DQM Operation in Phase 3

R.Itoh, KEK

1. Introduction

- Real time monitoring of the data quality is essential for the stable data taking.

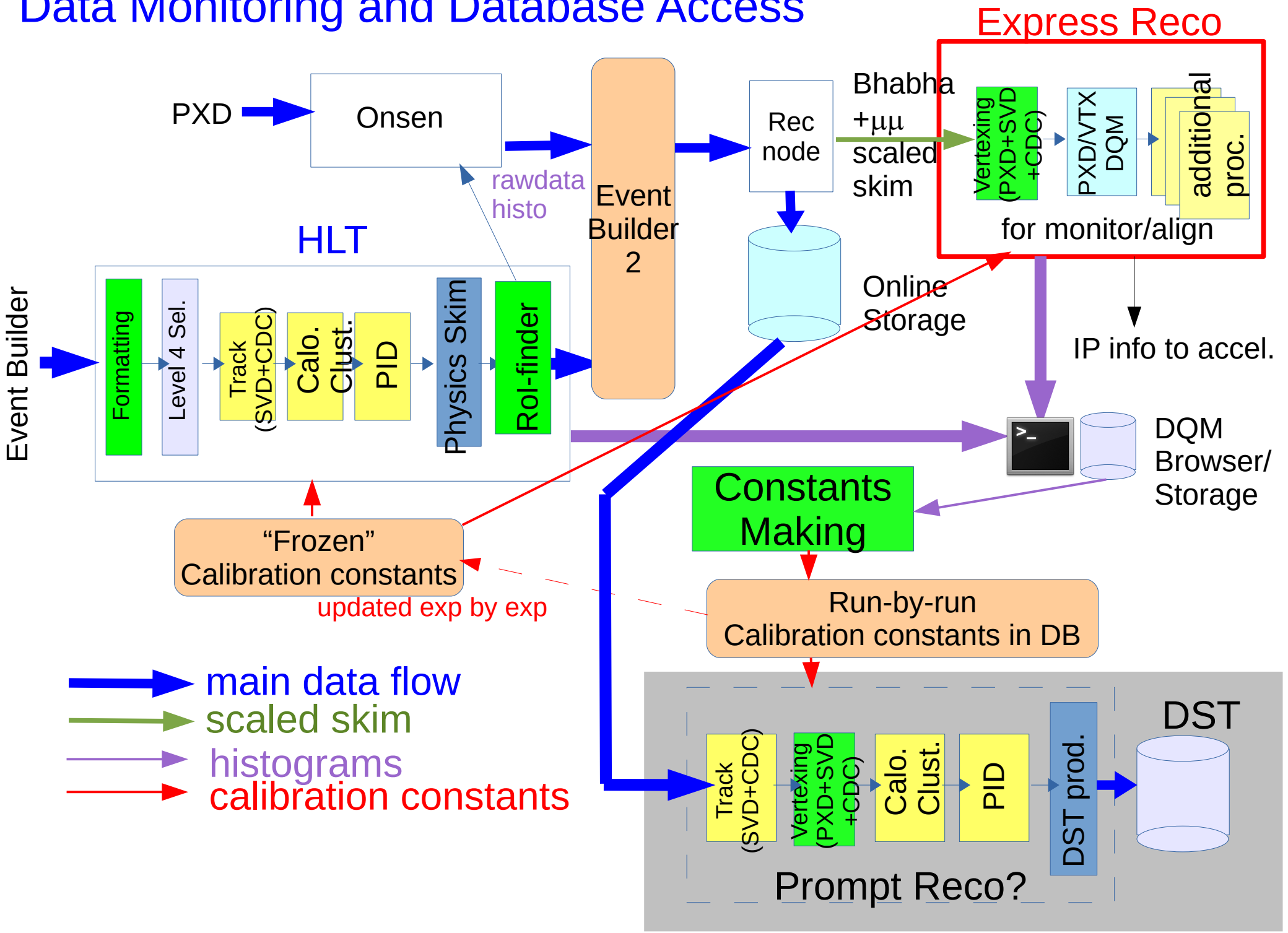
- The monitoring consists of
  * Detector performance monitor (i.e. hit-map, gain variation, etc.)
  * Physics performance monitor (i.e. momentum resolution, etc.)
  * Trigger performance (Various dist. used in L1/HLT selection)

- In Belle II DAQ, such monitoring is performed on HLT and Express Reco.

- The real time monitoring is implemented by the periodical "spy" of 1D and 2D histograms from live basf2 processes.

- At run ends, the accumulated histograms and N-tuples(TTrees) are supposed to be left on the storage for quick analysis and calibration.

Data Monitoring and Database Access



What is DQM?

DQM : Data Quality Monitoring
  * Viewpoint from DAQ

- The basic functionality of DQM is to accumulate histograms on various parameters obtained in the real time HLT and ExpressReco processing, and transfer them lively to the browser node to be monitored by shifters.

- At each run end, the histograms are stored in a file for the offline usage / comparison.

Technical challenge in DQM

- On HLT/Expressreco, histograms are accumulated on many cores (1600 cores for HLT, 160 cores for ExpReco) in the parallel processing.

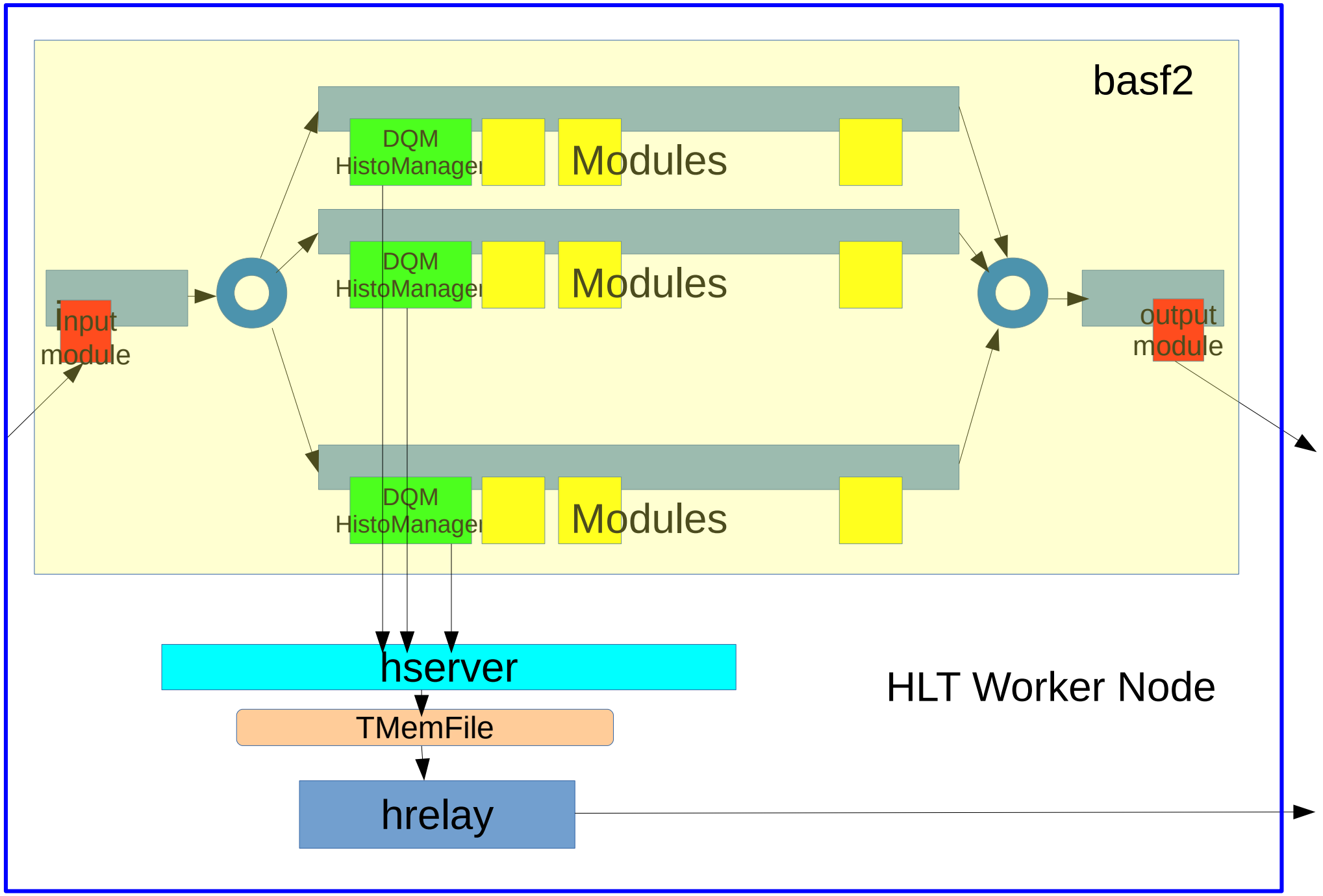
- The histograms have to be collected from these cores and merged lively and periodically.

- The histograms are collected in three steps.
  1) On each HLT worker node, a basf2 is running in parallel processing mode. The histograms are collected by the combination of “DQMHistoManager” module and “hserver”.

  2) The histograms merged on a hserver on each worker are transferred to the control node of the HLT unit by “hrelay” and then merged again by hserver.

  3) The histograms from multiple HLT units are merged on the DQM server node by using hrelay-hserver.

Step 1) Histogram collection on a worker node



DqmHistoManager module

- The module is compatible with HistoManager module.

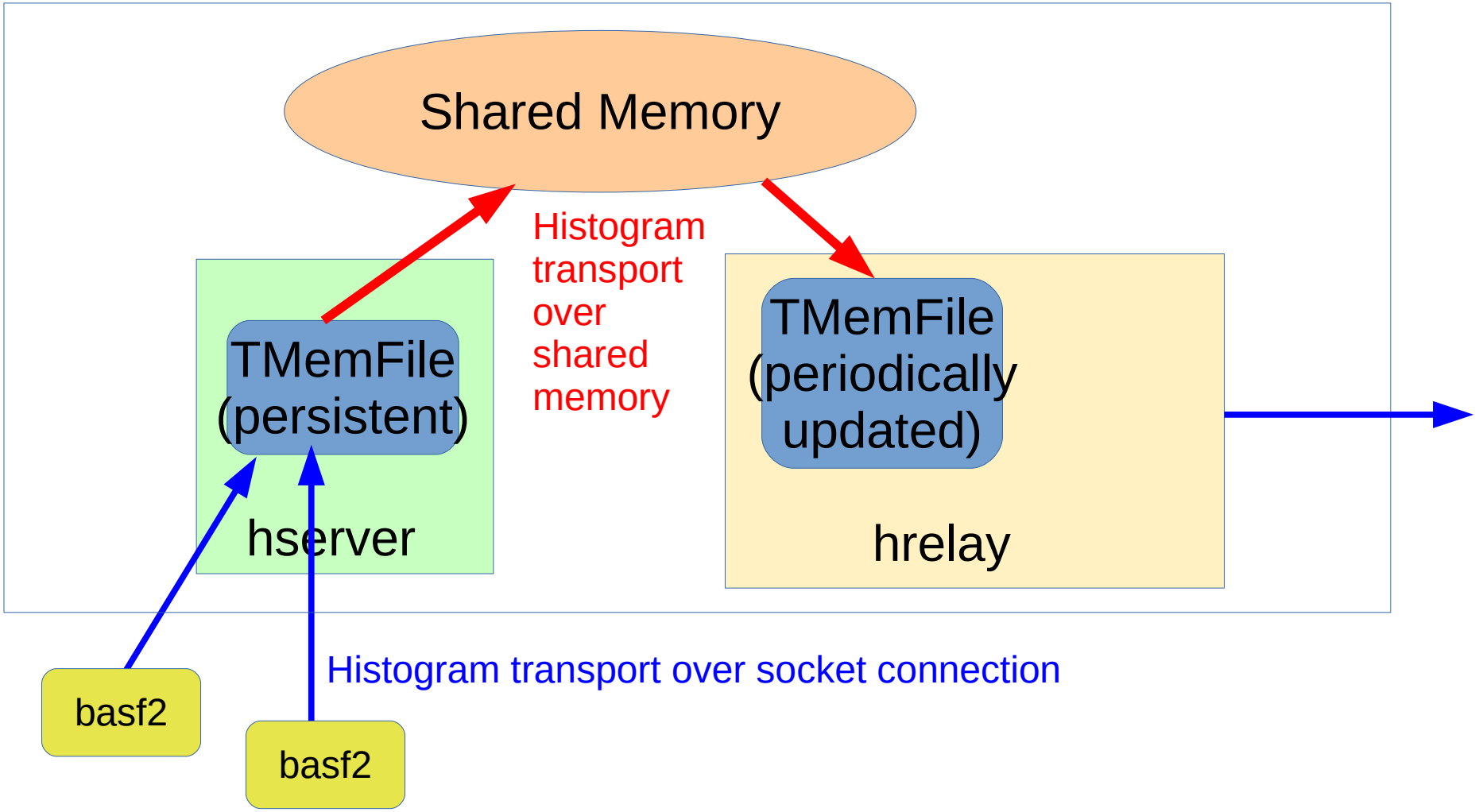
- The definition of all ROOT histograms is centralized to this module and the accumulated contents are dumped to network socket from each event process in addition to files.

- The dump is done at every preset event number interval.

- Only 1D/2D histograms are transferred to hserver at the dump while all histograms/tuples/trees are dumped in files.

- At the run end, the histograms/ tuples/trees files at last dump are collected and merged to a single file/ HLT unit.
  -> to be used for the constants-making.

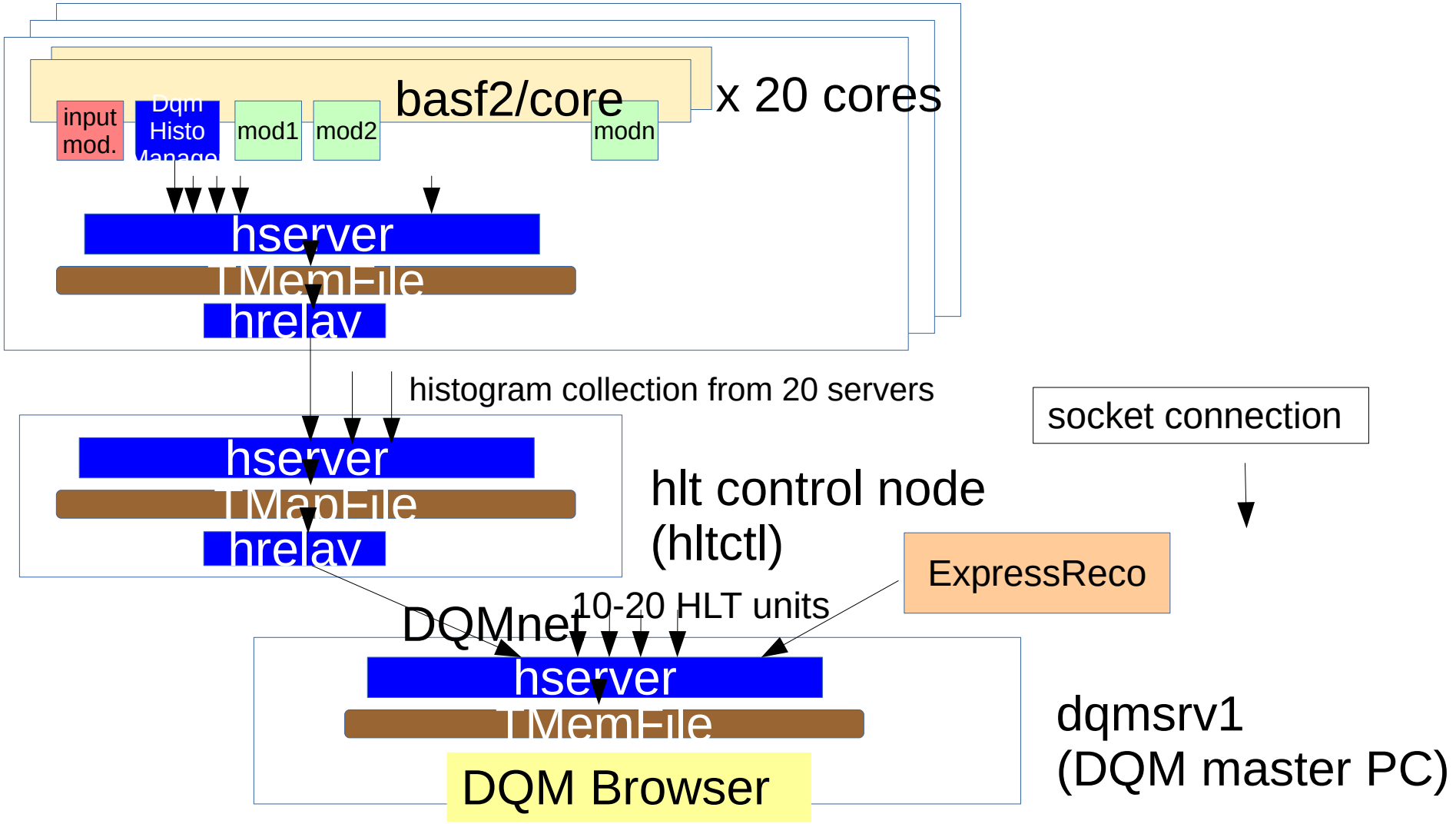
Histogram transport with TMemFile



- Former TMapFile was replaced with this new method.

Step 2 and 3) Collection of histograms via network

- Implemented by the repeated use of hserver and hrelay.



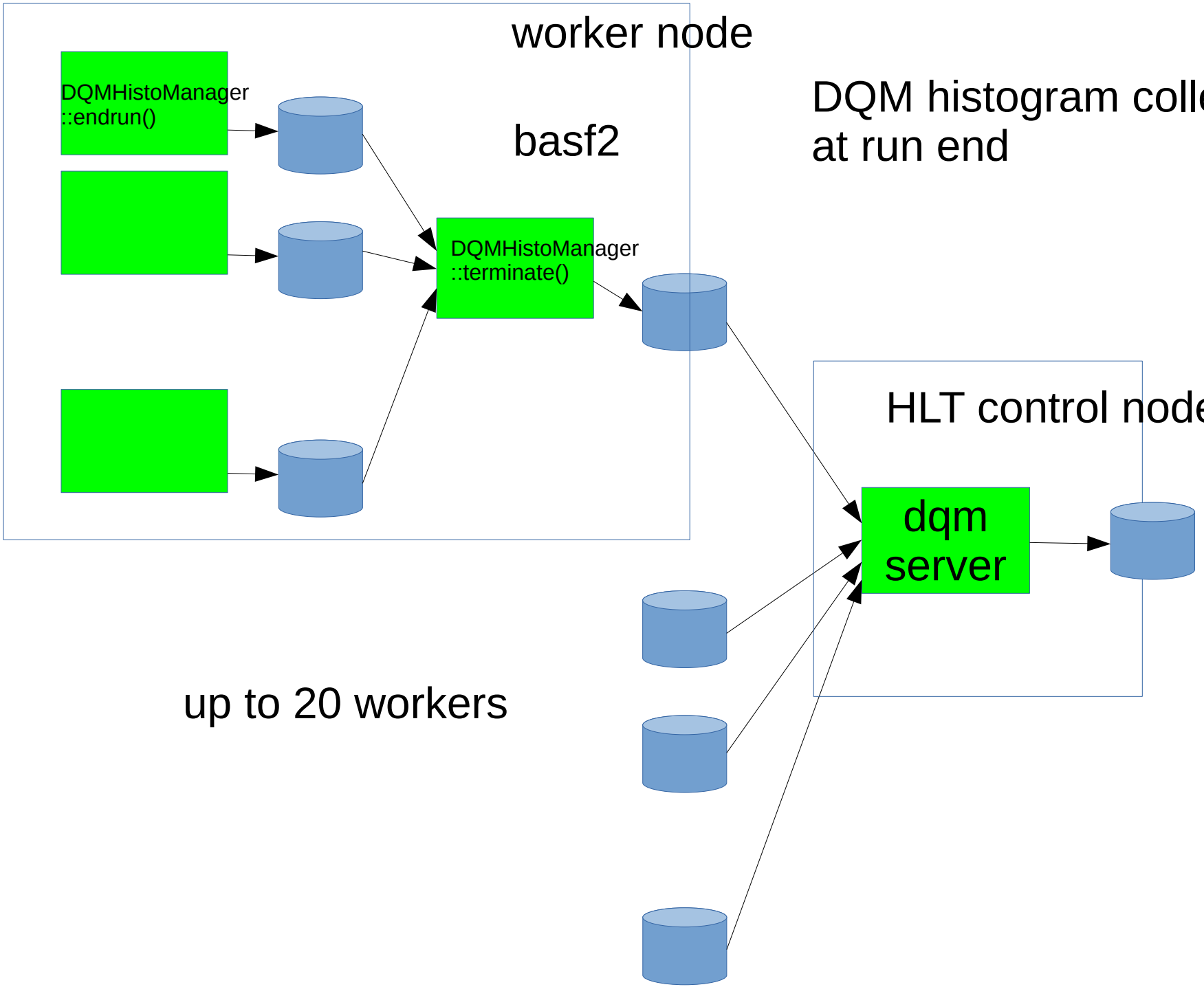
Histogram file writing at run end

- At each run end, the DQM histograms are supposed to be collected and stored in a file.

- In the previous implementation, the collection is done separately from the live DQM collection to manage not only 1D/2D histograms but also TTrees/TNTuples.

- The collection is done in 3 steps.

- When a run is stopped,
  1) basf2 is terminated by sending signal and the histograms accumulated in each event process are written to files corresponding to the processes in DqmHistoManager::endrun().
  2) The histogram files are collected and added in one file per worker node in DqmHistoManager::terminate().
  3) The control nodes collects the files on worker nodes and add them into a single file.


worker node
DQMHistoManager
::endrun()
basf2
DQMHistoManager
::terminate()
DQM histogram collection
at run end
HLT control node
dqm
server
up to 20 workers

- This mechanism includes three file writing. The number of DQM histograms in phase 3 was around 7500 and it took a long time to go through all 3 steps.

  -> Up to 5 minutes to stop a run!

- Several reasons.

  1) The files were first placed in NFS filesystem shared by all worker nodes on a GbE network.

     -> Changed to local file system, but still ~3min or so.

  2) The number of histograms was reduced to 1/3. But still took a few minutes to stop.

- Finally gave up to use this mechanism to leave DQM histograms in a file.

- Instead, the histograms collected by live histogram transport are stored in a file at run end. “DQMMASTER”

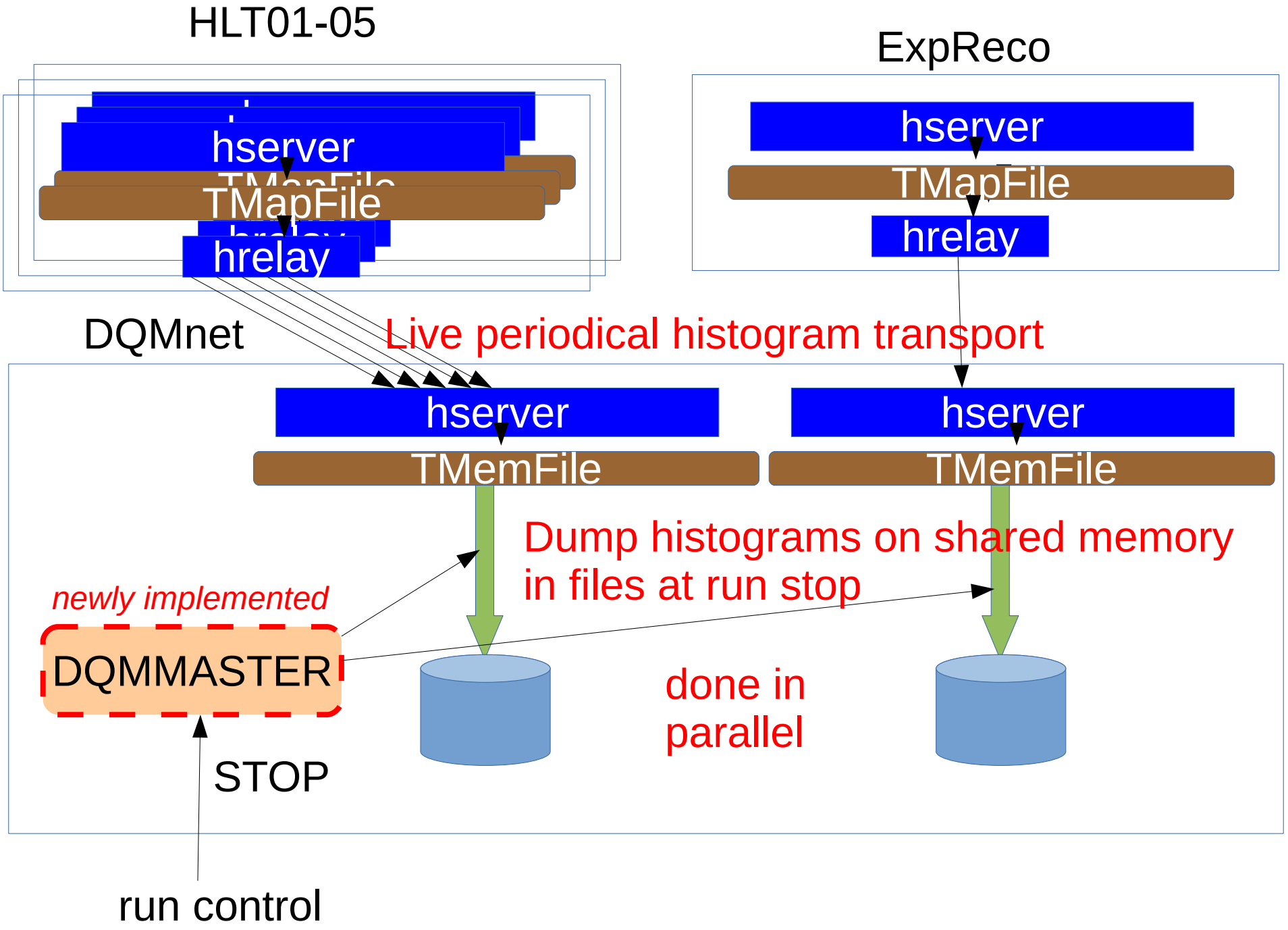
- Number of histograms has been reduced down to 2800 (thanks to the effort by TRG group), and the time for HLT stop was reduced to 2min. or so. But still long.

- Final solution:
  * To give up the histogram collection at each run end.
  * Instead, store the live histograms already transferred to the main node at run end.

  * For the purpose, a new NSM node called “DQMMASTER” is installed.
  * It receives the STOP signal from run control master and dump histograms on TMemFile into files.

  * It was implemented during last acc. maintenance day and now being operated.

    -> The stopping time reduced to ~30 sec.


HLT01-05
hserver
TMapFile
hrelay
ExpReco
hserver
TMapFile
hrelay
DQMnet
Live periodical histogram transport
hserver
TMemFile
hserver
TMemFile
Dump histograms on shared memory
in files at run stop
newly implemented
DQMMASTER
done in
parallel
STOP
run control

- The histogram files are copied to
  qasrv01:/data1/dqm/dqmsrv1/e0008/dqmhisto/
  hltdqm_e0008rxxxxxx.root, and
  erecodqm_e0008rxxxxxx.root.

- The directory structure of the file is different from previous one.
  * No subdirectories corresponding to subsystem.
  * Instead, the subsystem labels are attached to histogram titles.

- The histogram files are copied to kekcc by Boqun
  kekcc:/group/belle2/phase3/dqm/dqmsrv1/e0008/dqmhisto/.

- One Note:

  * In this implementation, we cannot guarantee that “all the events” are in the stored DQM files. Only up to the events where the histograms are transferred to the main node. (Events in a last few minutes before STOP are not accumulated in histograms)
  * Should be OK, since it is a kind of “snapshot” of online DQM.
  * For the detailed study, you can check full events in offline.