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

#### Belle II Data Acquisition (DAQ)

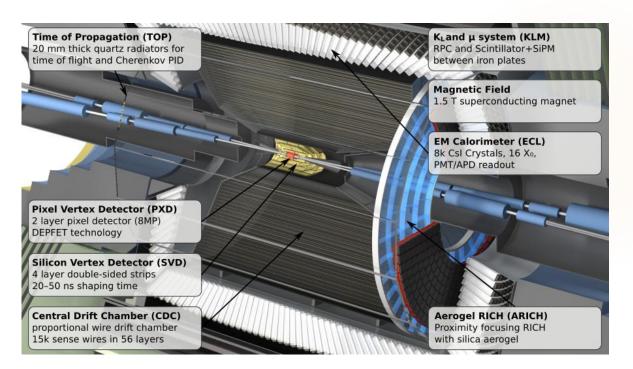
#### HARSH PURWAR

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



### Outline of the talk

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





#### 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.

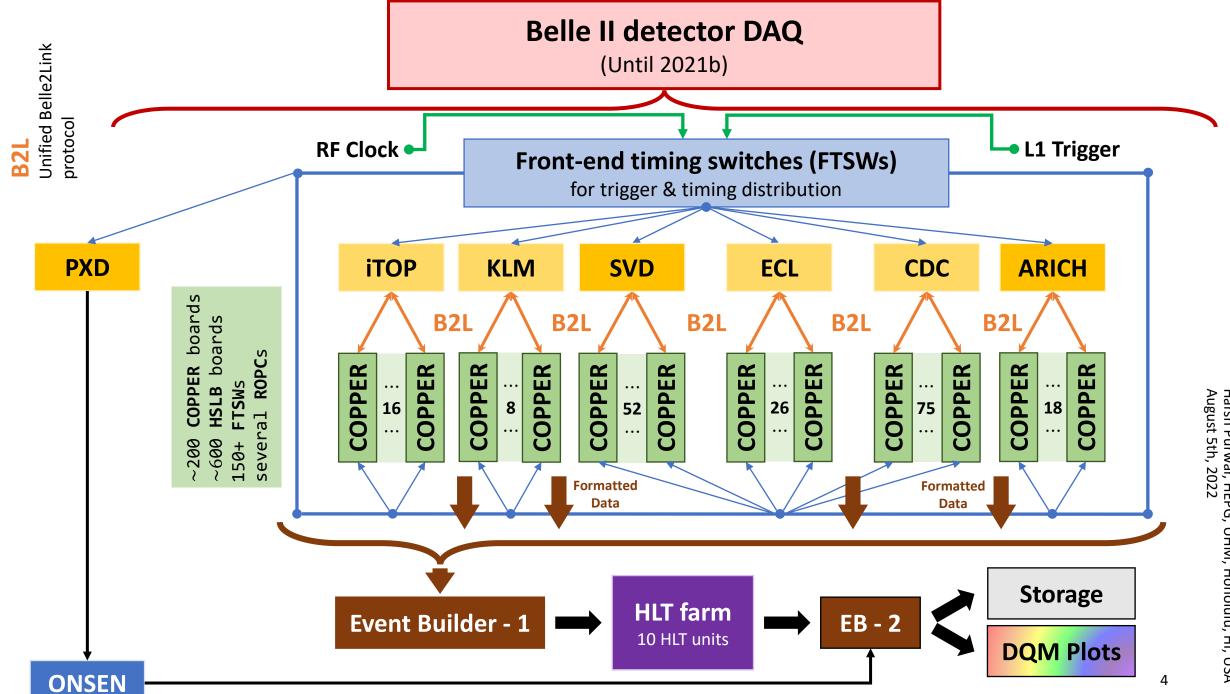


COPPER board

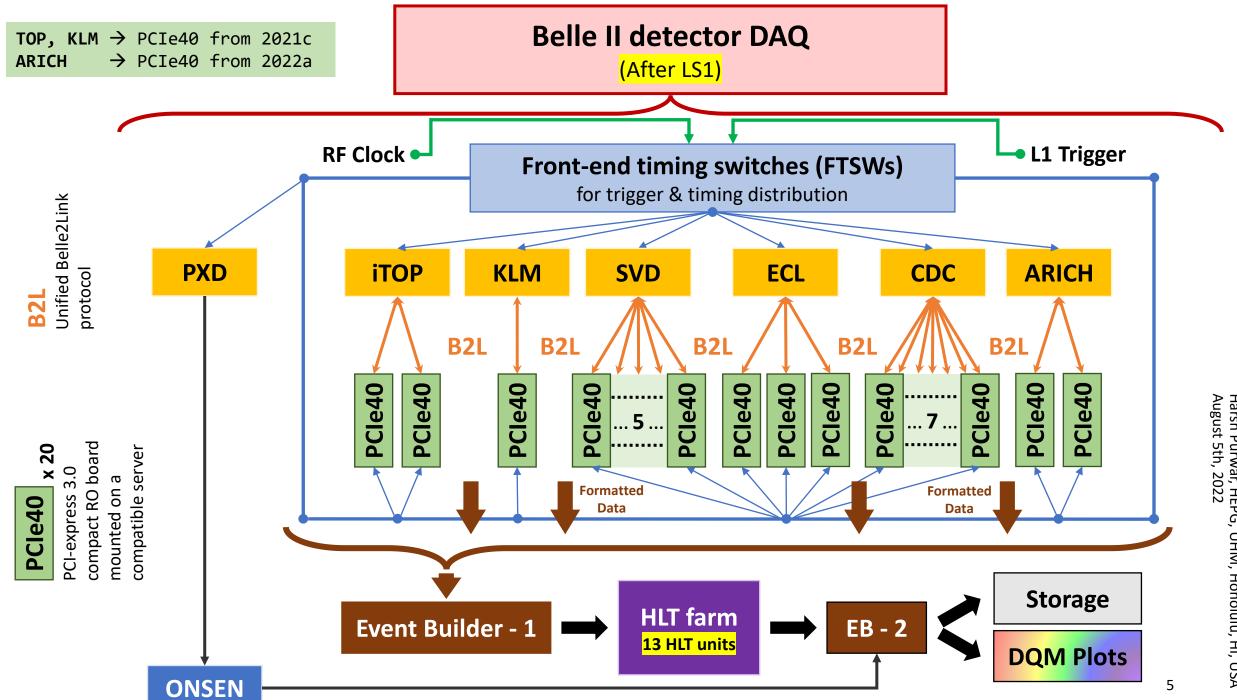
PCIe40 board

3

Transceiver module



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. <a href="https://stash.desy.de/projects/B2DAQ/repos/dag\_slc/browse">https://stash.desy.de/projects/B2DAQ/repos/dag\_slc/browse</a>

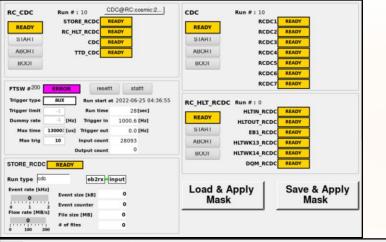


#### Global & Local run control GUIs

#### **Global Run Control GUI**

RC Command	Run st	un status Run control				TD Statu	s	Data flow			
STOP	Exp # : 27		RUNNING				G	RUNNING			
ABORT	Run # : 1:	34									
AUTO MODE OFF Detector states ( ABORT before you check or uncheck a subsystem)											
Run setting	□PXD	OFF	•	□ТОР	0	FF	□KLM	OF	F		
Run type : null											
Trigger type : poisson	<b>⊘SVD</b>	RUNN	NG		OFF		TRG RUNNIN		VING		
Dummy rate : 30000											
HLT script : beam reco monitor	<b>⊘CDC</b>	RUNN	NG	<b>⊴ECL</b>	RUNNING		HLT	RUNNING			
Trigger / Data status       Trigger # events :       Rate :       Trigger # events :       Rate :       Run start:         input       18300663       30.46 kHz       output       16435150       27.43 kHz       2022-07-01 13:34											
HLT01 H	LT02 HI	LT03 HL	T04 I	HLT05	HLT06	HLT07	HLT08	HLT09	HLT10		
# events : 0	0	0	0	0	3258917	3259599	3254472	3254100	3259775		
Rate : 0	0	0	0	0	5.6 kHz	5.8 kHz	5.7 kHz	5.6 kHz	5.6 kHz		
Flow : 0	0	0	0	0	414 MB/s	428 MB/s	419 MB/s	415 MB/s	420 MB/s		
ROISENDER NOTREADY EB2TX5 LUAD EB2TX1 NOTREADY EB2TX6 AUUH I EB2TX2 NOTREADY EB2TX7	NOTREADY EB2TX10 NOTREADY EB2TX11 NOTREADY EB2TX12		NOTREADY EE NOTREADY EE OFF EE	12TX20 OFF 12TX21 OFF 12TX22 OFF	EB2TX24 OFF EB2TX25 OFF EB2TX26 OFF EB2TX27 OFF EB2TX28 OFF	EB2TX29 OFF EB2TX30 OFF EB2TX31 OFF EB2TX32 OFF	SVD RUNNING STOP ABORT BOOT	Run # : 134 SVDRC RUNNING RSVD1 RUNNING RSVD2 RUNNING RSVD3 RUNNING RSVD4 RUNNING	RSVD5 RUNNING		
	RUNNING RUNNING LOAD ABORI BOOI	RTOP2 NOTRE		Run # : 108 RARI1 REA RARI2 REA		KLM NOTREAL LUAU ABURT BUUT		EADY ABORI BUOI	Run # : 134 RTRG1 RUNNING RCBTRGSRV RUNNING		
ECL         Run # : 134           RUNNING         RECL1         RUNNING           SLOP         RECL3         RUNNING           ADD/H1         BOUL1         BUIL			RC_I     RC_I     RC_I     RC_I	4LT01         OFF         S           4LT02         OFF         S           4LT03         OFF         S           4LT04         OFF         S	TORE01 OFF TORE02 OFF TORE03 OFF TORE04 OFF TORE05 OFF	RC_HLT07	RUNNING STOREOG RUNNING STOREO7 RUNNING STOREO8 RUNNING STOREO9 RUNNING STOREO0	RUNNING RUNNING RUNNING	C_ERECO1 OFF C_ERECO2 RUNNING MMASTER RUNNING		

#### **CDC Local Run Control GUI**



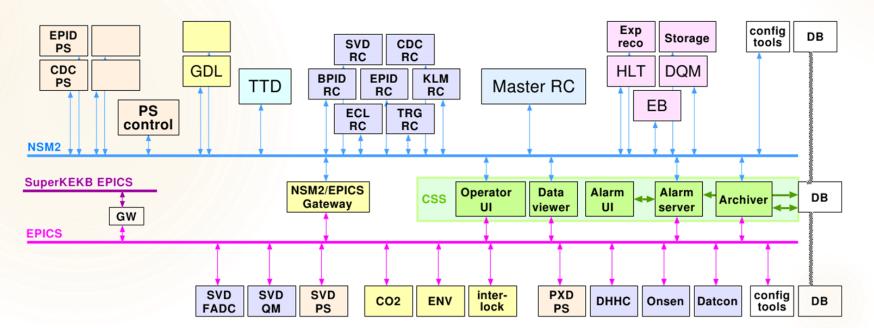
Hostname			_	TTD	DMA	DMA DMA [kBytes]		Size [Bytes]	Rate	[MB/s]	Program PCIe40		
✓ rodc1		READY	NOT	READY					0	0		0.00	0
								Belle2	ink-channe	H			0
020 - 00	-	00	0	120 -	01	-	10	0	276 - 02		0	200 - 03	
047 - 04	-	00	0	091 -	05	-	10	0	231 - 06		0	291 - 07	
046 - 08	-	00	0	090 -	09	-	101	0	230 - 10		0	290 - 11	
027 - 12		00	0	153 -	13	-	10	0	189 - 14		0	243 - 15	
026 - 16		00	0	152 -	17	-	10	0	188 - 18		0	242 - 19	
007 - 20		00	0	119 -	21	-	101	0	161 - 22		0	199 - 23	
006 - 24		00	0	118 -	25	-	101	0	160 - 26		0	198 - 27	
039 - 28		00	0	089 -	29	-	10	0	159 - 30		0	289 - 31	
038 - 32			0	088 -	33	-	10	0	158 - 34		0	288 - 35	
019 - 36		00	0	151 -	37	-	101	0	157 - 38		0	287 - 39	
018 - 40		00	0	150 -	41	-	101	0	156 - 42		0		
How to pr Push "Pro	ogram PC	on Feb. 18, 2 Ie40 firmware e40° and wait	e t until ti										

Update channel checkboxes and push "Save & apply Mask"

Load and apply the last saved mask setting Push "Load & Apply Mask" and then checkboxes should be updat Refresh OPI to confirm the update.

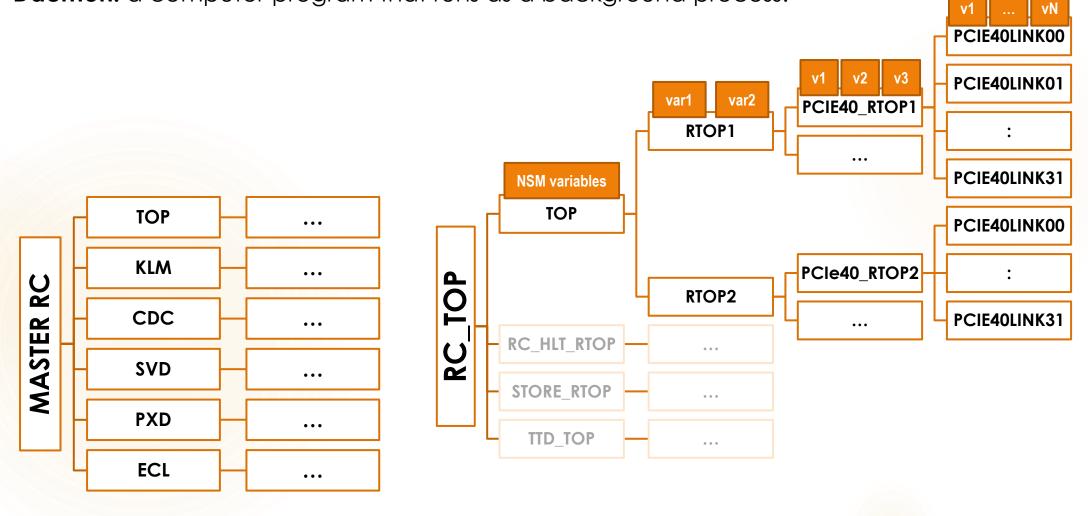
### Network shared memory (NSM2)

- All ROPC (rtopX, rk1mX, 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. <u>https://stash.desy.de/projects/B2DAQ/repos/daq\_restart/browse</u>



### 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.

**v**3 PCIE40LINK01 List of all NSM variables for monitoring & control RTOP1 **CDC** specific variables **PCIe40** specific variables cdc[1].adcth : int get set B2L EVTSZ : int get B2L FEEHW cdc[1].dac : int get set : int get PCIE40LINK31 : text get set B2L FEESER : int get cdc[1].date cdc[1].delay : int get set B2L\_FF\_FULL : int get cdc[1].fmver B2L FF USAGE : int get set : int get TOP cdc[1].indirectadc : int get set B2L LFF USAGE : int get PCIE40LINK00 cdc[1].indirectmon : int get set B2L\_MASK : int get B2L MAXEVTSZ cdc[1].mode : text get set : int get KLM RTOP2 cdc[1].ped[0].val : int get set B2L SETMASK : int get set RC cdc[1].ped[1].val : int get set **B2L STATUS** : int get NEVENTS cdc[1].ped[2].val : int get set : int get CDC PCIE40LINK31 NKBYTES : int get MASTER **Others:** cdc[1].ped[46].val : int get set **SVD** cdc[1].ped[47].val : int get set dbtable : text get fee.used cdc[1].tdcth : int get set : int get set cdc[1].tem : float get rcconfig : text get set PXD cdc[1].vcc5v : float get rcrequest : text get cdc[1].vccaux : float get rcstate : text get cdc[1].vccint : float get runtype : text get set ECL cdc[1].window : int get set version : text get

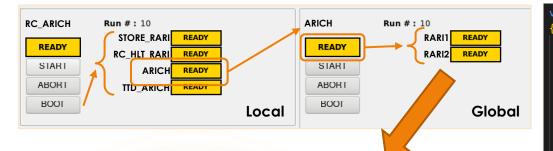
10

vN

**v1** 

PCIE40LINK00

#### 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++)</pre>
 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 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 CDCIndirectADCAccessHandler(vname + ".dac", callback, b2link, \*this)); callback.add(new CDCIndirectMonitorAccessHandler(vname + ".indirectmon", callback, b2link, \*this)); for (int i = 0; i < 48; i++) {</pre>

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));

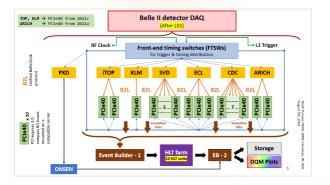
#### void TOPPcie40FEE::monitor(RCCallback& callback, B2LINK& b2link)

std::string m\_monitoring=""; callback.get("top\_monitoring", m\_monitoring); if(m\_monitoring=="0N") { 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

- 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.



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

# Data flow in Belle II DAQ

- Belle II detector DAQ (After LS3) (Belle II detector DAQ (Belle II detecto
- 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 2<sup>nd</sup> 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: <u>https://indico.belle2.org/event/6444/sessions/2337/attachments/17913/26632/B2SW2</u> <u>2\_Bennett\_DP.pdf</u>



## Data quality monitoring

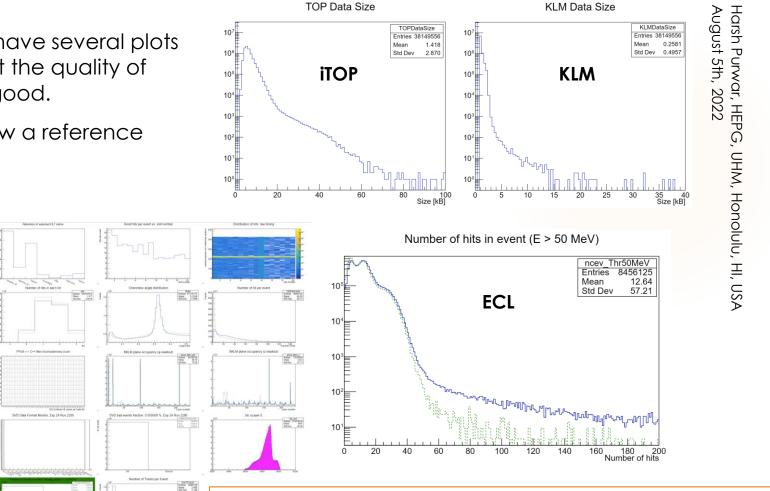
(DQM plots: <a href="https://dqm.belle2.org">https://dqm.belle2.org</a>)

- 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.

Hereiter
 Hereiter<

2280 3014950

Events Officer Events 3607828 March 24008 History 2,210



PXD

GOOD

Shifte

Expert

SVD

GOOD

GOOD

CDC

GOOD

GOOD

тор

GOOD

GOOD

ARI

GOOD

GOOD

ECL

GOOD

KLM

GOOD

GOOD

TRG

GOOD

GOOD

RAWDATA

GOOD



### **Planned improvements**

- 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

