Belle II Summer Workshop - 2022, Iowa State University, Ames, IA

Belle II Data Acquisition (DAQ)

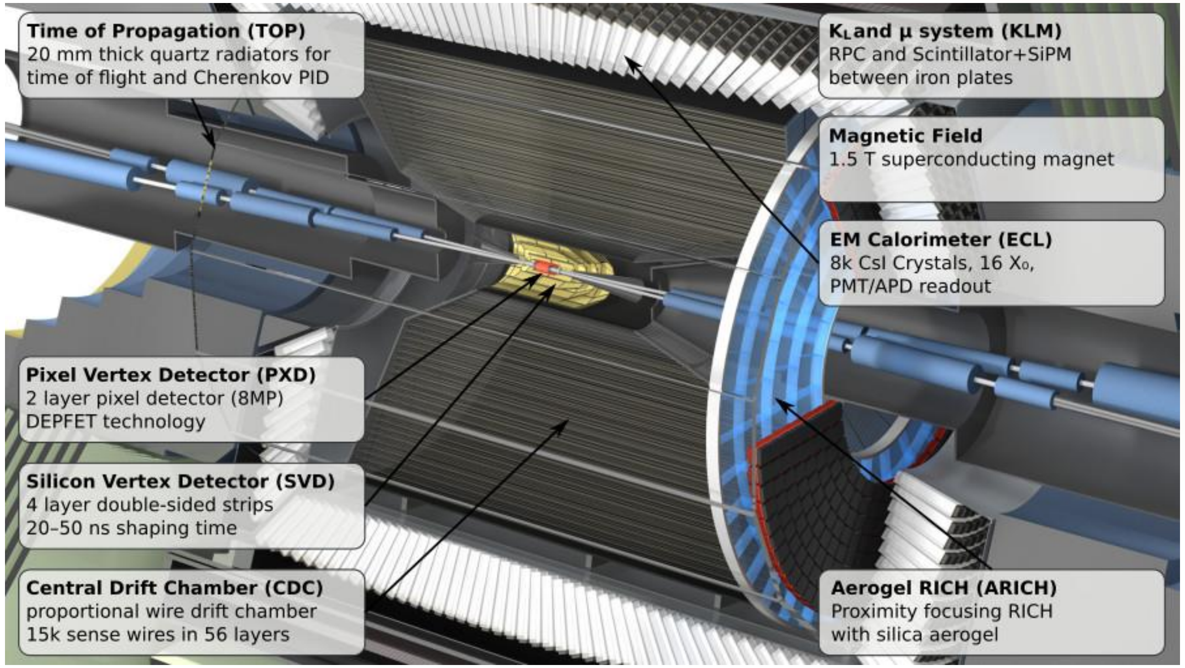
HARSH PURWAR

University of Hawaii at Mānoa (UHM), High Energy Physics Group
Department of Physics & Astronomy, Honolulu, HI, USA
Email: purwar@hawaii.edu

Outline of the talk

Harsh Purwar, HEPG, UHM, Honolulu, HI, USA
August 5th, 2022

- Belle II DAQ setup
- Detector initialization
- Run control GUIs, NSM
- Data flow in Belle II DAQ
- Data monitoring
- Planned improvements

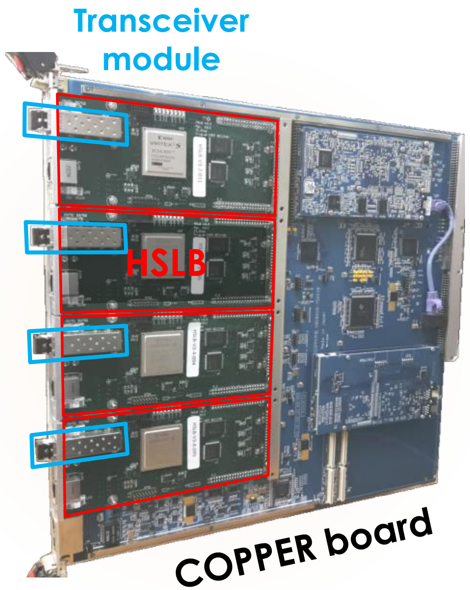


Harsh Purwar, HEPG, UHM, Honolulu, HI, USA
August 5th, 2022

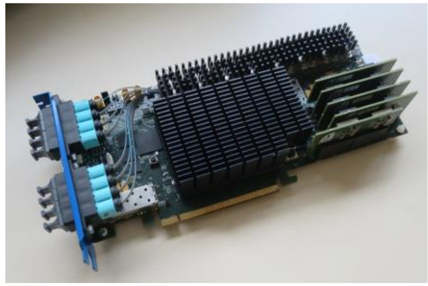
Belle II DAQ setup

Until 2021b

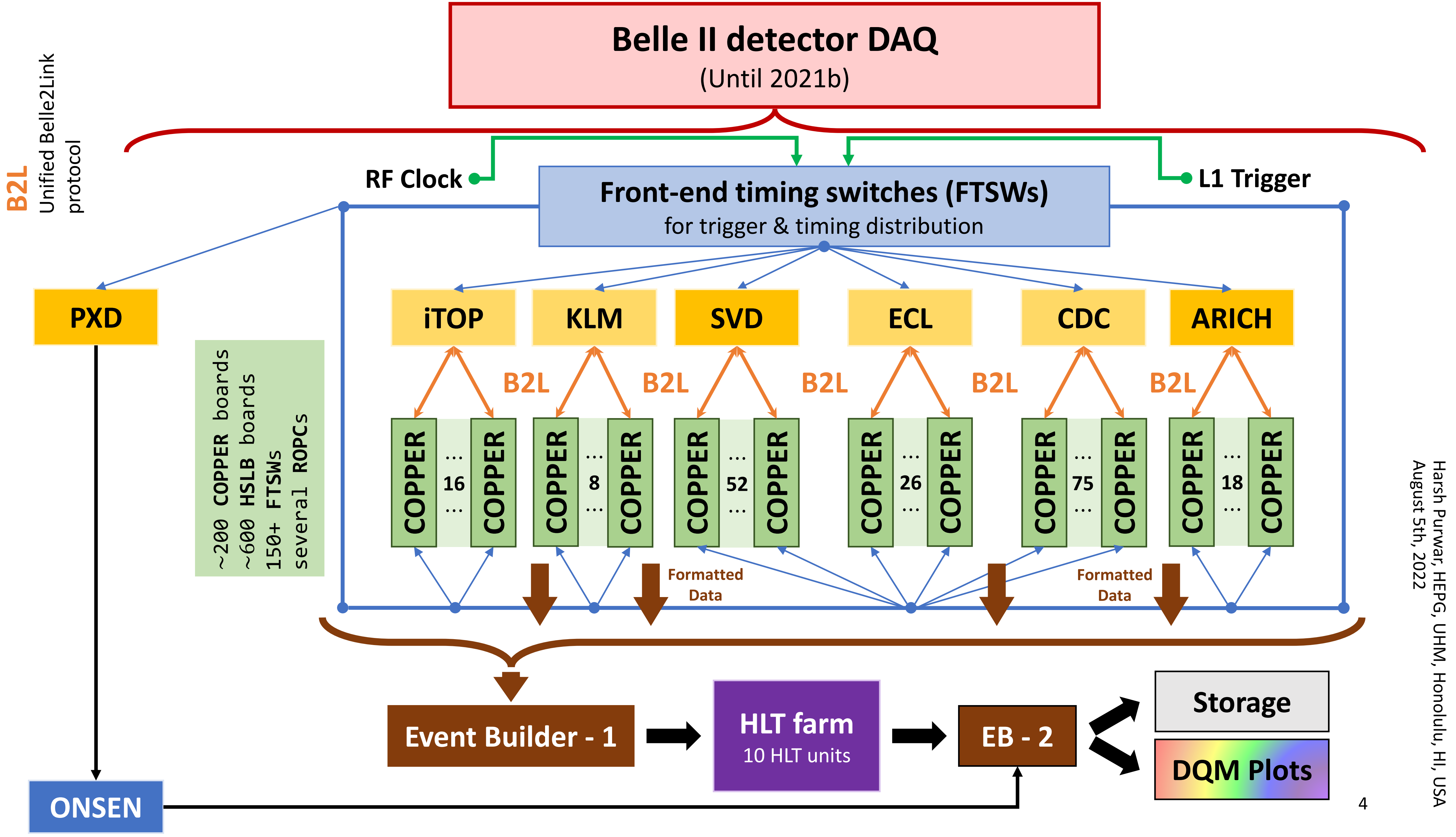
- Common unified readout system for all sub-detectors (except PXD)
  - COPPER – Common Pipelined Platform for Electronics Readout
  - HSLB – High-speed Link Board
  - All COPPER and HSLB boards are now replaced with PCIe40 board
- Unified timing & trigger distribution system (with FTSWs) for all FEEs and readout boards.
- **During LS1**, several improvements are planned to improve DAQ stability and reduce downtime.

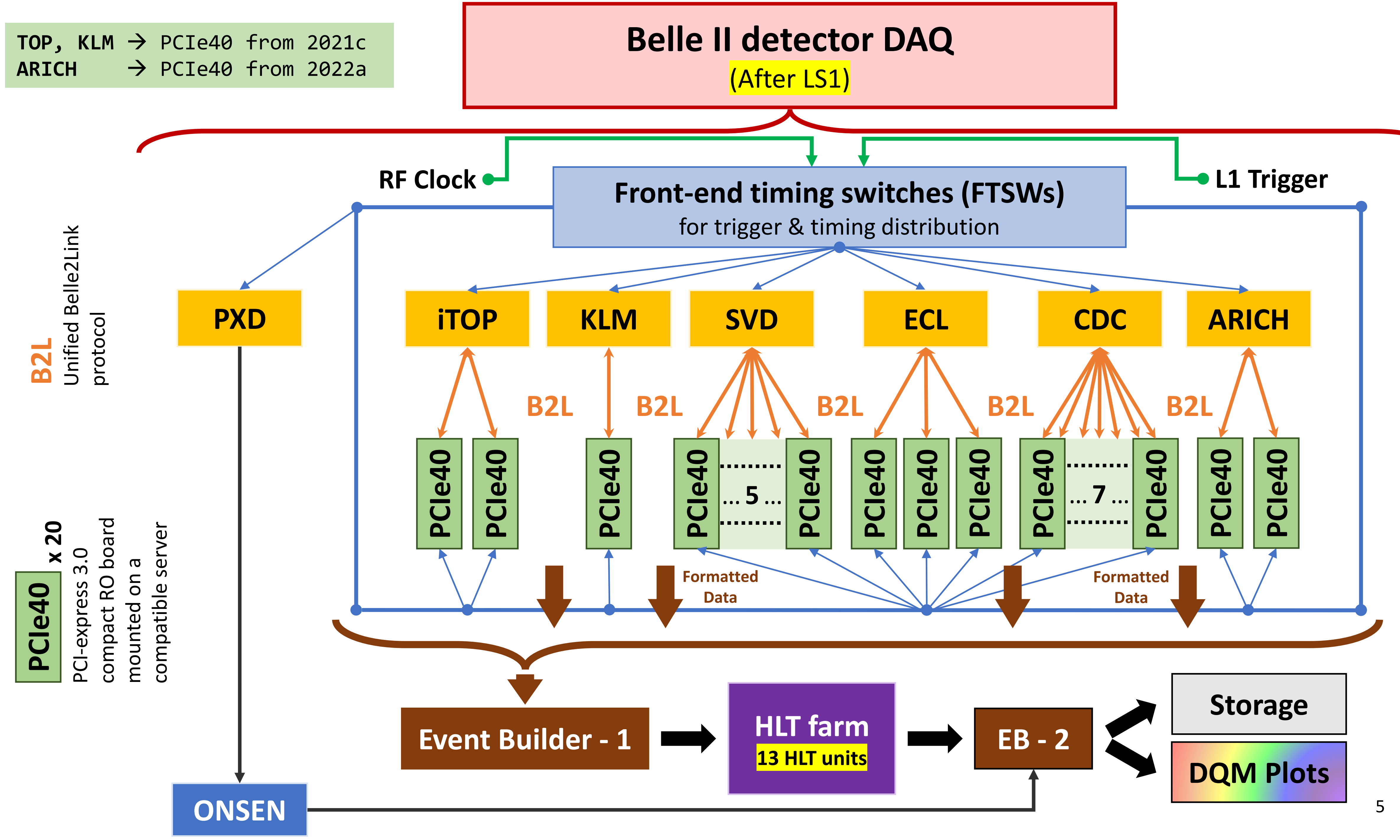


COPPER board



PCIe40 board


Belle II detector DAQ
(Until 2021b)
B2L
Unified Belle2Link protocol
RF Clock
L1 Trigger
Front-end timing switches (FTSWs)
for trigger & timing distribution
PXD
iTOP
KLM
SVD
ECL
CDC
ARICH
~200 COPPER boards
~600 HSLB boards
150+ FTSWs
several ROPCs
B2L
COPPER
16
8
52
26
75
18
Formatted Data
Event Builder - 1
HLT farm
10 HLT units
EB - 2
Storage
DQM Plots
ONSEN
Harsh Purwar, HEPG, UHM, Honolulu, HI, USA
August 5th, 2022



Harsh Purwar, HEPG, UHM, Honolulu, HI, USA
August 5th, 2022

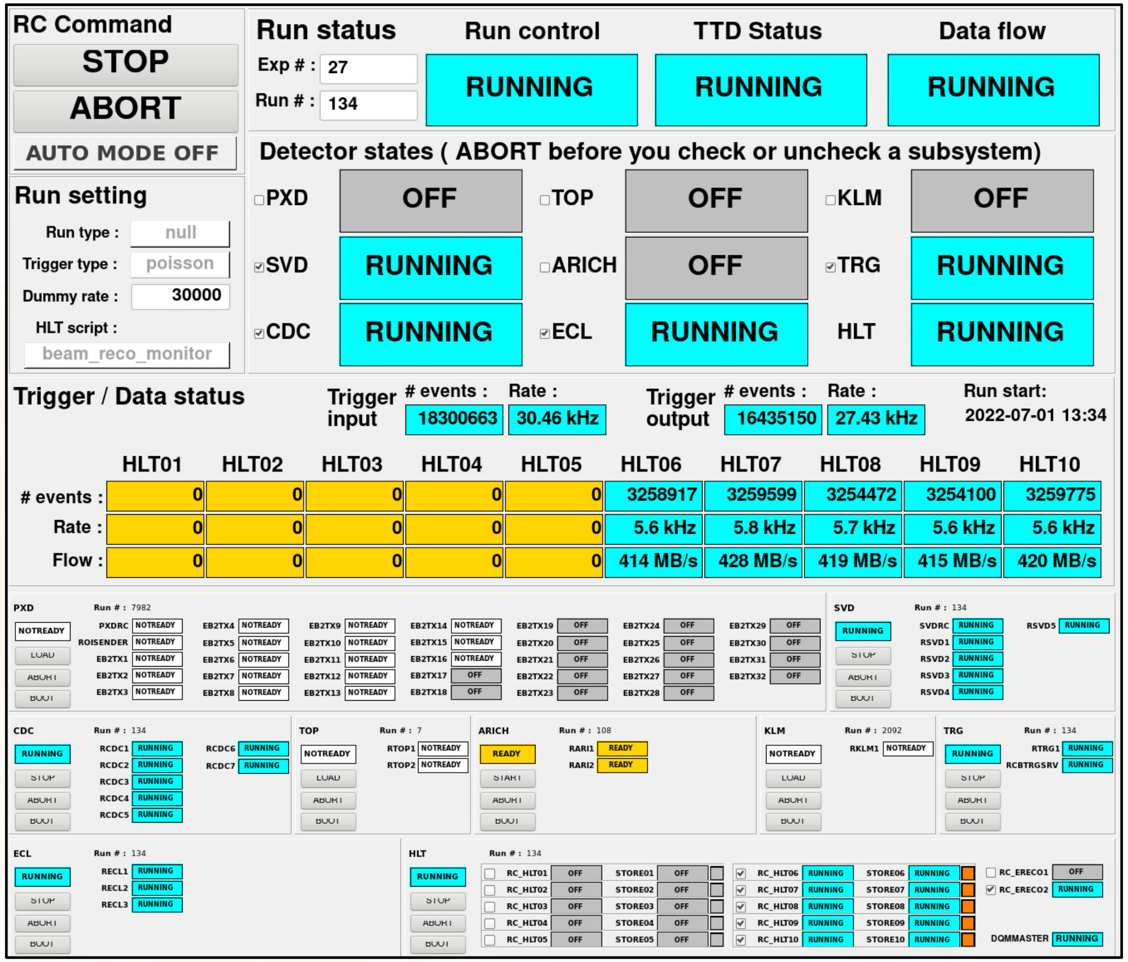
Detector initialization

(Detector slow control)

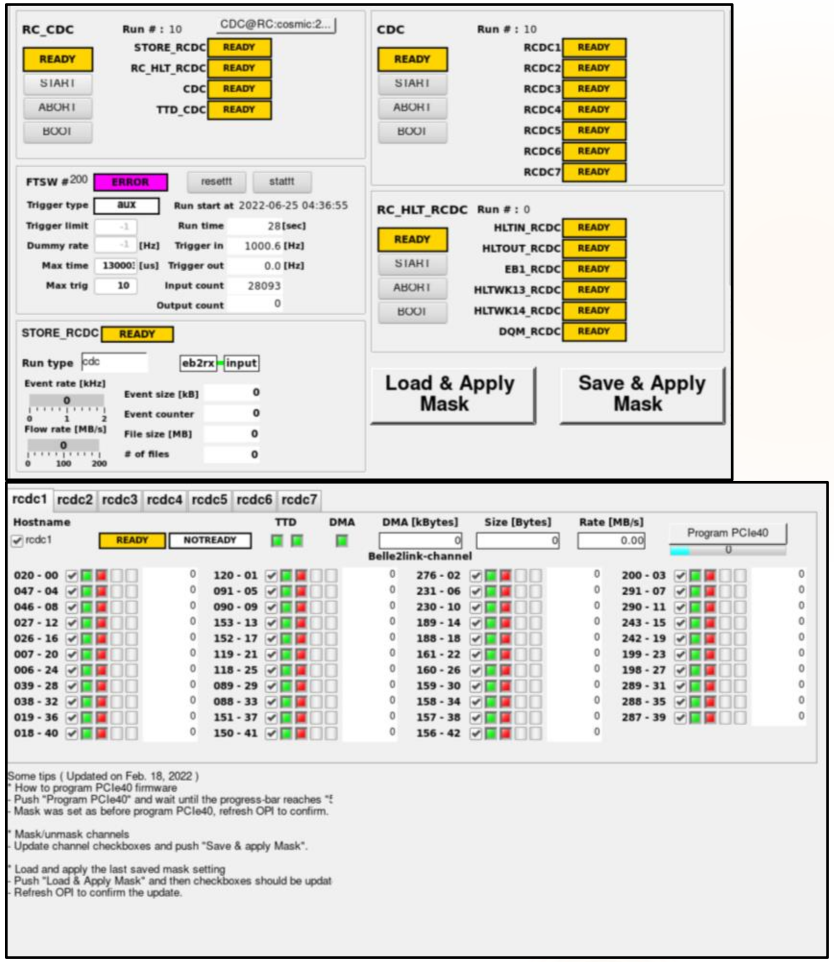
- Before reading good sensible data, the sub-detector FEs need to be initialized or configured, also referred as detector slow control (SLC).
  - Setting thresholds, window sizes, pedestals, mode of operation (e.g., raw or suppress modes for CDC), etc.
  - Basically, any detector specific setting required for correct data readout
- SLC also includes detector monitoring – temperature, humidity, voltage, etc.
- *Happens through the readout board (PCIe40), which interacts with the FEE over the same **B2L** (or optical link).*
- **daq_slc**: The slow control library for the full and consistent configuration of almost all the sub-detectors. https://stash.desy.de/projects/B2DAQ/repos/daq_slc/browse

Global & Local run control GUIs

Global Run Control GUI



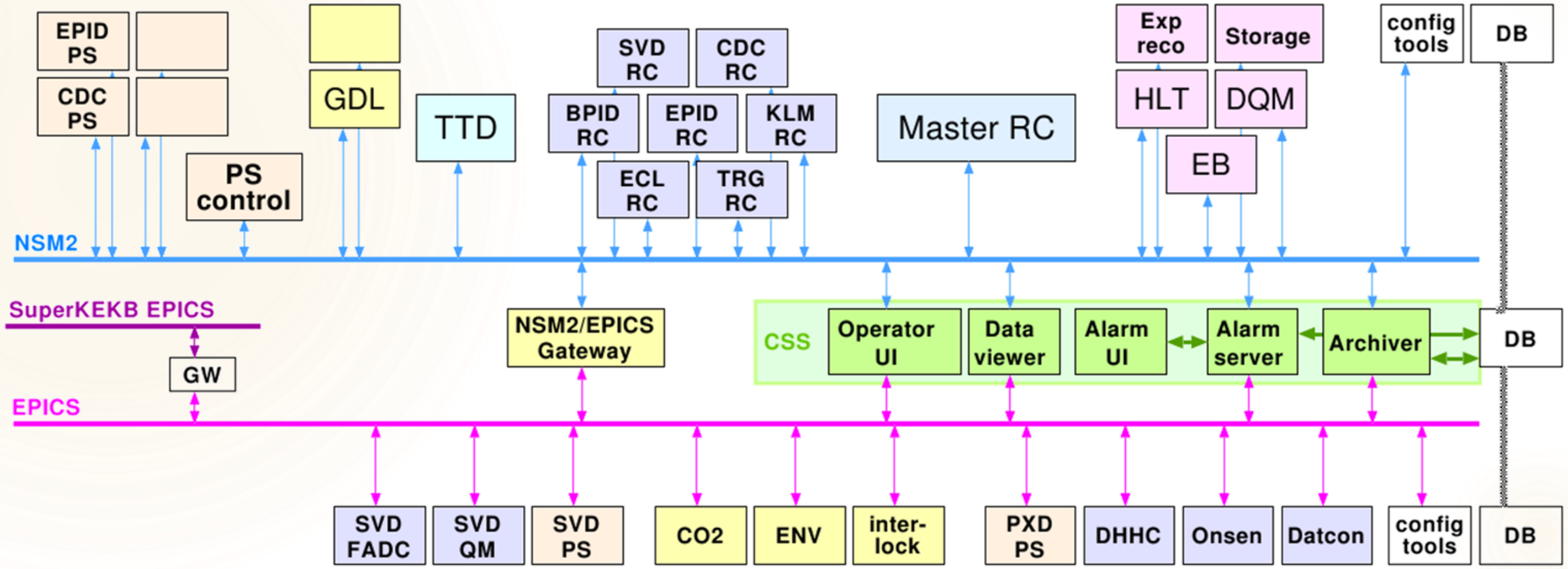
CDC Local Run Control GUI



Harsh Purwar, HEPG, UHM, Honolulu, HI, USA
August 5th, 2022

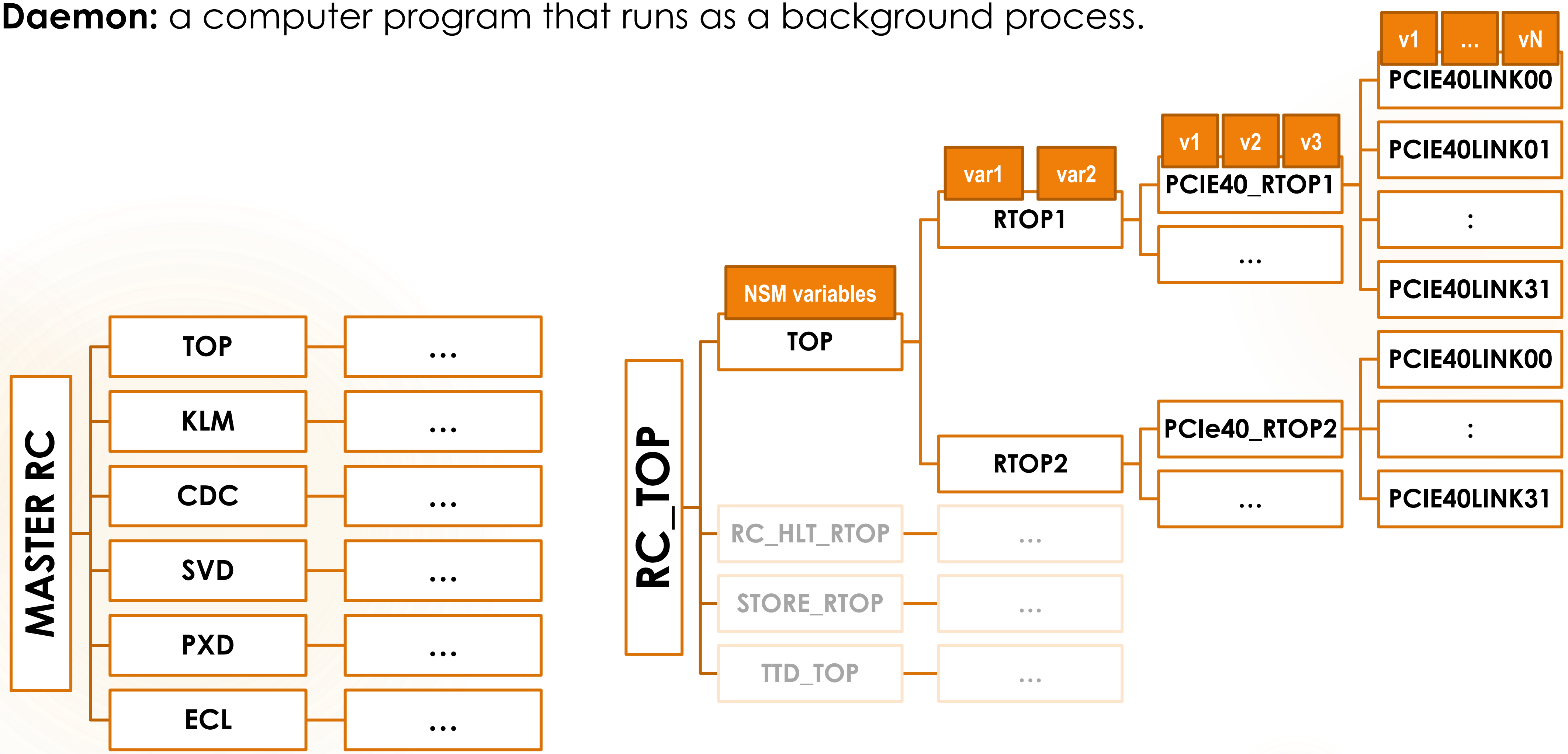
Network shared memory (NSM2)

- All ROPC (`rtopX`, `rklmX`, etc.), all HV monitoring servers, all HLT units, storage servers, etc. – basically every computer that is included in DAQ are all connected in a closed "trusted" network.
- Each computer can share information with any other machine within this closed network through NSM, EPICS PVs and DB (all included in **daq_slc**).
- Several 100s of NSM nodes exist within the network to facilitate SLC communication.
- **daq_restart**: Shell scripts to start/stop these NSM nodes. Extremely simple usage. https://stash.desy.de/projects/B2DAQ/repos/daq_restart/browse



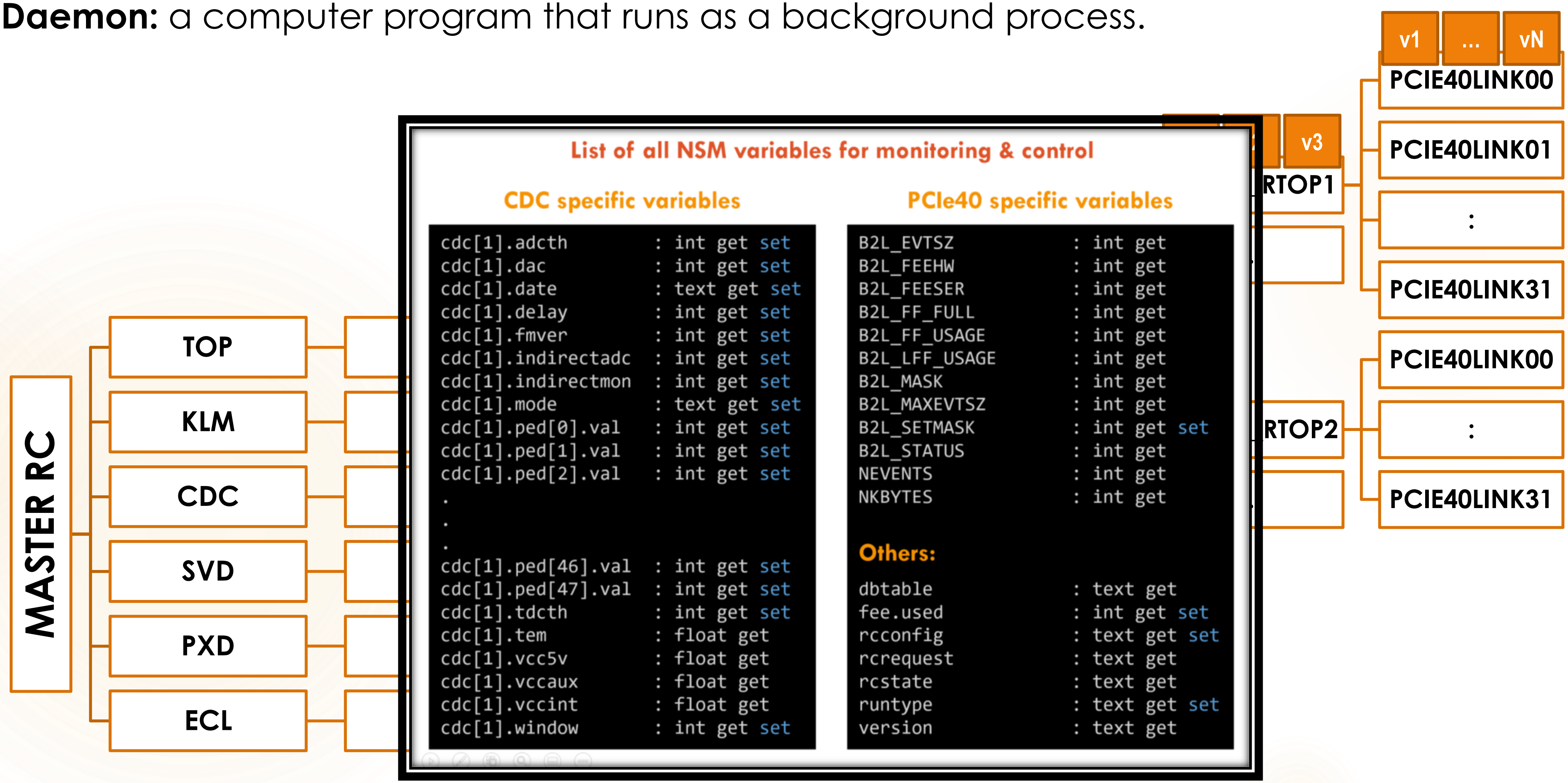
NSM nodes (daemons) & variables

Daemon: a computer program that runs as a background process.



NSM nodes (daemons) & variables

Daemon: a computer program that runs as a background process.



List of all NSM variables for monitoring & control

CDC specific variables

```
cdc[1].adcth          : int get set
cdc[1].dac            : int get set
cdc[1].date           : text get set
cdc[1].delay          : int get set
cdc[1].fmver          : int get set
cdc[1].indirectadc    : int get set
cdc[1].indirectmon    : int get set
cdc[1].mode           : text get set
cdc[1].ped[0].val     : int get set
cdc[1].ped[1].val     : int get set
cdc[1].ped[2].val     : int get set
.
.
.
cdc[1].ped[46].val    : int get set
cdc[1].ped[47].val    : int get set
cdc[1].tdcth          : int get set
cdc[1].tem            : float get
cdc[1].vcc5v          : float get
cdc[1].vccaux         : float get
cdc[1].vccint         : float get
cdc[1].window         : int get set
```

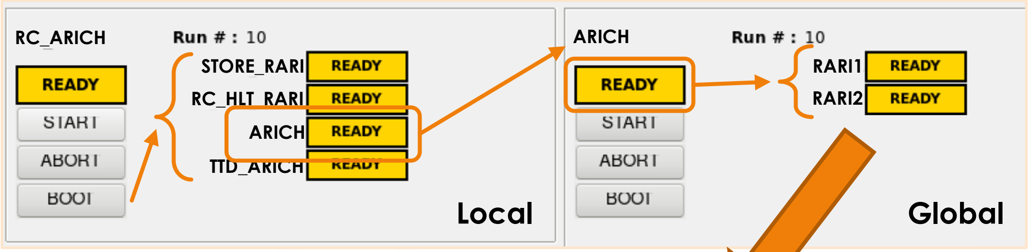
PCIe40 specific variables

```
B2L_EVTSZ             : int get
B2L_FEEHW             : int get
B2L_FEESER            : int get
B2L_FF_FULL           : int get
B2L_FF_USAGE          : int get
B2L_LFF_USAGE         : int get
B2L_MASK              : int get
B2L_MAXEVTSZ          : int get
B2L_SETMASK           : int get set
B2L_STATUS            : int get
NEVENTS               : int get
NKBYTES               : int get
```

Others:

```
dbtable               : text get
fee.used              : int get set
rcconfig              : text get set
rcrequest             : text get
rcstate               : text get
runtype               : text get set
version               : text get
```

NSM nodes & callback functions

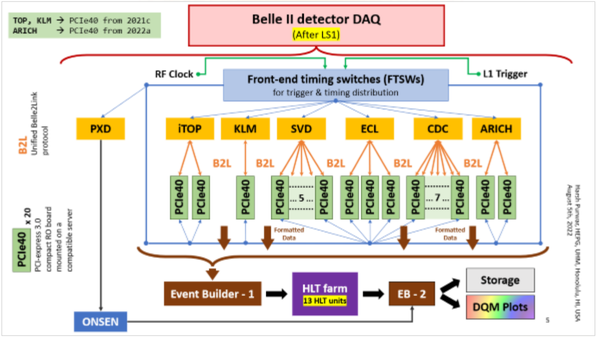



```
void ARICHPcie40FEE::boot(RCCallback& callback, B2LINK& b2link, const DBObject& obj)
{
  const std::string vname = StringUtil::form("arich[%d].", b2link.get_link()+baseid);
  int used[6] = { 1, 1, 1, 1, 1, 1 };
  for (size_t i = 0; i < 6; i++)
    callback.get(vname + StringUtil::form("feb[%d].used", i), used[i]);
  b2link.monitor();
  m_serial = b2link.get_info().feeser;
  ARICHPcie40Merger mer(callback, b2link);
  LogFile::debug("md_id=%d", m_serial);
  for (size_t i = 0; i < 6; i++) {
    callback.get(vname + StringUtil::form("feb[%d].used", i), used[i]);
    std::string path = StringUtil::form("db://arich/MB:%d:FEB:%d:", m_serial, i);
    m_o_feb[i] = callback.dbload(path);
    if (m_o_feb[i].getName().size() == 0) {
      path = "db://arich/MB:0:FEB:0:";
      m_o_feb[i] = callback.dbload(path);
    }
    const std::string cvname = vname + StringUtil::form("feb[%d].", i);
    path = "db://arich/" + m_o_feb[i].getName();
    callback.set(cvname + "path", path);
    LogFile::debug(path);
  }
  mer.boot(m_obj_merger, m_o_feb, used, m_serial, baseid);
  readback(callback, b2link, m_obj_merger);
}
```

```
void CDCPcie40FEE::init(RCCallback& callback, B2LINK& b2link, const DBObject&)
{
  std::string vname = StringUtil::form("cdc[%d]", b2link.get_link()+baseid);
  LogFile::warning(vname);
  callback.add(new CDCDateHandler(vname + ".date", callback, b2link, *this));
  callback.add(new CDCFirmwareHandler(vname + ".fmver", callback, b2link, *this));
  callback.add(new CDCDataFormatHandler(vname + ".mode", callback, b2link, *this));
  callback.add(new CDCWindowHandler(vname + ".window", callback, b2link, *this));
  callback.add(new CDCDelayHandler(vname + ".delay", callback, b2link, *this));
  callback.add(new CDCADCThresholdHandler(vname + ".adcth", callback, b2link, *this));
  callback.add(new CDCTDCThresholdHandler(vname + ".tdcth", callback, b2link, *this));
  callback.add(new CDCIndirectADCAccessHandler(vname + ".indirectadc", callback, b2link, *this));
  callback.add(new CDCDACControlHandler(vname + ".dac", callback, b2link, *this));
  callback.add(new CDCIndirectMonitorAccessHandler(vname + ".indirectmon", callback, b2link, *this));
  for (int i = 0; i < 48; i++) {
    std::string vname = StringUtil::form("cdc[%d].ped[%d].val", b2link.get_link()+baseid, i);
    callback.add(new CDCPedestalHandler(vname, callback, b2link, *this, i));
  }
  callback.add(new NSMVHandlerFloat(vname + ".tem", true, false, 0));
  callback.add(new NSMVHandlerFloat(vname + ".vccint", true, false, 0));
  callback.add(new NSMVHandlerFloat(vname + ".vccaux", true, false, 0));
  callback.add(new NSMVHandlerFloat(vname + ".vcc5v", true, false, 0));
}
```

```
void TOPPcie40FEE::monitor(RCCallback& callback, B2LINK& b2link)
{
  std::string m_monitoring="";
  callback.get("top_monitoring", m_monitoring);
  if(m_monitoring=="ON") {
    int id = b2link.get_link();
    map<int, BoardStackStatus>::iterator it = m_statusMonitor.find(id);
    if (it != m_statusMonitor.end()) it->second.UpdateNSMCallbacks(b2link, callback);
    else callback.log(LogFile::DEBUG, StringUtil::form("Boardstack %d not found", id));
    SumSEM(callback, b2link.get_link());
  }
  else return;
}
```

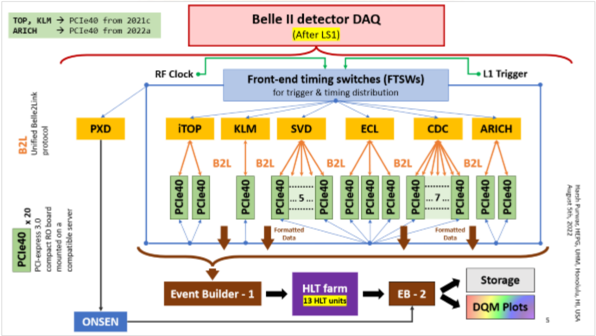
Data flow in Belle II DAQ



30 kHz with <1% deadtime

- Each sub-detector have their own electronics (FEE), that interacts with the sub-detector hardware.
- Raw data (voltage/current signals) from actual detector hardware is readout with detector specific FEE for each event of a run provided there was a L1 trigger issued.
- These incoming electronic signals are then converted to optical signals using a bi-directional optical transceiver module on the detector FEE side.
- The optical signals are then sent out from the detector FEEs (B2Link protocol) via optical fibers to another board, where the data is formatted, and certain checks are also performed by the readout board to ensure no data corruption happened during data transmission.
- All formatted data is then packed using **basf2's *daq*** module on the ROPC and sent out to event builder.

Data flow in Belle II DAQ



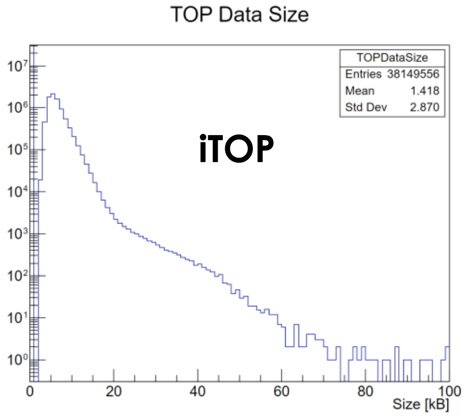
- The data is then passed on to the HLT cluster where the first full event reconstruction with data from all sub-detectors (except PXD) is done using the same basf2 framework.
- Reduced/filtered HLT data (from all 6 subdetectors) + PXD data from ONSEN then goes through the 2nd phase of the event building (10 kHz).
- The data is finally stored on the disks and (a part) is used for generating data quality plots.

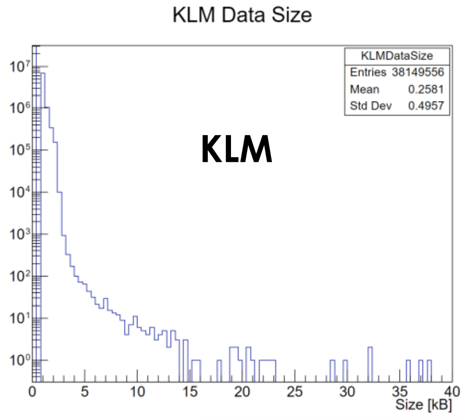
- More details about HLT filtering, etc. in Jake's talk: https://indico.belle2.org/event/6444/sessions/2337/attachments/17913/26632/B2SW22_Bennett_DP.pdf

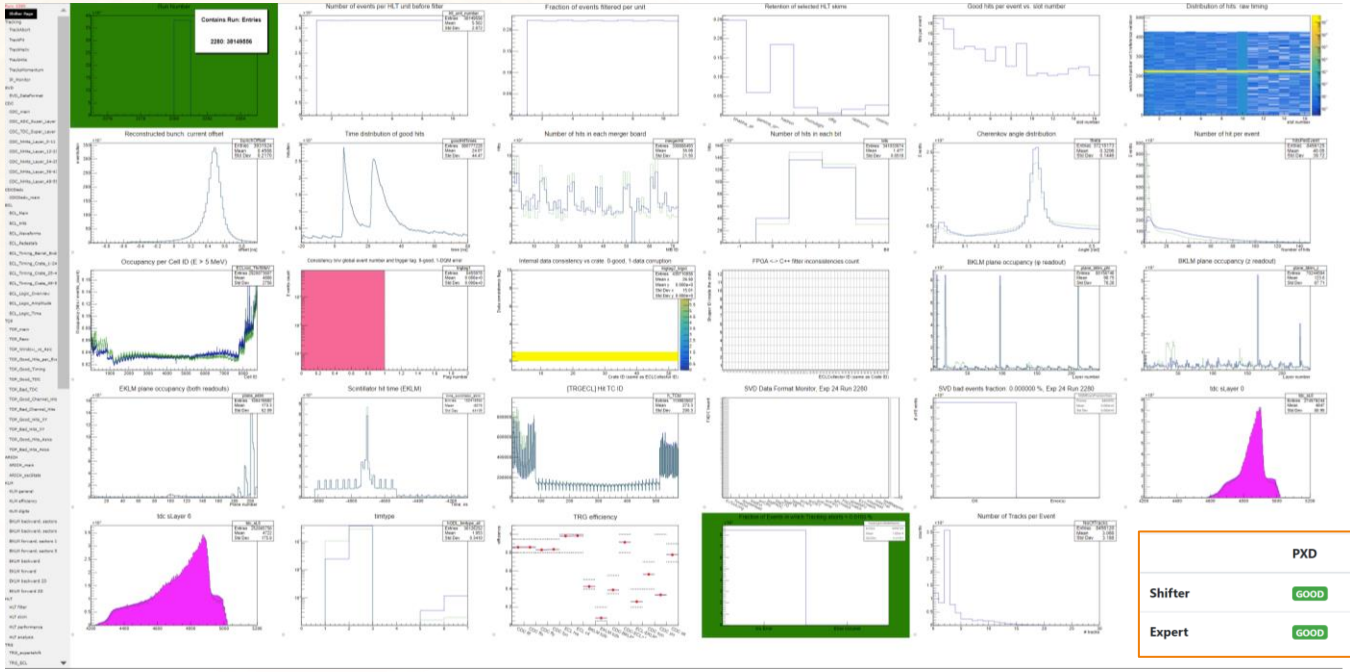
Data quality monitoring

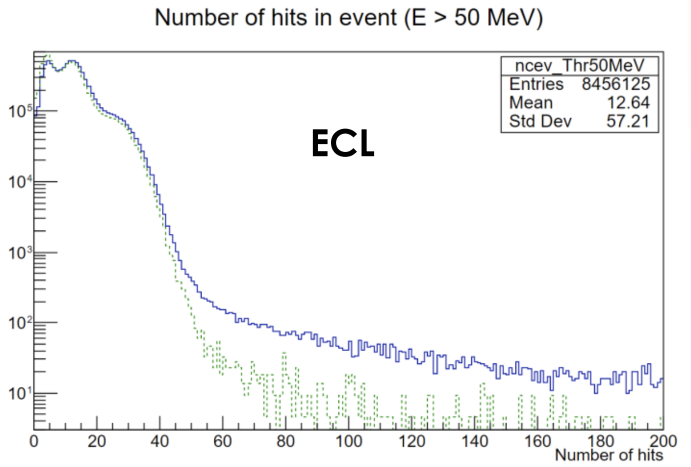
(DQM plots: https://dqm.belle2.org)

- For each sub-detector we have several plots that are used to ensure that the quality of the data being readout is good.
- Most of these plots also show a reference plot for comparison.









	PXD	SVD	CDC	TOP	ARI	ECL	KLM	TRG	RAWDATA
Shifter	GOOD	GOOD	GOOD	GOOD	GOOD	GOOD	GOOD	GOOD	
Expert	GOOD	GOOD	GOOD	GOOD	GOOD	GOOD	GOOD	GOOD	GOOD

Planned improvements

Harsh Purwar, HEPG, UHM, Honolulu, HI, USA
August 5th, 2022

- **Increment in the number of HLT units** from 10 to 13 providing an additional 1500 CPU cores for event reconstruction and data reduction.
- **Auto-recovery** for certain issues to reduce downtime
  - If among one of the known errors: **Pause** ongoing run, **Fix** the issue, **Resume** the run
- FTSW shares TTD info using ethernet (CAT) cables – prone to electronic noise
  - Exploring the option of replacing CAT cables with optical fibres for TTD
- Current PCIe40 throughput (i.e., b/w PCIe40 and ROPC) is **limited to 50 Gbps** on single PCI-express 3.0 lane. Experts are working on using both available (2 x8) PCI-express lanes to increase this throughput to 100 Gbps.
  - Plan on reducing the load on PCIe40 ROPCs by moving data corruption checks to only PCIe40 firmware

I hope this talk gave you an insight on how the Belle II detector operates and how the data that we analyze is recorded.

Thank you for your time and attention.

- **Harsh Purwar**
purwar@hawaii.edu