Overview and Tests

ØMQ

2019, August 27th

Nils Braun | IETP - KIT

WHY A NEW DATA TRANSPORTATION SCHEMA?

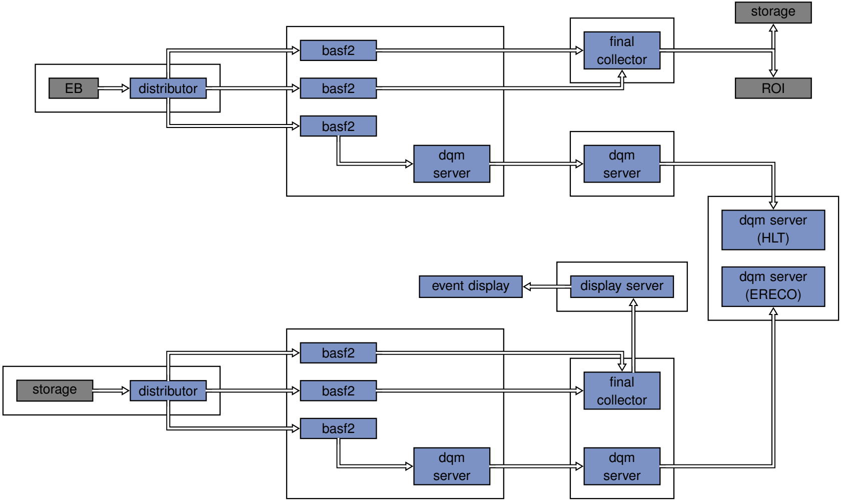
The current data transportation schema based on ring buffer works now very stable!

Still, we want some additional features:

- not possible to sent signals, e.g. run stop etc.
    - therefore: not easy to implement needed features like load geometry on startup, quick abort
- problems with residual state after abortion, sometimes cold restart needed (unpredictable)
- (sometimes) not all histograms can be stored at run end to have short stopping times
- (sometimes) events are missing (reported by PXD)

Important: It is maybe also possible to include the features we want in the current schema - however we discussed to use a more modern schema based on ØMQ.

OVERVIEW (A BIT SIMPLIFIED)



FURTHER INFORMATION

- basf2: `feature/zmq-on-hlt` and daq_slc: `feature/add-new-zmq-hlt-apps`
  - Smaller PRs are currently being reviewed/produced
- Slides in DAQ meetings and sent via DAQ mail list (many things here are taken from those slides)
- technical overview: `https://confluence.desy.de/display/BI/HLT+Data+Transportation+with+ZMQ`
- operations overview (currently being built): `https://confluence.desy.de/display/BI/ZMQ+HLT+operations`

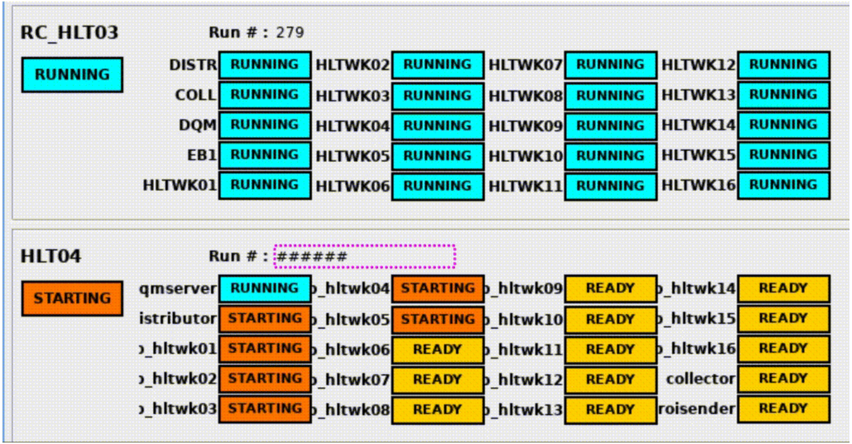
SETUP

- All tests performed with HLT03 on new framework (this means all numbers refer to one HLT unit!)
- `basf2` with ØMQ applications and conditions DB from `cvmfs`
- ~~`daq_slc` from home folder (so far) of `phystrig` user (allows faster turnaround)~~ Is now also on `cvmfs`
- My own restart scripts
- Most of the time: random poisson trigger and CDC included
    - Not really near to reality, but most of the results can be transferred
    - Did also tests with PXD and cosmics (see below)
- `hltwk10` behaves very strange (it is 10 times slower than all other workers, unrelated to ØMQ or ring buffers).
    - I excluded it basically for all tests with reconstruction (with passthrough it does not matter)
- I also tested `ERECO` with the ØMQ framework (not shown here, as it will not be used right now)

CORE FUNCTIONS WORK

- Runs with ”normal” rates (7 kHz with reconstruction and 12 kHz with passthrough on one HLT unit) very smoothly
  - These are actually really high rates for a single unit (compare: we expect 30 kHz with 20 units for normal data taking), but there is also basically no data content
  - At higher rates the storage was the limiting factor
- Many SALS and Start-Stop performed
- Longest run was approx 24h with 2 kHz
- DQM Histograms, ROI sending (also with very high rates), storage and cosmics run were tested

CSS



RC_HLT03 Run # : 279

RUNNING

DISTR	RUNNING	HLTWK02	RUNNING	HLTWK07	RUNNING	HLTWK12	RUNNING
COLL	RUNNING	HLTWK03	RUNNING	HLTWK08	RUNNING	HLTWK13	RUNNING
DQM	RUNNING	HLTWK04	RUNNING	HLTWK09	RUNNING	HLTWK14	RUNNING
EB1	RUNNING	HLTWK05	RUNNING	HLTWK10	RUNNING	HLTWK15	RUNNING
HLTWK01	RUNNING	HLTWK06	RUNNING	HLTWK11	RUNNING	HLTWK16	RUNNING

HLT04 Run # : ######

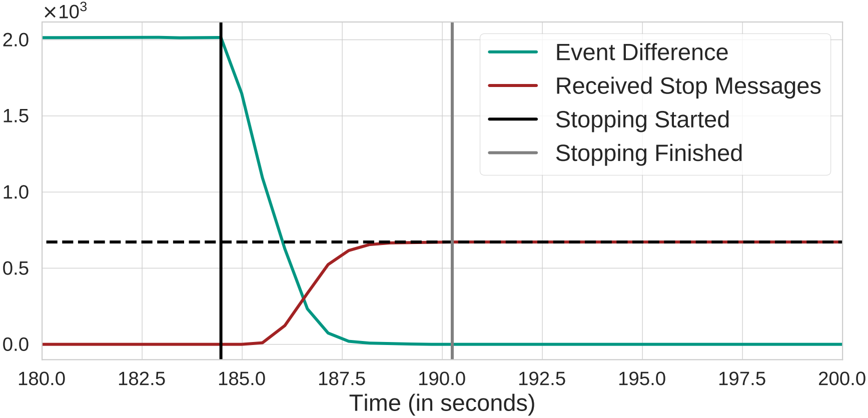
STARTING

qmserver	RUNNING	_hltwk04	STARTING	_hltwk09	READY	_hltwk14	READY
istributor	STARTING	_hltwk05	STARTING	_hltwk10	READY	_hltwk15	READY
_hltwk01	STARTING	_hltwk06	READY	_hltwk11	READY	_hltwk16	READY
_hltwk02	STARTING	_hltwk07	READY	_hltwk12	READY	collector	READY
_hltwk03	STARTING	_hltwk08	READY	_hltwk13	READY	roisender	READY

MONITORING

- Simon has already built a very nice CSS panel for the implementation (image on next page)
- The framework-own monitoring with `b2hlt_monitor.py` works and is really useful for understanding the behavior
  - If you do not know what `b2hlt_monitor.py` is, now would be a good time to have a look into the documentation :-) `https://confluence.desy.de/display/BI/ZMQ+HLT+operations`
  - It can also write out data files, which I use for plotting (jupyter notebook for this in the repo)
  - There is now also a unit overview resembling the former `nodedump` (but for all machines at once)
- Some of the monitoring is also exported via NSM variables (not heavily tested, but works so far), could be used for CSS variables (although the storage rate is already enough)

RUN STOP


×10^3
2.0
1.5
1.0
0.5
0.0
180.0
182.5
185.0
187.5
190.0
192.5
195.0
197.5
200.0
Time (in seconds)
Event Difference
Received Stop Messages
Stopping Started
Stopping Finished

STOPPING

- One requirement of the new implementation was that all events are transported
- Instead of closing the distributor on stop immediately, I wait until there is a zero event rate for 5 second (configuration parameter). Same for collector.
  - This means even after stopping such a run, events flow for several seconds (even up to one minute or longer)
- It was validated that:
  - Number of events at EB1 equals to the number of events processed by hltin and hltout
  - Number of entries in the event number counter of the DQM histograms is also the same
  - ~~Number of events at storage is the same~~ **FAILED!**

Important Note: At some point someone needs to debug the storage nodes to find out why!

STOP AND START AGAIN

Especially at high(er) rates, starting the next run after stopping leads to a strange incident:

- few events of the last run (8-12) still reach the EB1 and are given to the HLT before the events from the new run come
- Master detective Yamagata-san found out after some longer investigation: events remain in the FIFO of the COPPERS (this time it was CDC, do not know about the others) because stop does not lead to a flush
- This breaks some of my assumptions, so I needed to change parts of the code
    - Now it is not a technical problem for HLT anymore to receive mixed-run events, but it is still a but ”strange” (and has probably some second-order influence)
    - Can every system handle those additional events (I think storage dismisses them?)
    - What is our policy with those events?

OPEN TASKS AND OUTLOOK

- Pull requests (me)
- Fall-back abortion: kill the applications brutally if nothing else works (me?)
- Finally write the restart scripts in Oskars framework (Oskar, me)
- Eventually also tests with ”really real” data (collisions with all detectors) (. . .)
- Include ØMQ HLTs in global run control (Yamagata-san, Itoh-san)
  - Once everything is prepared: use common `cvmfs`
- Understand storage (someone?)
- Plan: HLT06-HLT09 will run with ØMQ during the upcoming data taking in fall