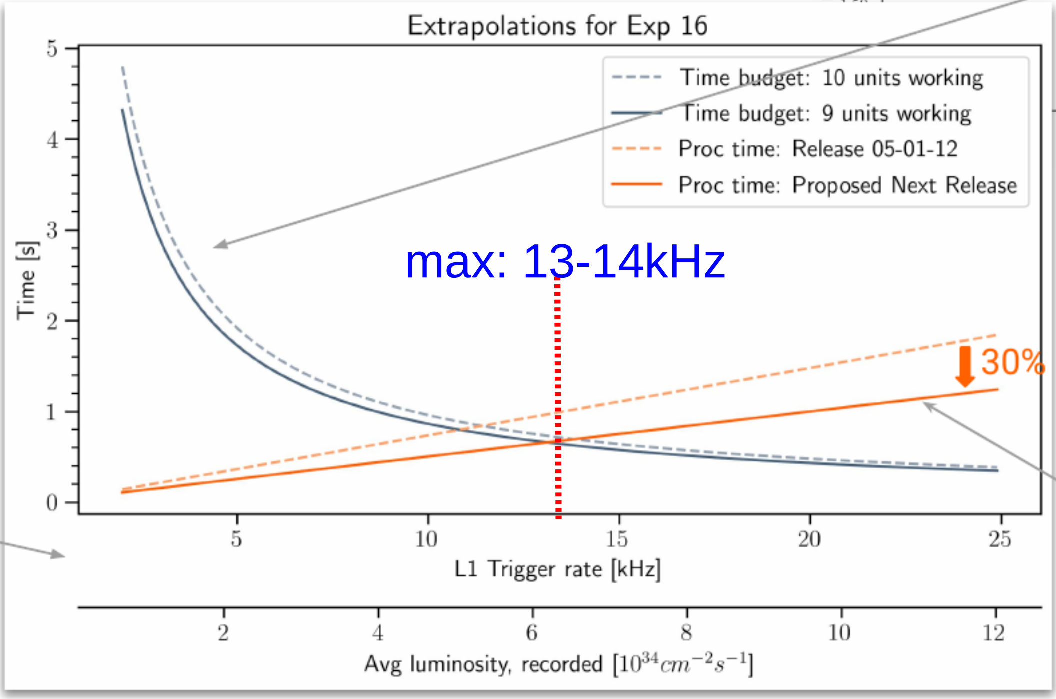
- Last reinforcement achieved 75% of the design number of cores(6400).
- At the same time, the operating system has been upgraded to CentOS7.

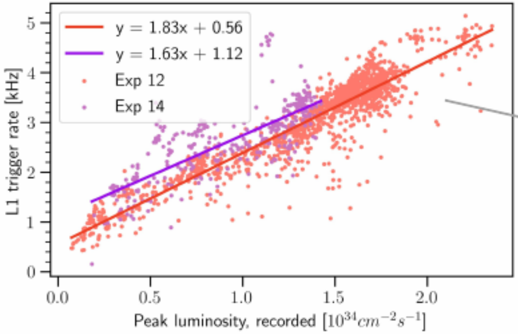
V.S.Vobbilisetti (IJCLab)

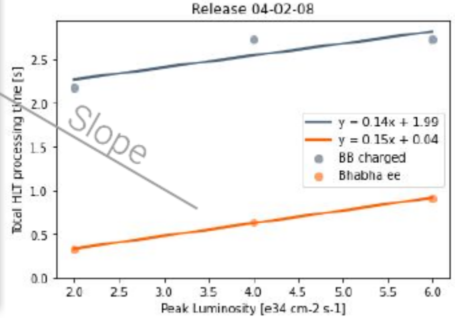
Estimates for 2021 a/b

- 2020 winter upgrade is included
- 9 out of 10 units working
- # processes is full, i.e., no out of memory problem
- No hardware under-performance (10% observed in Exp 14)









If the L1 trigger menu stays the same.

Should be safe for 2021 a/b

Generated samples at different luminosity (background conditions) to extrapolate processing time growth

Addition of 3 HLT/STORE units in LS1

- 18 new HLT workers + 2 sets of HLT control/STORE units are already in hand.

- 15 (or more) servers + 1 more set of HLT control and STORE units will be purchased by the end of FY2022.

=> 1.5 units will be built in autumn (Oct.-)
1.5 unit will be added in Jan.-Mar (2023)
-> In total : 13 HLT units; 6400 cores.

(one of them = HLT13 will be used as a test bench as before until it is really needed to process high rate)

Expected performance : up 20 kHz from 2023c run!

Note: HLT operation during summer was limited (up to 5 units) to save the power consumption.

3. HLT processing framework

HLT framework consists of four subframeworks

1. Data flow framework for parallel processing
   * Consistent extension of basf2 parallel processing
     utilizing the same RingBuffer
   * Socket interface from/to RingBuffer performs actual data flow
     between nodes.

2. Control framework to synchronize the operation of servers.
   * Own control framework based on native NSM2
     (independent of daq_slc)
   * External interface to daq_slc

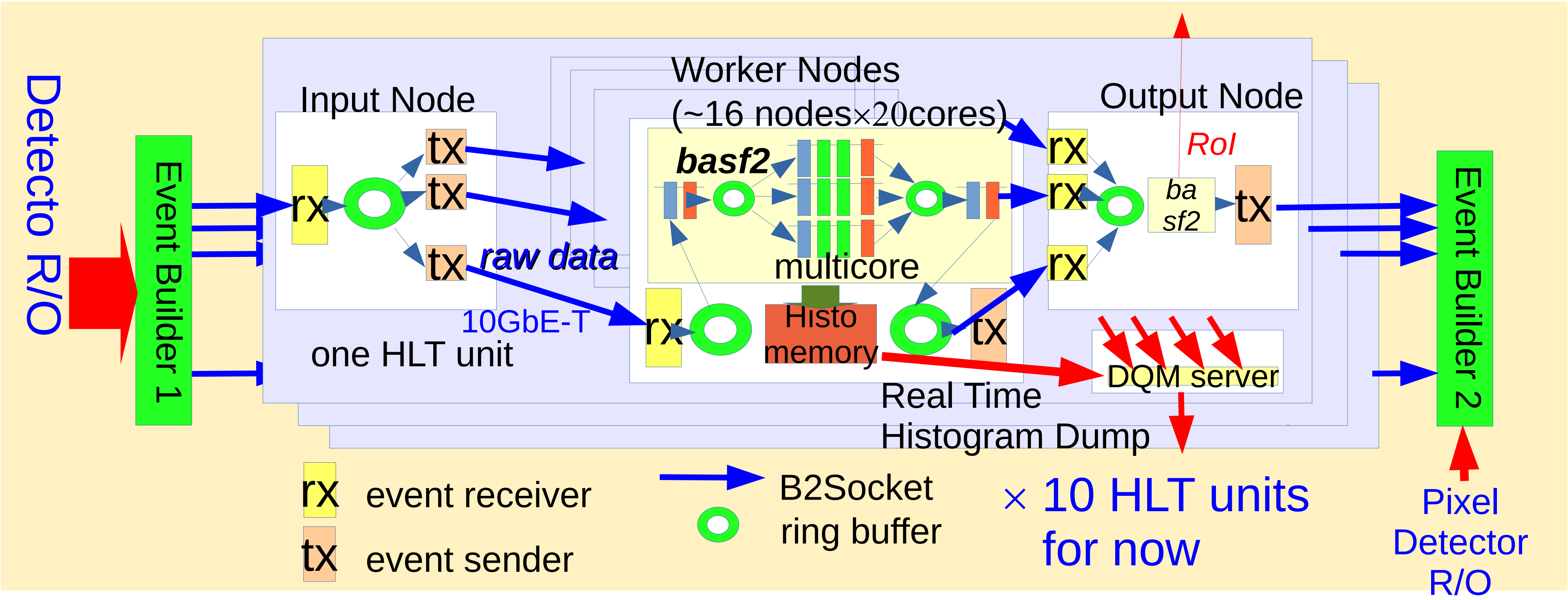
3. Live histogram collection

* Framework to “spy” and “collect” lively accumulated histograms.
* It includes the transport of histograms to other node over socket connection.

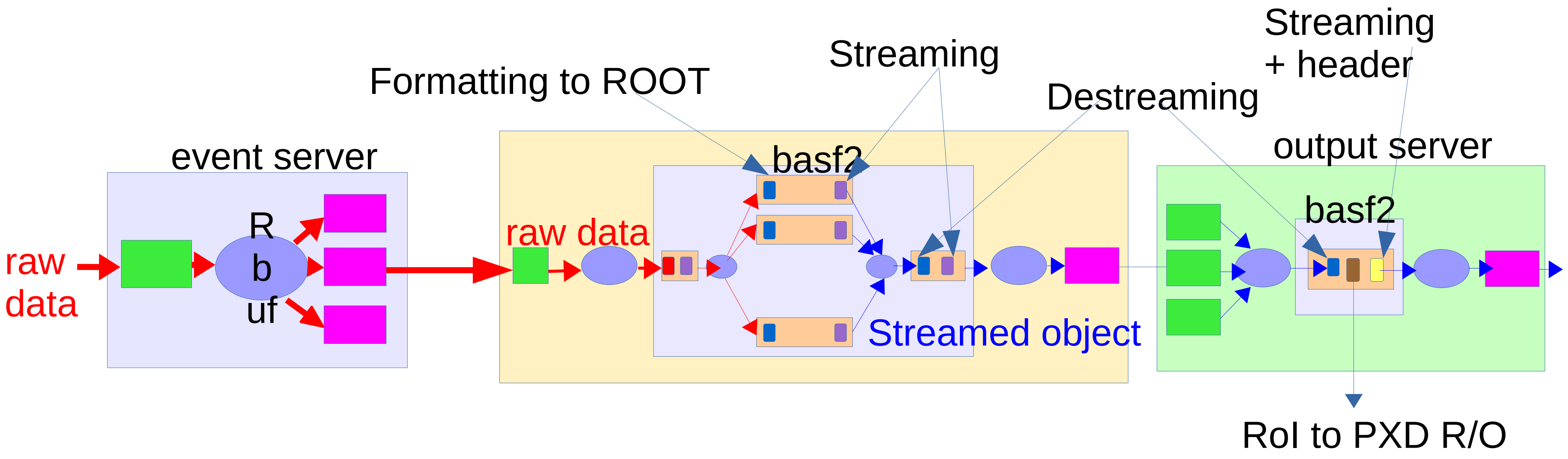
4. RoI transport

* A mechanism to extract RoI from HLT processed results and send them to PXD readout.

1. Data flow framework (Original framework : RFARM)

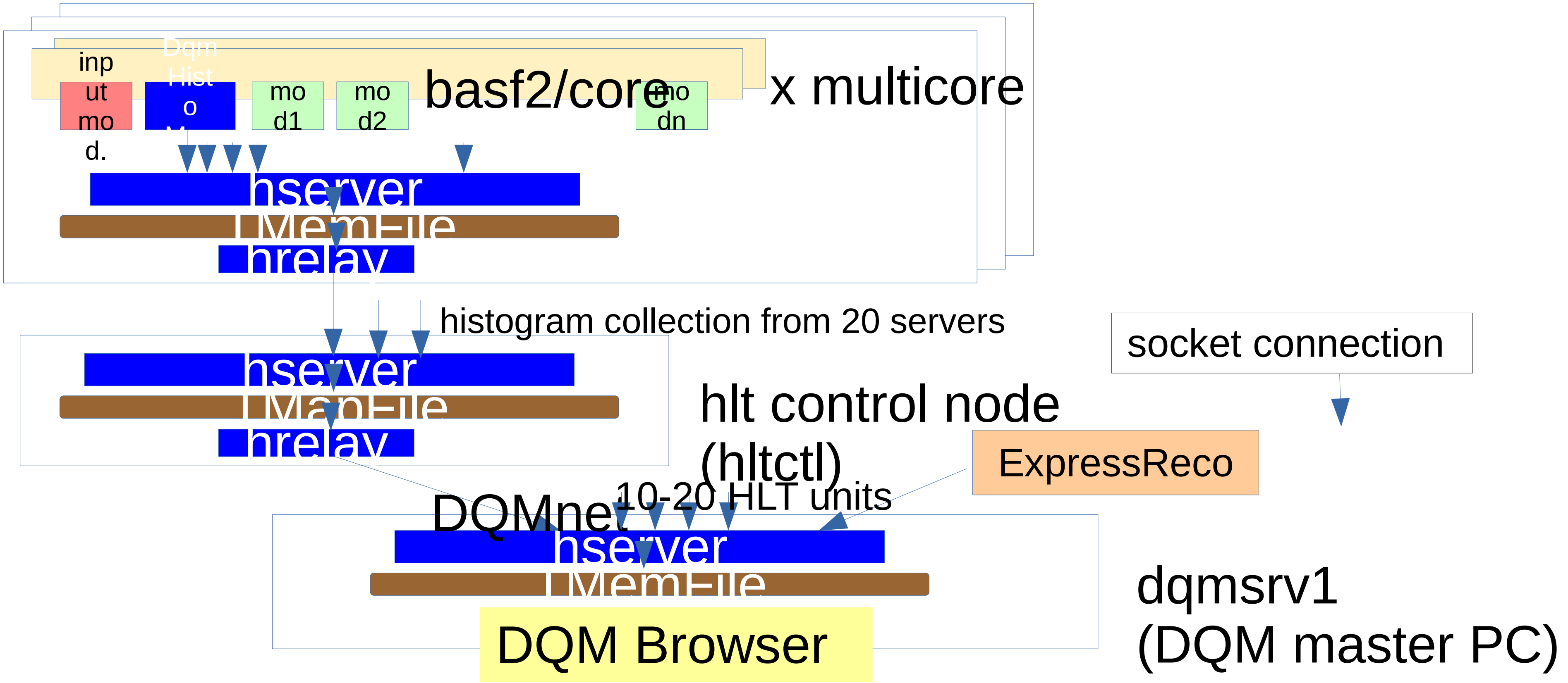


Raw data flow on HLT

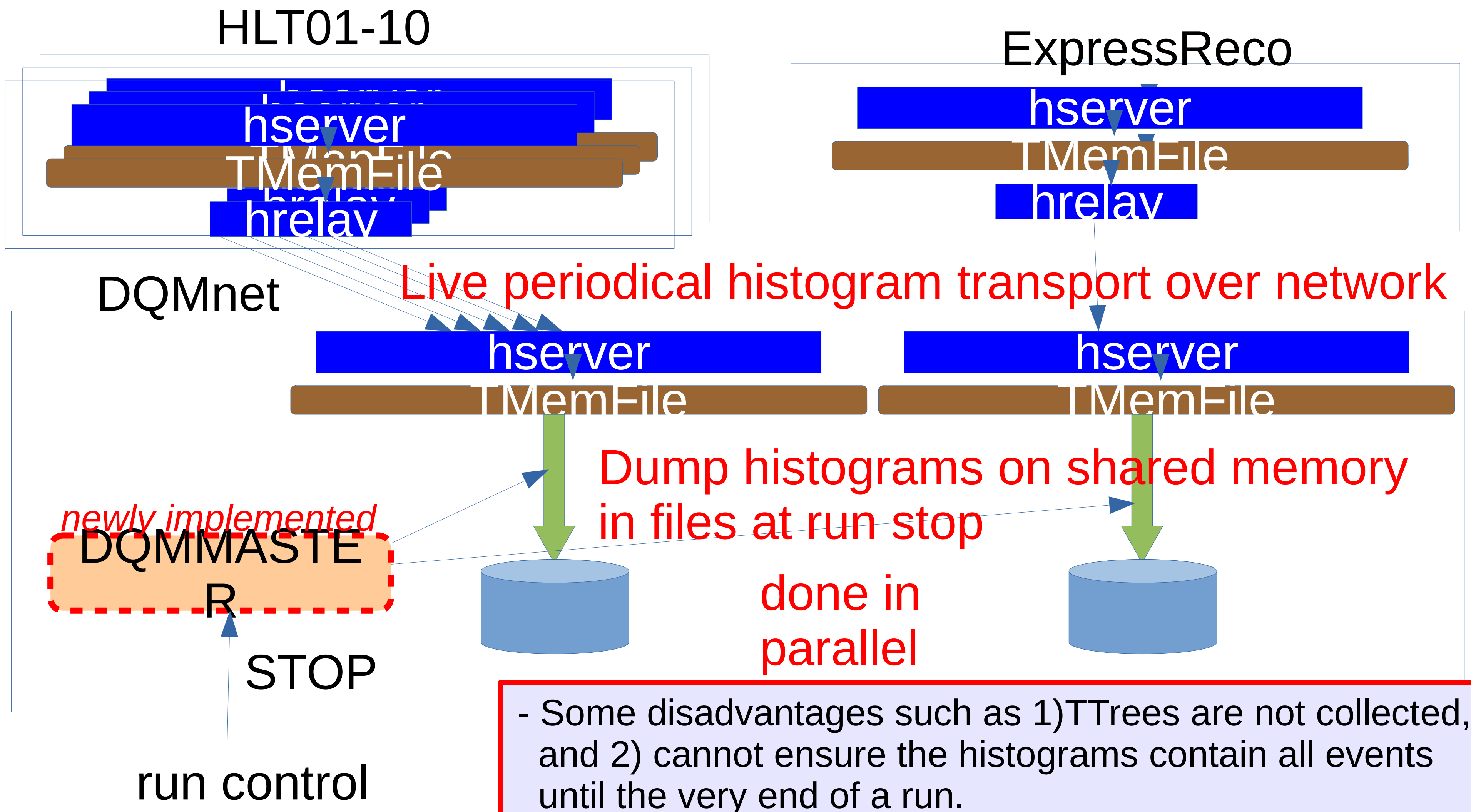


Histogram Collection

- Implemented by the repeated use of hserver and hrelay.



Real Time Histogram Transport and New Histogram Store



- Some disadvantages such as 1)TTrees are not collected, and 2) cannot ensure the histograms contain all events until the very end of a run.

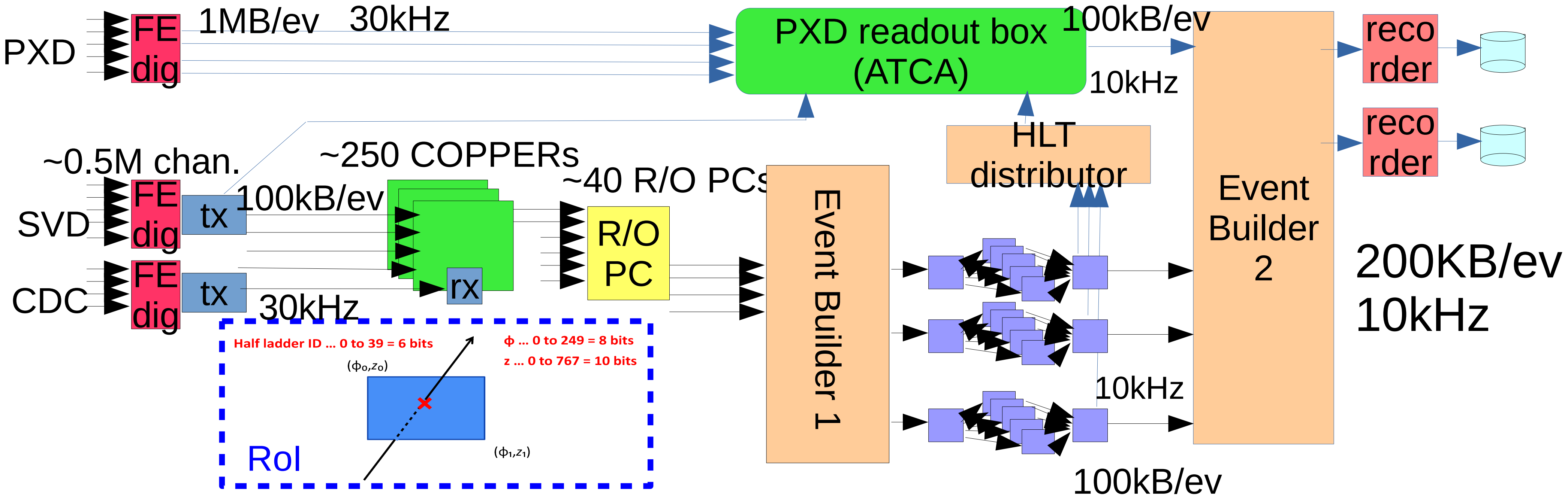
RoI transport

- RoIs have to be sent to PXD readout (ONSEN) for the data reduction.

- RoIs are calculated by the tracking results in the HLT processing and placed in DataStore as one of the raw data object.

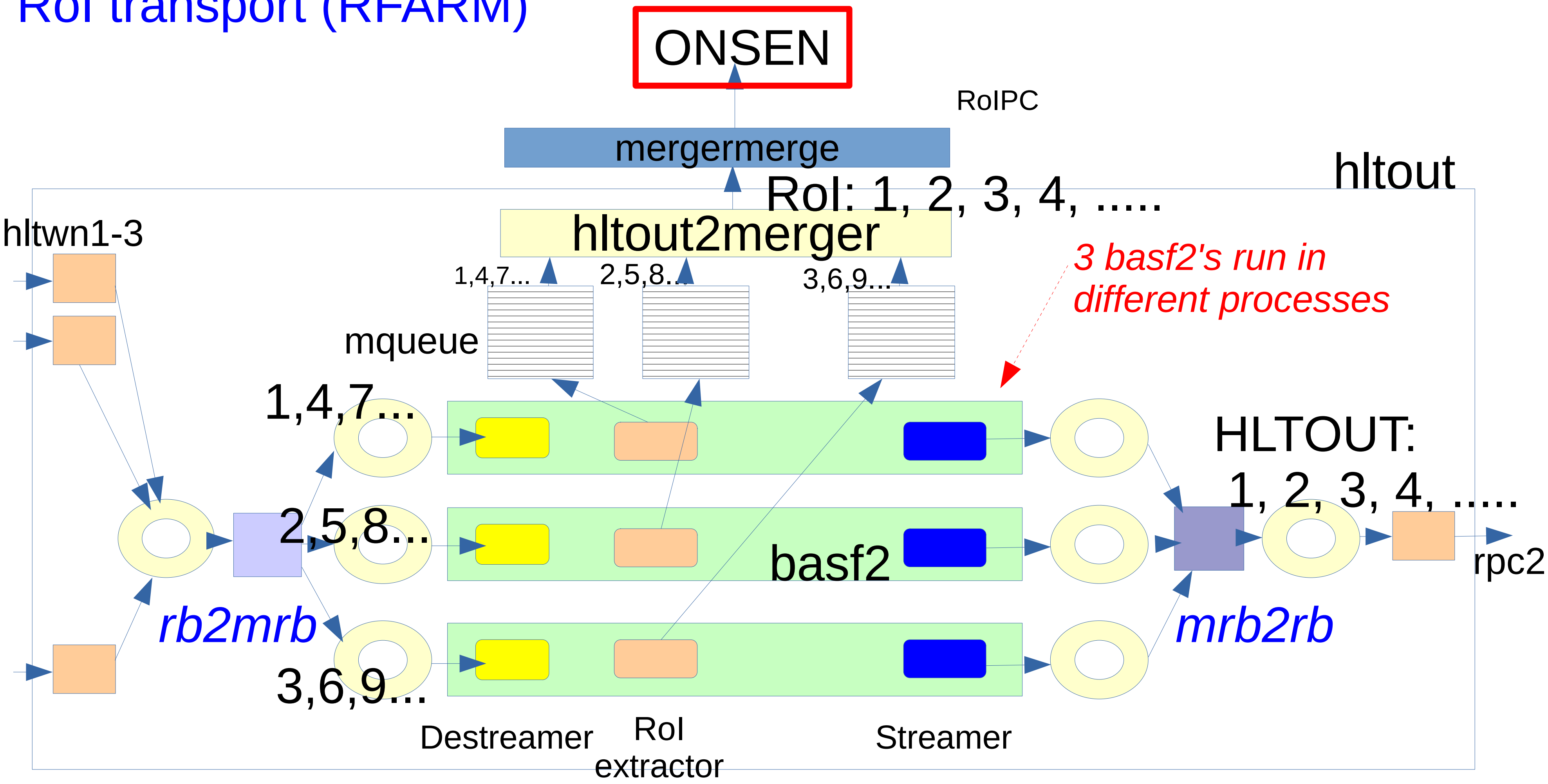
- They are collected in the output server of HLT and RoIs are extracted from the streamed object.

- The extracted binary are sent to ONSEN through the network connection.



- PXD yields a large event sized data when occupancy is high (>1MB) and it cannot be processed by COPPERs, nor recorded without event reduction.
- Data size reduction by 1) extrapolate HLT-reconstructed tracks to the surface of PXD sensors (Region of Interest), 2) send the RoIs to PXD readout box, and 3) discard hits not in RoIs. -> 1/10 reduction is expected.
- RoIs are sent only for HLT-selected events, and the rate reduction is also applied.

RoI transport (RFARM)



* rb2mrb, mrb2rb, and hltout2merger distribute/pick up records in turn to/from ringbuffers/mqueues in the same order.

What was the problem in RFARM framework?

- Heavy dependence on RingBuffer + raw socket I/O.
  * RingBuffer is a home made tool utilizing old-fashioned UNIX IPC:
    Shared Memory and Semaphore.
  * The handling of IPCs is somewhat messy.
    + They remain even after the job exit.
    + Removal of IPC resource sometimes fails in the signal handling.
    + Unexpected IPC locking.

  => *caused frequent operation stuck of HLT when stopping/aborting.*

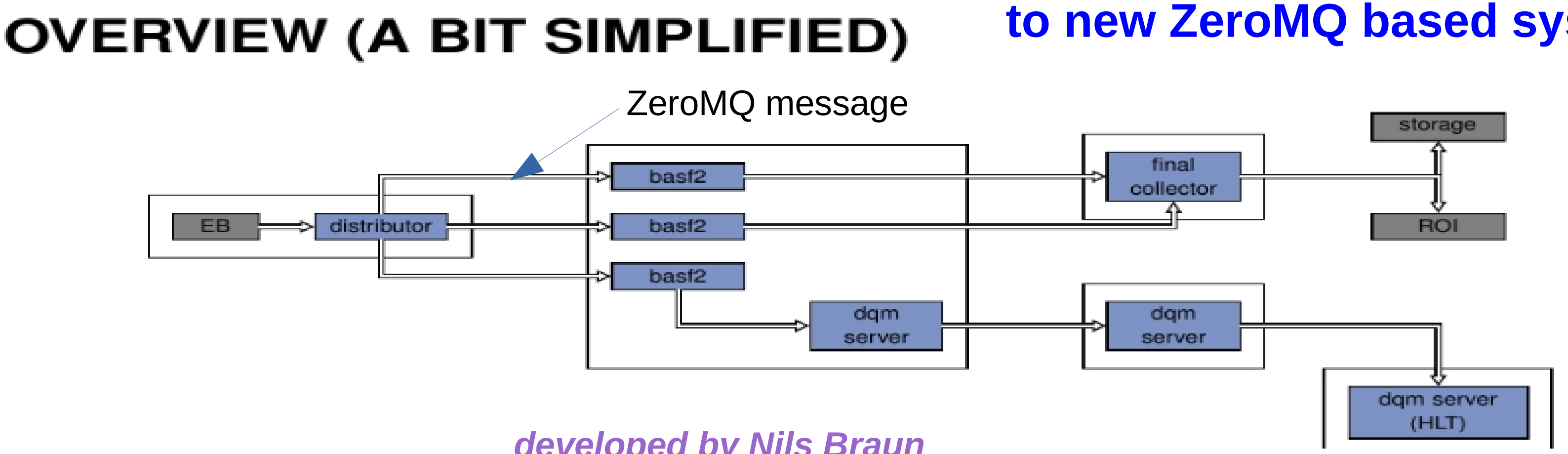
- Slow control was also home made and needs the interface to daq_slc.

- RoI extraction from streamed objects was complicated and slow.

- Initialization was done after the receipt of the first event.
  => ~30sec delay to start actual processing.

ZeroMQ HLT (2020-)

Framework was switched to new ZeroMQ based system



developed by Nils Braun

- RingBuffers are replaced with ZeroMQ message transport.

- Initialization of processing is done when making the system ready (not at the time of receipt of the first event) by using modified version of basf2.

- RoI binary is embedded in ZeroMQ message as a separate packet.

- System control is integrated in the Belle II standard slow control package (daq_slc).

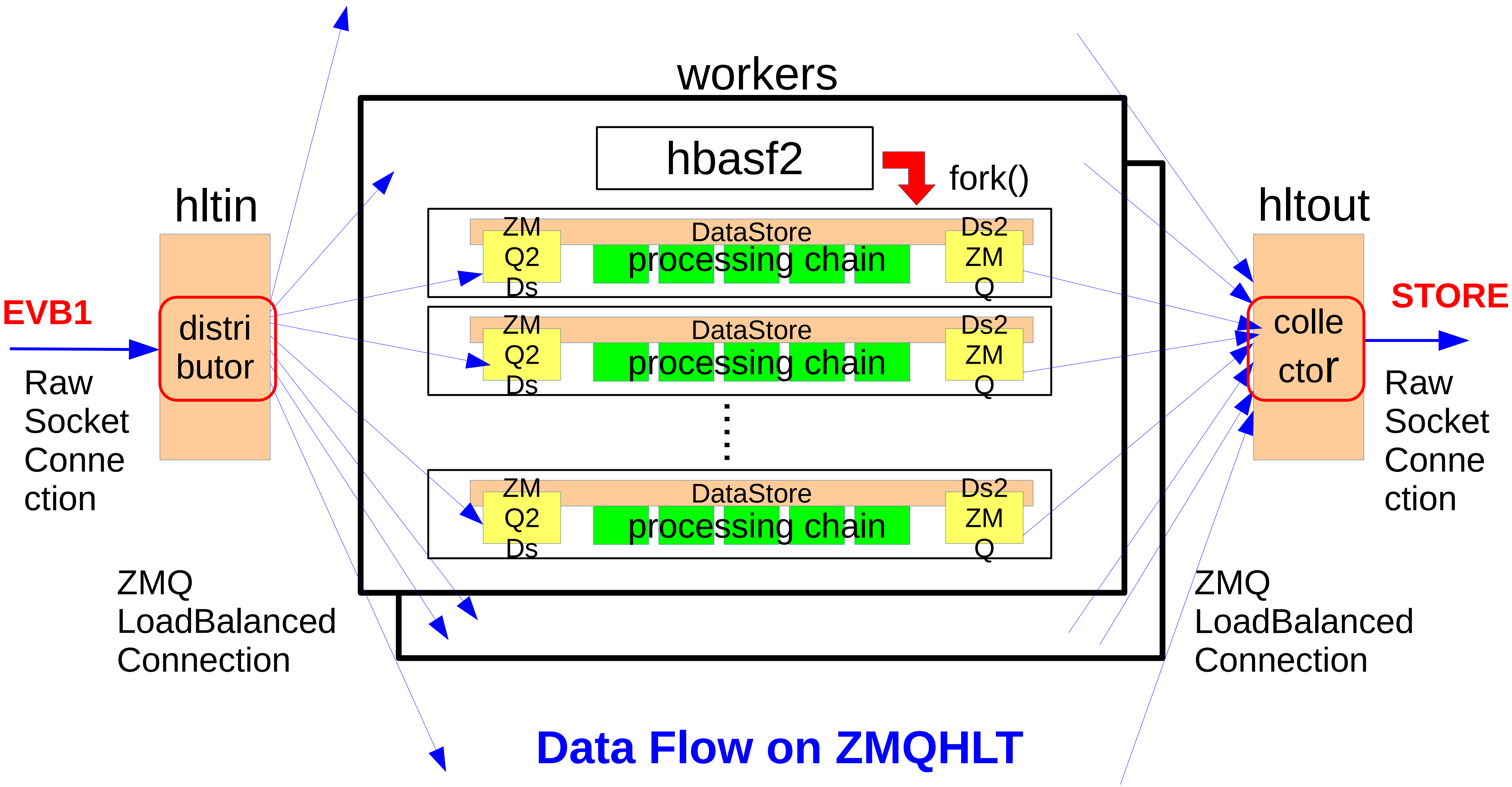
ZeroMQ

- An open-source package for the general message passing.
  -> Strong community support. Standard in HEP community.

- The usage resembles to that of the standard UNIX socket, but it has various functions.

- It supports “lock-free” 1-to-N and N-to-1 connection with a variety of connection style including load-balanced pipeline.

https://zeromq.org/



Data Flow on ZMQHLT

hbasf2

- hbasf2 is yet another implementation of basf2 specialized for the use in HLT. Directly called from a python3 script.

- Main difference from basf2 is that it receives the event from hltin and sends output to hltout directly by each event process via load-balanced ZMQ connection w/o mediating input/output processs.

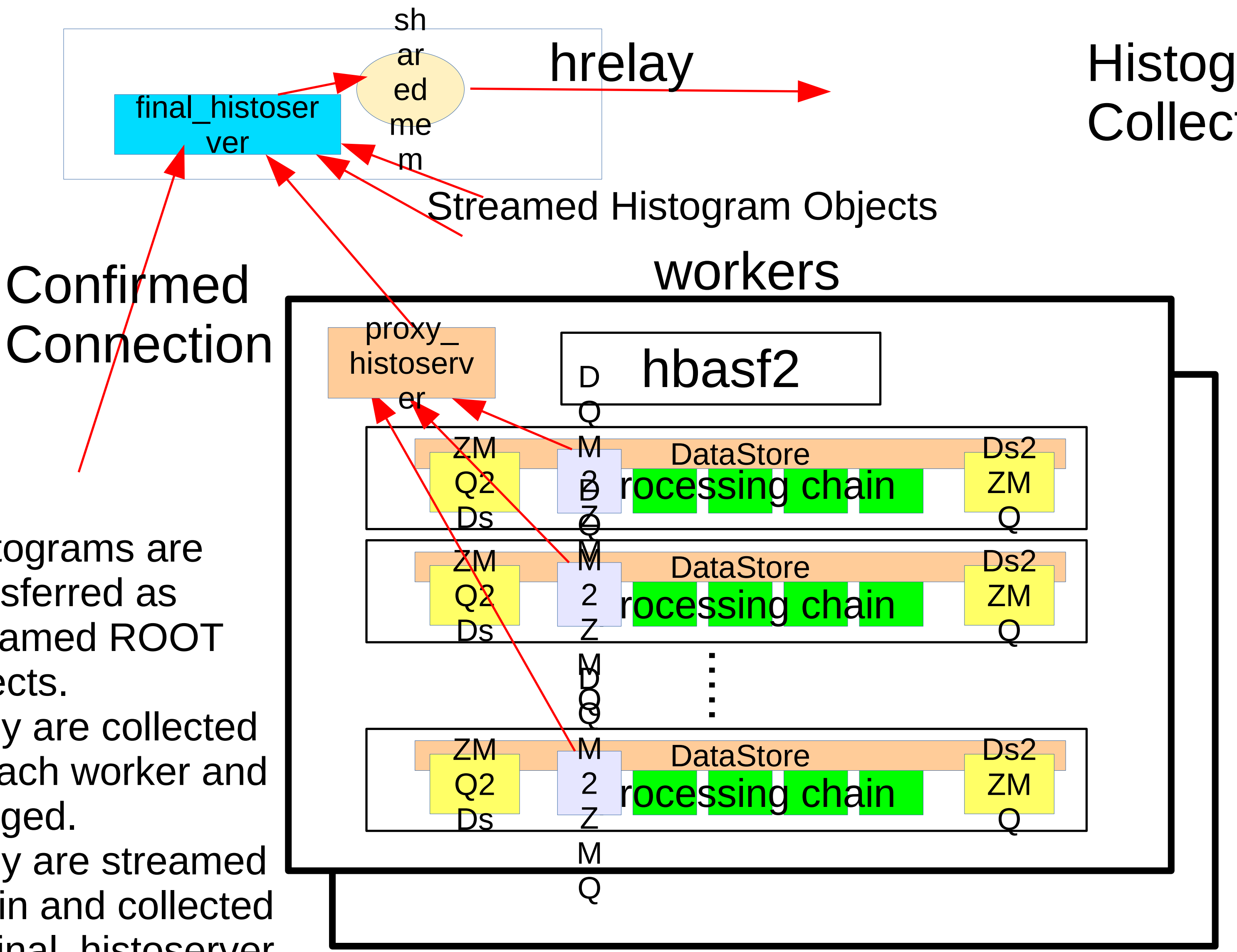
  -> Data flow is much simplified.

- When starting hbasf2, before forking out event processes, it performs all module initialization by sending a dummy event.

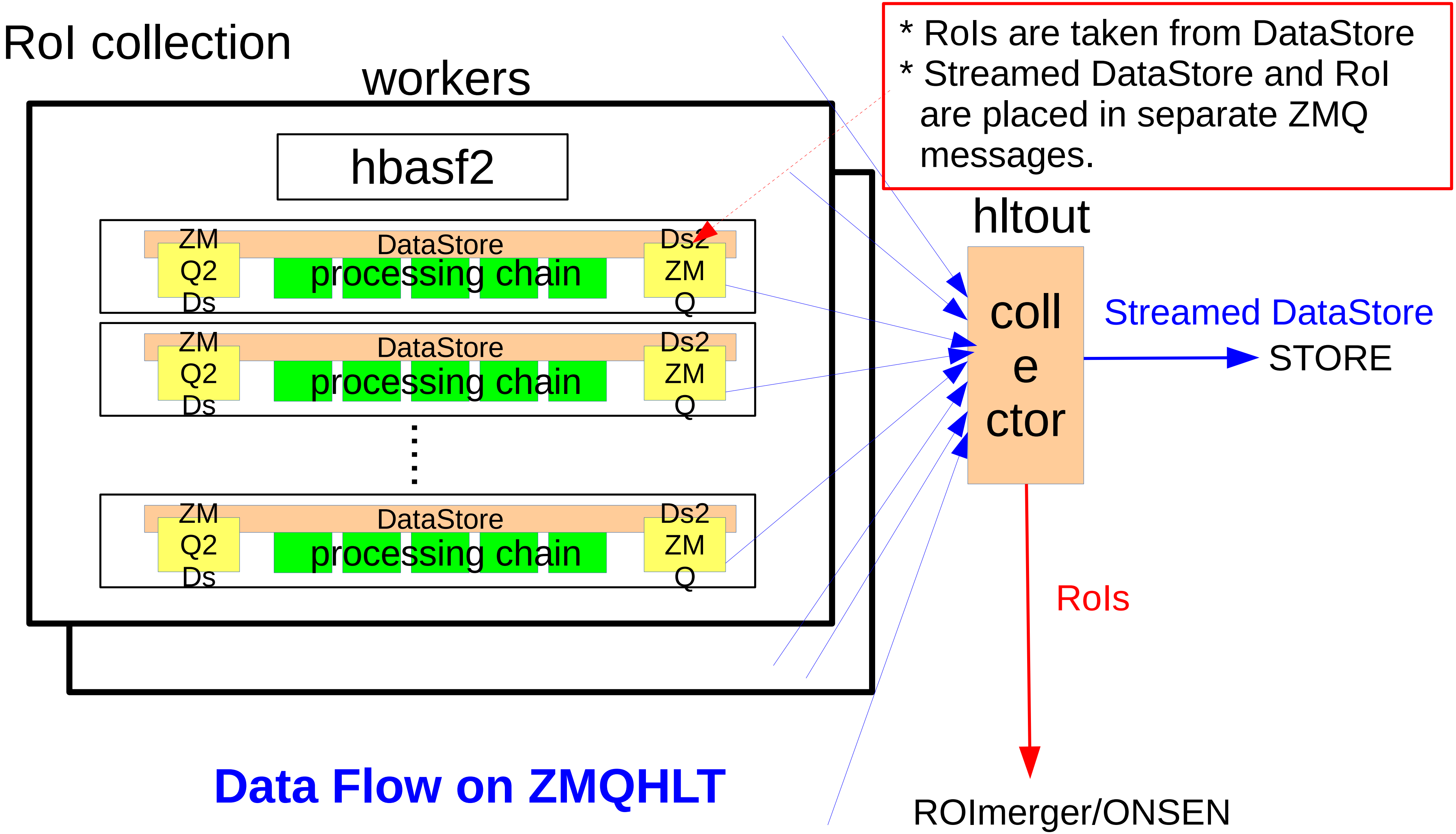
  -> It is done at “LOAD”

  -> Ready to process events promptly after run start.

Histogram Collection with ZMQ



- Histograms are transferred as streamed ROOT objects.
- They are collected in each worker and merged.
- They are streamed again and collected by final_histoserver.



Data Flow on ZMQHLT

HLT software : body of HLT processing

- Deployment of HLT processing software is managed by Seokhee.

- “Online” version of Belle2 library is released by Giacomo every two weeks together with the update of database (online global tag).

- Whenever a new version is released,
  * Giacomo test it offline using the recorded raw data.
  * Seokhee updates cvmfs and database on the maintenance day.

- In most cases, the library update have been working well without serious troubles.

- But one bad experience was there. When one new version (after offline test passed) was deployed in HLT and started cosmic ray run, frequent seg-fault occurred. => caused missing RoI!
  -> took some time to fix the problem.

- Issues were
  * When seg-fault occurs, the worker process is not recovered and the processing power drops and the event is lost.
  * The test of library with massive raw data is not done because hbasf2 parallel processing cannot run offline.

- Seokhee is now establishing a full test bench to test a new version in real HLT setup with a real data flow.

- But the process recovery mechanism should be implemented in the framework itself.

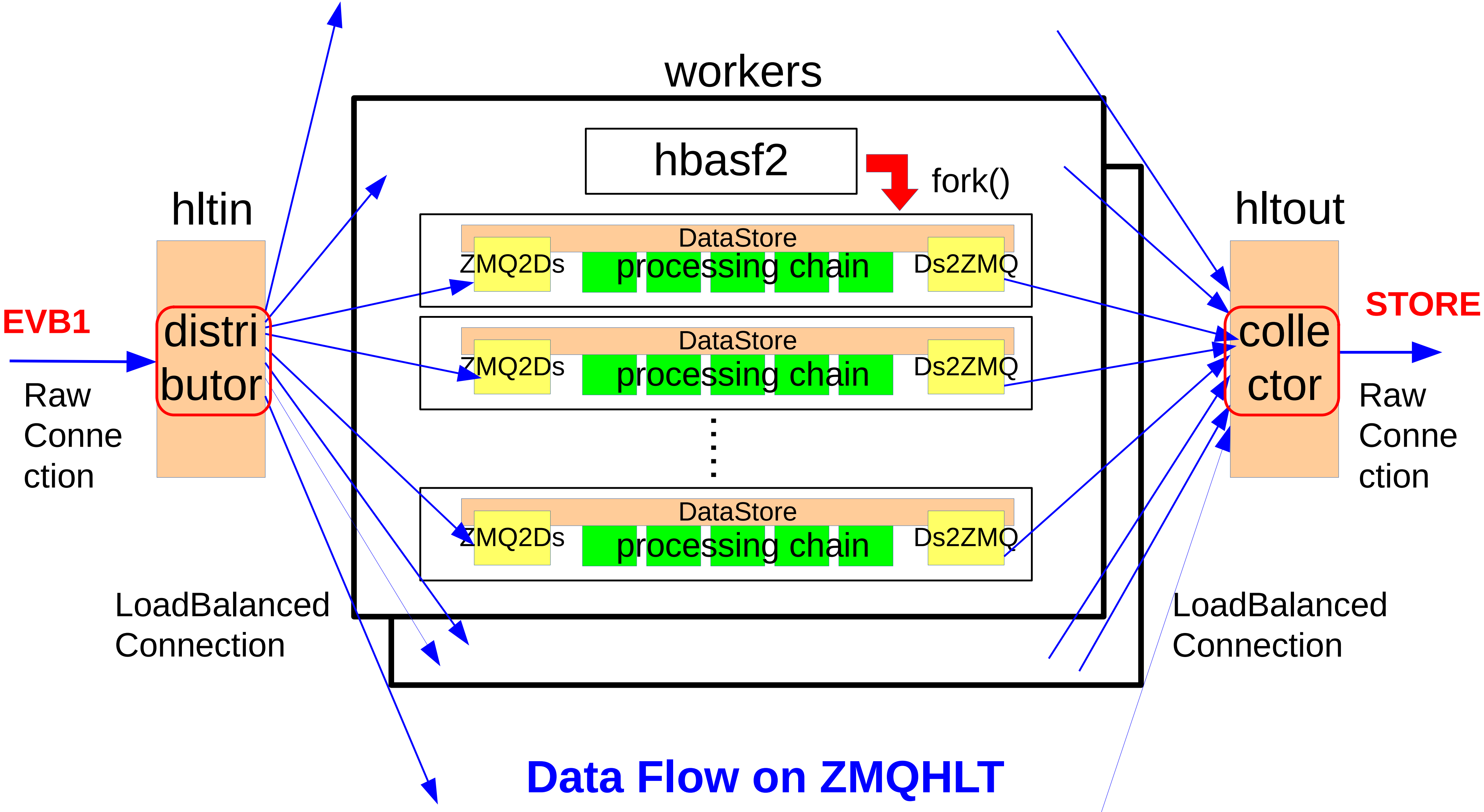
- Offline test scheme of parallel processing on multicore should also be implemented.
  <- Previous RFARM framework utilized original offline basf2 as
     the core framework which has the parallel processing capability
     on multicore (w/ IPC RingBuffer).

New core framework

2. basf2 recovery in HLT processing

- In current hbasf2 framework, when one of the event process dies in the middle of processing (ex. seg fault), the event is lost and the process is not restarted.
  -> * Source of “missing events/RoIs”
  * Processing power is lost and not recovered.

- hbasf2 cannot be invoked as a stand alone application with a parallel processing turned on. This makes the offline debugging of HLT processing difficult for rare troubles.
  <- Original basf2 used in RFARM could do this, but it is based on IPC ring buffer.



Data Flow on ZMQHLT

Idea of new implementation

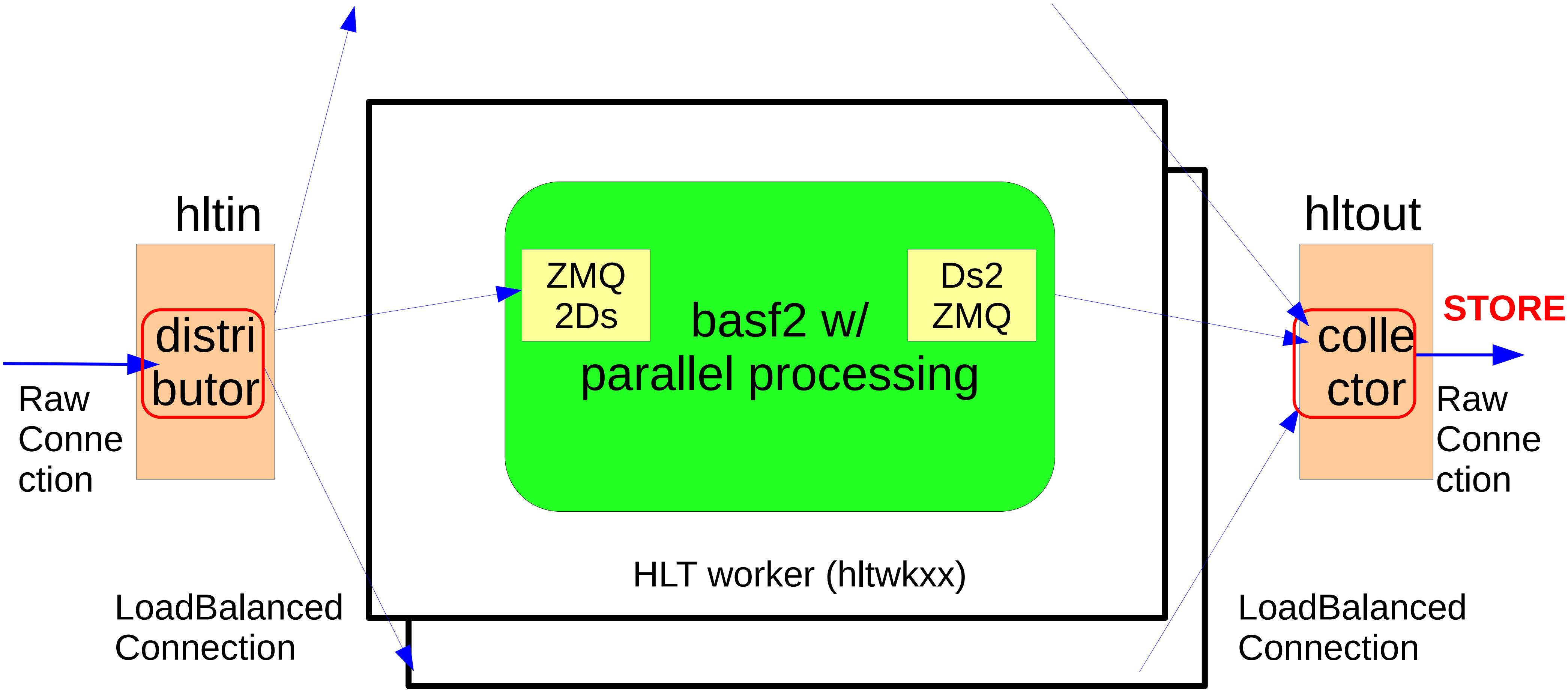
- Keep the framework outside workers unchanged (ZMQ-HLT).

- Replace the framework inside a worker (hbasf2) with the improved original basf2 parallel processing framework.

- At the beginning of each event process, the event data are copied to a buffer, and it is removed when processed successfully.

- If the process dies, basf2 mother process moves the faulty event to the output buffer (with a bad-event tag) and restart a new event process.

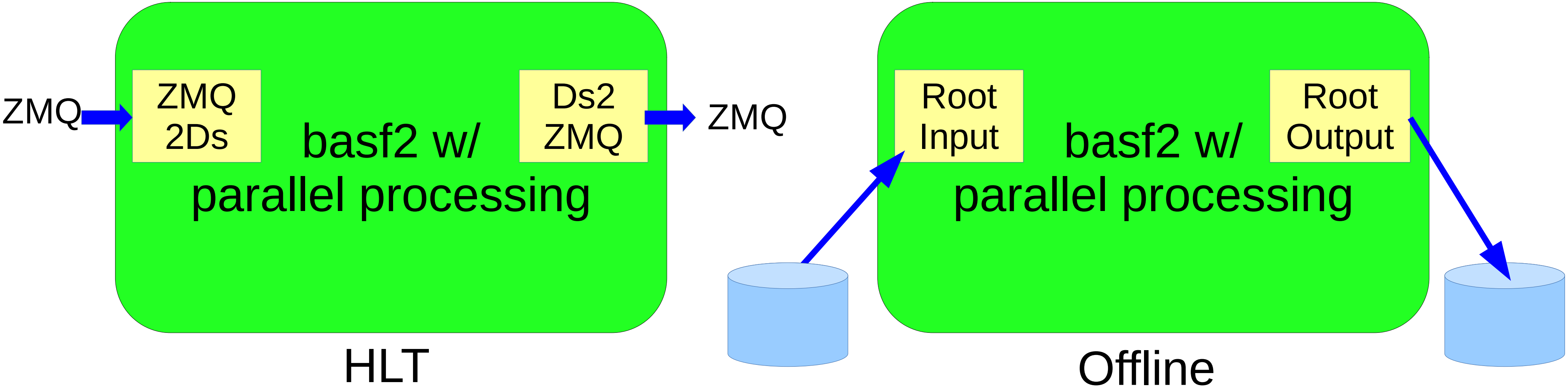
- IPC RingBuffer should be replaced with a better implementation.


hltin
distri
butor
Raw
Conne
ction
LoadBalanced
Connection
ZMQ
2Ds
basf2 w/
parallel processing
Ds2
ZMQ
HLT worker (hltwkxx)
hltout
colle
ctor
STORE
Raw
Conne
ction
LoadBalanced
Connection

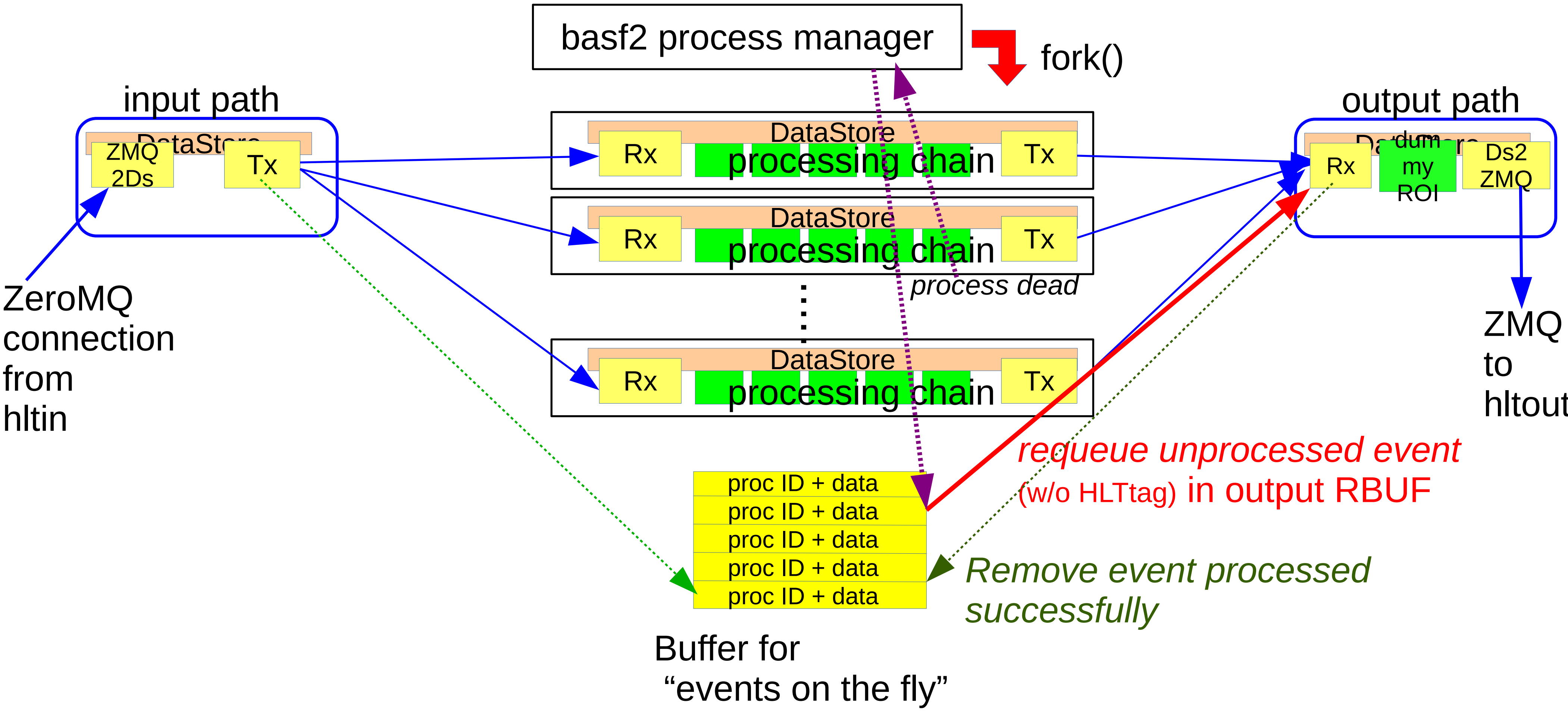
- A.Baur implemented N-to-1/1-to-N lock-free event transport using ZMQ IPC socket to replace RingBuffer in original basf2.

- In addition, the salvage mechanism of faulty event is also implemented using ZMQ broadcast.

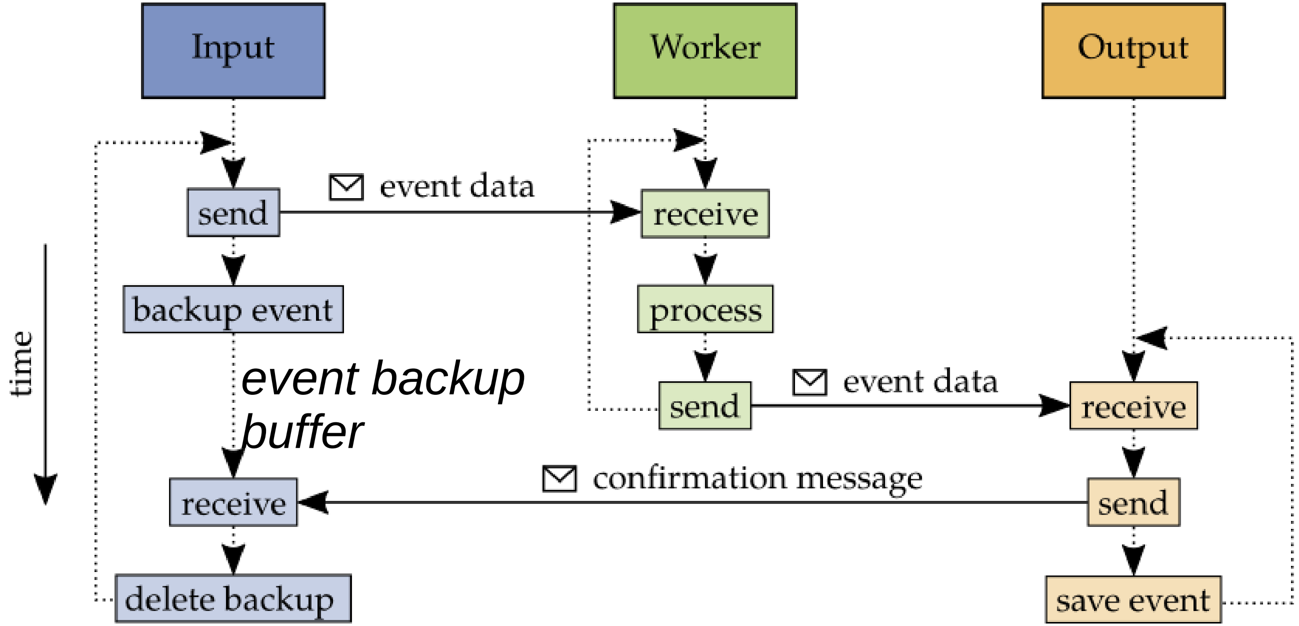
- Parallel processing in offline is possible.



Improved "basf2" data flow on a worker



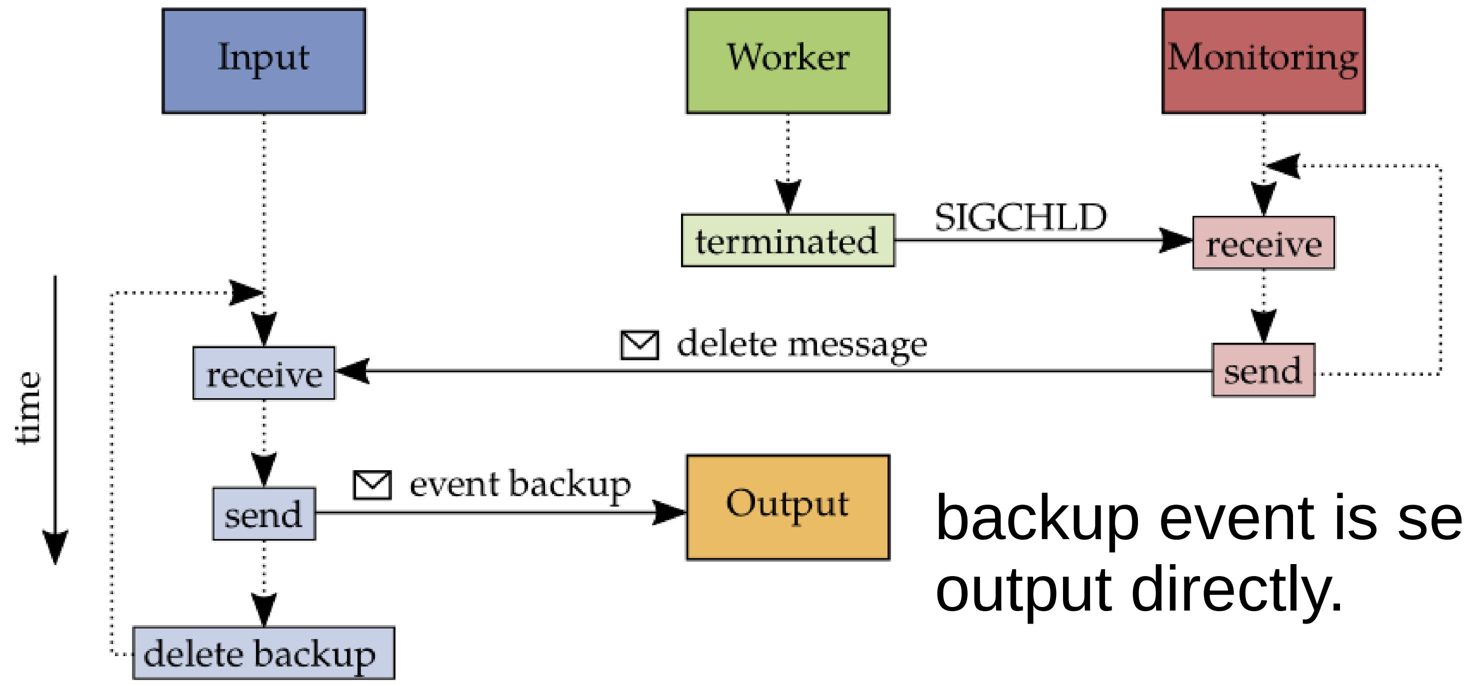
Event salvage mechanism



normal processing

Figure 4.2: The event backup is implemented in the input process. Every by the input process sent event is saved in the event backup list. Is an event received by the output process, a confirmation message with the unique event identification in the data frame is sent to the broadcast. If the input process receives such a confirmation message, the respective event is removed from the event backup list.

event process is crashed.



backup event is sent to output directly.

Figure 4.4: If a worker terminates unexpectedly, the monitoring process receives a SIGCHLD. According to that signal, a delete message with the corresponding worker identity is sent to the input process. The input process sends the respective event data from the corresponding worker to the output process.

Test of ZMQ-basf2 in offline

1. Event flow with ZMQ-based connection

- Real HLT script (beam_reco_monitor) is used.

- SeqRootInput/Output modules to read/write pre-recorded raw data files replacing ZMQ2Ds and Ds2ZMQ modules.

- Minor bug fixes to the ZMQ-basf2 source (DQM related).

- Data file : Exp 26, Run 1968. One SROOT file from QAS.

- Process granularity : 70 (on a 40 core server)

HLT script used for the test

```
### Input path
path = basf2.create_path()
path.add_module ('SeqRootInput', inputFileName=argvs[1] )

### Histogram handling
path.add_module ('HistoManager', histoFileName='testhist.root' )

### Body of processing
processing.add_hlt_processing ( path, run_type=constants.RunTypes.beam,
softwaretrigger_mode=constants.SoftwareTriggerModes.monitor)

### Output path
path.add_module ( 'SeqRootOutput', outputFileName='testout.sroot', saveObjs=sav\
e_objects )

### Monitoring
path.add_module ( 'Progress' )
```

```
[INFO] Starting event processing, random seed is set to '324c021bb99fc621d2b59cf874d6aa5ec48aea88561c63788ad271dfa1759273'
[INFO] Input Path [SeqRootInput -> HistoManager -> ZMQTxInput]
[INFO] Main Path [ZMQRxWorker -> [HistoManager -> Sum_Wait -> PruneDataStore -> Gearbox -> Geometry -> Sum_Initialization -> TTDUnpacker -> SVDUnpacker -> CDCUnpac
ker -> ECLUnpacker -> TOPUnpacker -> TOPRawDigitConverter -> ARICHUnpacker -> KLMUnpacker -> TRGGDLUnpacker -> TRGGDLSummary -> TRGECLUnpacker -> TRGGRLUnpacker ->
 TRGTOPUnpacker -> TRGCDCTSFUnpacker -> TRGCDCTSFUnpacker -> TRGCDCTSFUnpacker -> TRGCDCTSFUnpacker -> TRGCDCTSFUnpacker -> TRGCDCTSFUnpacker -> TRGCDCTSFUnpacker
-> TRGCDCT3DUnpacker -> TRGCDCT3DUnpacker -> TRGCDCT3DUnpacker -> TRGCDCT3DUnpacker -> CDCTriggerUnpacker -> Sum_Unpackers -> EventsOfDoomBuster(? >=1[EventErrorFl
ag -> KeepMetaData] ) -> Sum_EventsofDoomBuster -> ECLWaveformFit -> ECLDigitCalibrator -> ECLEventT0 -> ECLCRFinder -> ECLLocalMaximumFinder -> ECLSplitterN1 -> E
CLSplitterN2 -> ECLShowerCorrector -> ECLShowerCalibrator -> ECLShowerShape -> ECLClusterPSD -> ECLCovarianceMatrix -> Sum_Clustering -> RegisterEventLevelTracking
Info -> SVDClusterizer -> SVDMissingAPVsClusterCreator -> SVDTrackingEventLevelMdstInfoFiller -> SVDSpacePointCreator -> SetupGenfitExtrapolation -> TFCDC_WireHitP
reparer -> TFCDC_ClusterPreparer -> TFCDC_SegmentFinderFacetAutomaton -> TFCDC_AxialTrackFinderLegendre -> TFCDC_TrackQualityAsserter -> TFCDC_StereoHitFinder -> T
FCDC_SegmentTrackCombiner -> TFCDC_TrackQualityAsserter -> TFCDC_TrackQualityEstimator -> TFCDC_TrackExporter -> IPTrackTimeEstimator -> CDCHitBasedT0Extraction ->
 CDCTrackingEventLevelMdstInfoFiller -> CDCToSVDSpacePointCKF_backward -> CDCToSVDSpacePointCKF_forward -> SectorMapBootstrap -> SegmentNetworkProducer -> TrackFin
derVXDCellOMat -> AddVXDTrackCandidateSubSets -> QualityEstimatorVXD -> BestVXDTrackCandidatesSelector -> SPTCvirtualIPRemover -> SVDOverlapResolver -> SPTCmomentu
mSeedRetriever -> SPTC2RTConverter -> DAFRecoFitter -> CDCToSVDSeedCKF_backward -> DAFRecoFitter -> CDCToSVDSeedCKF_forward -> RelatedTracksCombiner -> ToCDCCKF ->
 CDCCKFTracksCombiner -> PruneRecoTracks -> PruneRecoTracks -> PruneRecoTracks -> PruneRecoTracks -> FullGridChi2TrackTimeExtractor -> TrackFinderMCTruthRecoTracks
 -> MCRecoTracksMatcher -> IPTrackTimeEstimator -> Combined_DAFRecoFitter -> TrackCreator -> Sum_Prefilter_Tracking -> CDCDedxPID -> VXDDedxPID -> Ext -> TOPChanne
lMasker -> TOPBunchFinder -> TOPReconstructor -> ARICHFillHits -> ARICHReconstructor -> EventT0Combiner -> OnlineEventT0Creator -> ECLFinalizer -> MCMatcherECLClus
ters -> KLMReconstructor -> KLMClustersReconstructor -> MCMatcherKLMClusters -> Muid -> ECLTrackClusterMatching -> ECLClusterProperties -> ECLChargedPID -> MdstPID
 -> KLMExpert -> ClusterMatcher -> ECLTrackBremFinder -> Sum_Posttracking_Reconstruction -> SoftwareTrigger -> Sum_HLT_Filter_Calculation -> SoftwareTriggerHLTDQM_
before_filter -> StatisticsTimingHLTDQM -> TRGECLDQM -> TRGGDLDQM -> TRGTOPDQM -> TRGGRLDQM -> TRGCDCTSFDQM -> TRGCDCTSFDQM -> TRGCDCTSFDQM -> TRGCDCTSFDQM -> TRGC
DCTSFDQM -> TRGCDCTSFDQM -> TRGCDCTSFDQM -> TRGCDCT2DDQM -> TRGCDCT3DConverter -> TRGCDCT3DDQM -> TRGCDCT3DConverter -> TRGCDCT3DDQM -> TRGCDCT3DConverter -> TRGCD
CT3DDQM -> TRGCDCT3DConverter -> TRGCDCT3DDQM -> CDCTriggerNeuroDQM -> Sum_HLT_DQM_before_filter -> V0Finder -> PruneRecoHits -> Sum_Postfilter_Reconstruction -> P
articleLoader_pi+:skim -> PListCopy_pi+:skim -> ParticleSelector_applyCuts_pi+:skim -> ParticleLoader_pi+:hadb -> PListCopy_pi+:hadb -> ParticleSelector_applyCuts_
pi+:hadb -> ParticleLoader_pi+:tau -> PListCopy_pi+:tau -> ParticleSelector_applyCuts_pi+:tau -> ParticleLoader_gamma:skim -> PListCopy_gamma:skim -> ParticleSelec
tor_applyCuts_gamma:skim -> ParticleSelector_applyCuts_gamma:skim -> ParticleLoader_K_S0:V0 -> pi+ pi- -> PListCutAndCopy_K_S0:V0_MassWindow -> ParticleVertexFitte
r_K_S0:V0_MassWindow -> ParticleSelector_applyCuts_K_S0:V0_MassWindow -> ParticleLoader_pi+:all -> ParticleCombiner_K_S0:RD -> pi+:all pi-:all -> ParticleVertexFit
ter_K_S0:RD -> ParticleSelector_applyCuts_K_S0:RD -> PListMerger_K_S0:merged -> PListCutAndCopy_K_S0:dstSkim -> ParticleLoader_Lambda0:V0 -> p+ pi- -> PListCutAndC
opy_Lambda0:V0_MassWindow -> TreeFitter_Lambda0:V0_MassWindow -> ParticleSelector_applyCuts_Lambda0:V0_MassWindow -> DuplicateVertexMarker -> ParticleSelector_appl
yCuts_Lambda0:V0_MassWindow -> ParticleLoader_pi+:all -> ParticleLoader_p+:all -> ParticleCombiner_Lambda0:RD -> p+:all pi-:all -> TreeFitter_Lambda0:RD -> Particl
eSelector_applyCuts_Lambda0:RD -> DuplicateVertexMarker -> ParticleSelector_applyCuts_Lambda0:RD -> PListMerger_Lambda0:merged -> ParticleLoader_K+:dstSkim -> PLis
tCopy_K+:dstSkim -> ParticleSelector_applyCuts_K+:dstSkim -> ParticleLoader_pi+:dstSkim -> PListCopy_pi+:dstSkim -> ParticleSelector_applyCuts_pi+:dstSkim -> Parti
cleLoader_gamma:loose -> PListCopy_gamma:loose -> ParticleSelector_applyCuts_gamma:loose -> ParticleSelector_applyCuts_gamma:loose -> ParticleCombiner_pi0:loose ->
 gamma:loose gamma:loose -> PListCutAndCopy_pi0:veryLooseFit -> ParticleVertexFitter_pi0:veryLooseFit -> ParticleCombiner_D0:ch1 -> K-:dstSkim pi+:dstSkim -> Parti
cleCombiner_D0:ch2 -> K-:dstSkim pi+:dstSkim pi0:veryLooseFit -> ParticleCombiner_D0:ch3 -> K-:dstSkim pi+:dstSkim pi-:dstSkim pi+:dstSkim -> ParticleCombiner_D0:c
h4 -> K_S0:dstSkim pi+:dstSkim pi-:dstSkim -> ParticleCombiner_D*+:ch1 -> D0:ch1 pi+:all -> ParticleCombiner_D*+:ch2 -> D0:ch2 pi+:all -> ParticleCombiner_D*+:ch3
-> D0:ch3 pi+:all -> ParticleCombiner_D*+:ch4 -> D0:ch4 pi+:all -> PListCopy_D*+:d0pi -> ParticleLoader_pi+:offip -> PListCopy_pi+:offip -> ParticleSelector_applyC
uts_pi+:offip -> ParticleLoader_pi+:GoodTrackForHLT -> PListCopy_pi+:GoodTrackForHLT -> ParticleSelector_applyCuts_pi+:GoodTrackForHLT -> ParticleLoader_K+:GoodTra
ckForHLT -> PListCopy_K+:GoodTrackForHLT -> ParticleSelector_applyCuts_K+:GoodTrackForHLT -> ParticleLoader_gamma:all -> ParticleSelector_applyCuts_gamma:all -> Pa
rticleCombiner_pi0:all -> gamma:all gamma:all -> MCMatch_pi0:all -> PListCutAndCopy_pi0:GoodPi0ForHLT -> ParticleCombiner_D0:KpiForHLT -> K-:GoodTrackForHLT pi+:Go
odTrackForHLT -> ParticleCombiner_D0:Kpipi0ForHLT -> K-:GoodTrackForHLT pi+:GoodTrackForHLT pi0:GoodPi0ForHLT -> ParticleCombiner_D0:KpipipiForHLT -> K-:GoodTrackF
orHLT pi+:GoodTrackForHLT pi+:GoodTrackForHLT pi-:GoodTrackForHLT -> ParticleCombiner_D+:KpipiForHLT -> K-:GoodTrackForHLT pi+:GoodTrackForHLT pi+:GoodTrackForHLT
-> ParticleCombiner_D*+:D0_KpiForHLT -> D0:KpiForHLT pi+:GoodTrackForHLT -> ParticleCombiner_D*+:D0_Kpipi0ForHLT -> D0:Kpipi0ForHLT pi+:GoodTrackForHLT -> Particle
Combiner_D*+:D0_KpipipiForHLT -> D0:KpipipiForHLT pi+:GoodTrackForHLT -> ParticleCombiner_B+:BtoD0pi_KpiForHLT -> anti-D0:KpiForHLT pi+:GoodTrackForHLT -> Particle
Combiner_B+:BtoD0pi_Kpipi0ForHLT -> anti-D0:Kpipi0ForHLT pi+:GoodTrackForHLT -> ParticleCombiner_B+:BtoD0pi_KpipipiForHLT -> anti-D0:KpipipiForHLT pi+:GoodTrackFor
HLT -> ParticleCombiner_B0:B0toDpi_KpipiForHLT -> D-:KpipiForHLT pi+:GoodTrackForHLT -> ParticleCombiner_B0:B0toDstarPi_D0pi_KpiForHLT -> D*-:D0_KpiForHLT pi+:Good
TrackForHLT -> ParticleCombiner_B0:B0toDstarPi_D0pi_KpipipiForHLT -> D*-:D0_KpipipiForHLT pi+:GoodTrackForHLT -> ParticleCombiner_B0:B0toDstarPi_D0pi_Kpipi0ForHLT
-> D*-:D0_Kpipi0ForHLT pi+:GoodTrackForHLT -> PListCopy_B+:BtoCharmForHLT -> PListCopy_B0:BtoCharmForHLT -> SoftwareTrigger -> Sum_HLT_Skim_Calculation -> PXDROIFi
nder -> Sum_ROI_Finder -> TTDDQM -> SoftwareTriggerHLTDQM -> SoftwareTriggerHLTDQM_skim_nobhabha -> SVDUnpackerDQM -> cdcDQM7 -> CDCDedxDQM -> ECLDQM -> ECLDQMEXTE
NDED -> TOPDQM -> KLMDQM -> TRGGDLDQM -> TRGTOPDQM -> TrackingHLTDQM -> ARICHDQM -> ParticleLoader_mu+:KLMDQM -> PListCopy_mu+:KLMDQM -> ParticleSelector_applyCuts
_mu+:KLMDQM -> ParticleLoader_gamma:physDQM -> PListCopy_gamma:physDQM -> ParticleSelector_applyCuts_gamma:physDQM -> ParticleSelector_applyCuts_gamma:physDQM -> P
articleLoader_pi+:physDQM -> PListCopy_pi+:physDQM -> ParticleSelector_applyCuts_pi+:physDQM -> ParticleLoader_mu+:physDQM -> PListCopy_mu+:physDQM -> ParticleSele
ctor_applyCuts_mu+:physDQM -> ParticleCombiner_pi0:physDQM -> gamma:physDQM gamma:physDQM -> ParticleCombiner_K_S0:physDQM -> pi-:physDQM pi+:physDQM -> ParticleCo
mbiner_Upsilon:physDQM -> mu-:physDQM mu+:physDQM -> ParticleLoader_pi+:evtshape -> PListCopy_pi+:evtshape -> ParticleLoader_gamma:evtshape -> PListCopy_gamma:evts
hape -> ParticleSelector_applyCuts_gamma:evtshape -> EventShape -> PhysicsObjectsDQM -> KLMDQM2 -> ParticleLoader_mu+:DQM_HLT -> PListCopy_mu+:DQM_HLT -> ParticleS
elector_applyCuts_mu+:DQM_HLT -> ParticleCombiner_Upsilon(4S):IPDQM_HLT -> mu+:DQM_HLT mu-:DQM_HLT -> ParticleVertexFitter_Upsilon(4S):IPDQM_HLT -> IPDQM -> Sum_HL
T_DQM_filtered -> ROIPayloadAssembler -> Sum_ROI_Payload_Assembler -> DAQMonitor -> DelayDQM -> Sum_HLT_DQM_all_events -> KeepRawData -> Sum_Close_Event -> ZMQTxW
orker]
[INFO] Output Path [ZMQRxOutput -> [HistoManager -> SeqRootOutput -> Progress]]
```

```
top - 15:06:44 up 8 days, 21:55,  3 users,  load average: 68.22, 44.14, 23.98
Tasks: 869 total,  72 running, 797 sleeping,   0 stopped,   0 zombie
%Cpu(s): 88.0 us,  0.8 sy,  0.0 ni, 10.8 id,  0.4 wa,  0.0 hi,  0.0 si,  0.0 s
KiB Mem : 97578280 total, 68372984 free, 21345452 used,  7859844 buff/cache
KiB Swap: 67108860 total, 67108860 free,        0 used. 75571552 avail Mem

   PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
224185 itoh      20   0 2408432   1.6g 132092 R 100.3  1.7   4:32.25 basf2
224520 itoh      20   0 2435012   1.6g 131176 R 100.3  1.7   4:32.73 basf2
224548 itoh      20   0 2420804   1.6g 132028 R 100.3  1.7   4:33.04 basf2
224554 itoh      20   0 2402736   1.6g 132140 R 100.3  1.7   4:32.98 basf2
224830 itoh      20   0 2425152   1.6g 132148 R 100.3  1.7   4:33.44 basf2
224848 itoh      20   0 2413780   1.6g 132288 R 100.3  1.7   4:33.33 basf2
224080 itoh      20   0 2154824   1.5g 219696 R 100.0  1.6   5:40.06 basf2
224148 itoh      20   0 2406408   1.6g 130996 R 100.0  1.7   4:32.13 basf2
224151 itoh      20   0 2402716   1.6g 132060 R 100.0  1.7   4:32.45 basf2
224169 itoh      20   0 2402984   1.6g 130956 R 100.0  1.7   4:32.61 basf2
224172 itoh      20   0 2415012   1.6g 131020 R 100.0  1.7   4:31.79 basf2
224175 itoh      20   0 2415404   1.6g 130980 R 100.0  1.7   4:32.73 basf2
224221 itoh      20   0 2418520   1.6g 132160 R 100.0  1.7   4:32.18 basf2
224242 itoh      20   0 2416280   1.6g 131092 R 100.0  1.7   4:32.60 basf2
224312 itoh      20   0 2401228   1.6g 130968 R 100.0  1.7   4:32.50 basf2
224380 itoh      20   0 2409608   1.6g 131184 R 100.0  1.7   4:32.24 basf2
224400 itoh      20   0 2407336   1.6g 131048 R 100.0  1.7   4:32.92 basf2
224446 itoh      20   0 2410644   1.6g 131560 R 100.0  1.7   4:32.87 basf2
224463 itoh      20   0 2391020   1.6g 132224 R 100.0  1.7   4:32.60 basf2
224469 itoh      20   0 2395472   1.6g 132096 R 100.0  1.7   4:32.79 basf2
224509 itoh      20   0 2418820   1.6g 132056 R 100.0  1.7   4:32.80 basf2
224537 itoh      20   0 2429760   1.6g 132060 R 100.0  1.7   4:32.88 basf2
224570 itoh      20   0 2421080   1.6g 131012 R 100.0  1.7   4:33.32 basf2
224590 itoh      20   0 2401804   1.6g 131100 R 100.0  1.7   4:33.13 basf2
224610 itoh      20   0 2410312   1.6g 131240 R 100.0  1.7   4:32.77 basf2
224618 itoh      20   0 2413320   1.6g 131164 R 100.0  1.7   4:32.98 basf2
224648 itoh      20   0 2411752   1.6g 132020 R 100.0  1.7   4:33.10 basf2
224689 itoh      20   0 2413660   1.6g 132080 R 100.0  1.7   4:32.97 basf2
224697 itoh      20   0 2405692   1.6g 131372 R 100.0  1.7   4:32.93 basf2
224705 itoh      20   0 2455684   1.6g 132184 R 100.0  1.8   4:33.06 basf2
224713 itoh      20   0 2414540   1.6g 130940 R 100.0  1.7   4:33.23 basf2
224730 itoh      20   0 2409160   1.6g 132084 R 100.0  1.7   4:33.43 basf2
224738 itoh      20   0 2448232   1.6g 131748 R 100.0  1.7   4:33.15 basf2
224761 itoh      20   0 2403336   1.6g 131080 R 100.0  1.7   4:33.45 basf2
224777 itoh      20   0 2422736   1.6g 130988 R 100.0  1.7   4:33.45 basf2
224798 itoh      20   0 2396972   1.6g 131024 R 100.0  1.7   4:33.57 basf2
224858 itoh      20   0 2417700   1.6g 131580 R 100.0  1.7   4:33.40 basf2
```

- Full CPU consumption is confirmed.

- No bottleneck observed in framework.

2. Test of salvage of faulty event

- Insert “TheKiller” module in the HLT processing script which generates various troubles in the processing. Seg fault was generated using this module at 10th event.

- The output file was examined and checked that the “faulty” event is properly transferred to the output.

- The restart of new event process after the seg fault is also checked.

```
[INFO] ==========================================================================
[INFO] Processed:   1 runs,      8/     0 events.
[INFO] ==========================================================================
[INFO] DataStore collections in event 20997130
[INFO] ==========================================================================
[INFO] Type                    Name                          #Entries          <Event>
[INFO] EventMetaData           EventMetaData
[INFO] ROIpayload              ROIpayload
[INFO] SoftwareTriggerResult  SoftwareTriggerResult
[INFO] SoftwareTrigger::SoftwareTriggerVariables  SoftwareTriggerVariables
[INFO] OnlineEventT0[]         OnlineEventT0s                0
[INFO] ROIid[]                 ROIs                          0
[INFO] RawARICH[]              RawARICHs                     0
[INFO] RawCDC[]                RawCDCs                       0
[INFO] RawECL[]                RawECLs                       0
[INFO] RawFTSW[]               RawFTSWs                      1
[INFO] RawKLM[]                RawKLMs                       0
[INFO] RawPXD[]                RawPXDs                       4
[INFO] RawSVD[]                RawSVDs                       0
[INFO] RawTOP[]                RawTOPs                       0
[INFO] RawTRG[]                RawTRGs                       0
[INFO]
[INFO] --------------------------------------------------------------------------
[INFO] Type                    Name                          #Entries     <Persistent>
[INFO] ProcessStatistics       ProcessStatistics
[INFO]
[INFO] ==========================================================================
[INFO] Processed:   1 runs,      9/     0 events.
[INFO] ==========================================================================
[INFO] DataStore collections in event 20997830
[INFO] ==========================================================================
[INFO] Type                    Name                          #Entries          <Event>
[INFO] EventMetaData           EventMetaData
[INFO] OnlineEventT0[]         OnlineEventT0s                0
[INFO] ROIid[]                 ROIs                          0
[INFO] RawARICH[]              RawARICHs                     0
[INFO] RawCDC[]                RawCDCs                       0
[INFO] RawECL[]                RawECLs                       0
[INFO] RawFTSW[]               RawFTSWs                      1
[INFO] RawKLM[]                RawKLMs                       0
[INFO] RawPXD[]                RawPXDs                       4
[INFO] RawSVD[]                RawSVDs                       0
[INFO] RawTOP[]                RawTOPs                       0
[INFO] RawTRG[]                RawTRGs                       0
[INFO]
[INFO] --------------------------------------------------------------------------
```

Event processed normally. SoftwareTriggerResults is there.

Seg-faulted event. Only RawData are there with Empty RoI

Test bench with full data flow with 20 workers (HLT03)

- Input source : eb1rx is turned off. Instead, the raw data are fed into HLT distributor directly using “nc” on hltin.

```
hltin% nc -l 5121 < /data1/itoh/01968/physics.0026.01968.HLT1.f00009.rawdata
```

- Output sink: storagerd is not used. Instead, the data from HLT collector are received by “nc”

```
[stordaq@storage store03]$ nc tostor 4100 >  /dev/null
```

Id	Node	State
RC	RC_HLT03	READY
01	DISTRIBUTOR_HL	READY
02	COLLECTOR_HLT0	READY
03	DQMSERVER_HLT0	READY
04	EB1_HLT03	OFF
05	EVP_HLTWK01_HL	READY
06	EVP_HLTWK02_HL	READY
07	EVP_HLTWK03_HL	READY
08	EVP_HLTWK04_HL	READY
09	EVP_HLTWK05_HL	READY
10	EVP_HLTWK06_HL	READY
11	EVP_HLTWK07_HL	READY
12	EVP_HLTWK08_HL	READY
13	EVP_HLTWK09_HL	READY
14	EVP_HLTWK10_HL	READY
15	EVP_HLTWK11_HL	READY
16	EVP_HLTWK12_HL	READY
17	EVP_HLTWK13_HL	READY
18	EVP_HLTWK14_HL	READY
19	EVP_HLTWK15_HL	READY
20	EVP_HLTWK16_HL	READY
21	EVP_HLTWK17_HL	READY
22	EVP_HLTWK18_HL	READY
23	EVP_HLTWK19_HL	READY
24	EVP_HLTWK20_HL	READY

- Full 20 worker configuration

- Controlled using “rcrequest”

- Just by “LOAD”ing the system, the raw data can be processed.

Replacement of hbasf2 with native ZMQ-basf2

- hbasf2 (as a python3 script) is directly invoked from hltworkerd and a minor modification to hltworkerd is necessary.

daq_slc/apps/hltd/src/HLTWorkerCallback.cc

```
StringList& basf2Command = m_commands["basf2"];
/* for hbasf2
basf2Command.push_back("python3");
basf2Command.push_back(basf2Script);
basf2Command.push_back("--input");
basf2Command.push_back(StringUtil::form("tcp://%s:%d", inputHost.c_str(), inputPort));
basf2Command.push_back("--output");
basf2Command.push_back(StringUtil::form("tcp://%s:%d", outputHost.c_str(), outputPort));
basf2Command.push_back("--dqm");
basf2Command.push_back(StringUtil::form("tcp://localhost:%d", dqmInternalPort));
*/
// for naked zmq-basf2
basf2Command.push_back("basf2"); // Use native basf2 instead of hbasf2/python3
basf2Command.push_back("--zmq"); // Use ZMQ-basf2
basf2Command.push_back(basf2Script);
basf2Command.push_back(StringUtil::form("tcp://%s:%d", inputHost.c_str(), inputPort)); // input as the first arg
basf2Command.push_back(StringUtil::form("tcp://%s:%d", outputHost.c_str(), outputPort)); // output as the 2nd arg
basf2Command.push_back(StringUtil::form("tcp://localhost:%d", dqmInternalPort)); // dqm as the 3rd arg
```

HLT script for the test with ZMQ-basf2

The same ZMQ I/O and DQM modules used in hbasf2

i# Local DB specification

basf2.conditions.override_globaltags()
basf2.conditions.globaltags=["online"]
local_db_path = constants.DEFAULT_DB_FILE_LOCATION

basf2.conditions.metadata_providers =
["file://" + basf2.find_file(local_db_path + "/metadata.sqlite")]
basf2.conditions.payload_locations = [basf2.find_file(local_db_path)]

Parallel processing

basf2.set_nprocesses(multiprocessing.cpu_count()-5)

save_objects = constants.ALWAYS_SAVE_OBJECTS +
constants.RAWDATA_OBJECTS
basf2.set_streamobjs(save_objects)

# Logging
basf2.set_log_level(basf2.LogLevel.ERROR)
# Online Realm
basf2.set_realm("online")

Input path
path = basf2.create_path()
path.add_module ('HLTZMQ2Ds', input=argvs[1])

Histogram handling
path.add_module ('HLTDQM2ZMQ', output=argvs[3])

Body of processing
processing.add_hlt_processing (path, run_type=constants.
RunTypes.beam, softwaretrigger_mode=
constants.SoftwareTriggerModes.monitor)

Output path
path.add_module ('HLTDs2ZMQ', output=argvs[2], raw=True)

Monitoring
path.add_module ('Progress')

Run
basf2.print_path (path)
basf2.process (path)

ZMQHLT data flow monitor

* All the workers are fully working.

dead workers		0
ready queue size	0	
data size	19972.4	30021.24
event rate	646.221732	651.754455
registered workers	20	20
raw socket state	connected	connected

	data size	hosts	event rate	events	ready messages
from hltwk01	30080.04	1	19.527817	1484164	
to hltwk01	14542.92	1	20.831297	1484271	0.0
from hltwk02	24458.68	1	22.395819	1498880	
to hltwk02	14092.84	1	25.144091	1498987	0.0
from hltwk03	22232.76	1	29.127094	1497897	
to hltwk03	13871.84	1	29.685902	1498005	0.0
from hltwk04	28267.36	1	25.358105	1504033	
to hltwk04	13093.68	1	27.699765	1504141	0.0
from hltwk05	24474.36	1	27.478399	1498529	
to hltwk05	9227.72	1	26.656239	1498637	0.0
from hltwk06	27331.00	1	27.348619	1502338	
to hltwk06	9459.12	1	20.484355	1502446	0.0
from hltwk07	33827.48	1	16.229725	1492028	
to hltwk07	19453.00	1	21.414395	1492135	0.0
from hltwk08	28427.08	1	16.921951	1487186	
to hltwk08	14224.84	1	19.388329	1487298	0.0
from hltwk09	30523.56	1	23.036765	1497724	
to hltwk09	20891.04	1	21.510119	1497832	0.0
from hltwk10	29160.80	1	20.296934	1499314	
to hltwk10	15980.92	1	17.830401	1499422	0.0
from hltwk11	35026.92	1	23.442243	1479231	
to hltwk11	23767.80	1	23.504977	1479339	0.0
from hltwk12	34541.08	1	19.521210	1500704	
to hltwk12	19771.00	1	20.871223	1500812	0.0
from hltwk13	32799.48	1	23.004530	1501775	
to hltwk13	17682.20	1	22.908225	1501883	0.0
from hltwk14	40463.20	1	16.883485	1505603	
to hltwk14	15730.32	1	19.817143	1505711	0.0
from hltwk15	26496.08	1	23.915252	1499650	
to hltwk15	14992.28	1	17.012141	1499758	0.0
from hltwk16	28104.72	1	22.756988	1496163	
to hltwk16	12455.48	1	18.221939	1496271	0.0
from hltwk17	30588.72	1	64.315306	3432117	
to hltwk17	18491.44	1	65.027391	3432321	0.0
from hltwk18	36979.32	1	73.898328	3437316	
to hltwk18	24598.64	1	73.951149	3437520	0.0
from hltwk19	31410.80	1	59.539138	4201290	
to hltwk19	16895.00	1	56.641443	4201518	0.0
from hltwk20	27916.84	1	81.335361	4420387	

- Confirmed to work stably more than a few days with repeated use of a portion of recorded real raw data.

- The last step is the implementation of on-the-fly switching mechanism between hbasf2 and zmq-basf2 in hltworkerd.
    -> mixed-operation of two frameworks unit-by-unit becomes possible. Maybe necessary at the first debugging stage of new framework in beam run.

- Plan: complete the development by the end of this year and submit a PR to merge mods in Belle2 library/daq_slc.
    => Test in GCR/HRT from early next year.

Backup Slides

RoI feedback to Pixel Readout

- The results of tracking using Silicon Vertex Detector(SVD) and Central Drift Chamber(CDC) are extrapolated onto the surface of PXD and boxes (Region of Interest:ROIs) are defined.
- The coordinates of ROIs are sent to PXD readout and only the hits in the boxes are saved.
- The reduction factor is expected to be better than 1/10.