Data Acquisition and Online Monitoring Tools

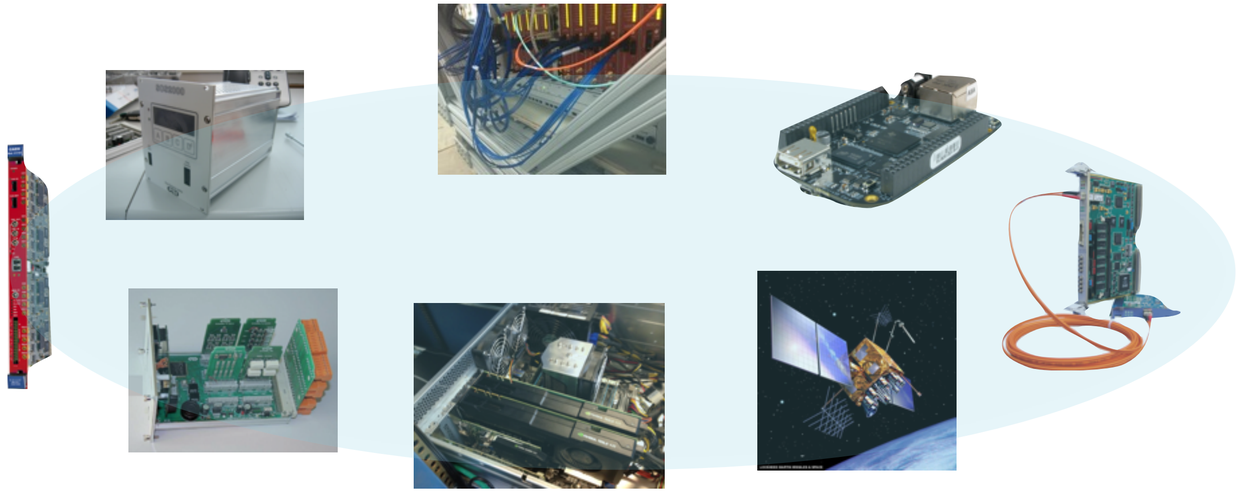
Wes Gohn
University of Kentucky
2 October 2017

Charge Questions

- Charge Questions 1h: “a description of the roles and responsibilities of each institution.”
  - will list institutions with contributions to DAQ systems

- Charge Question 2: “Has it been demonstrated that the experiment is ready for physics-quality data taking? If not, what actions are required to make it ready? Is there a clear plan for monitoring (the beam and) the data quality and has the associated infrastructure been tested? If not, what actions are required to adequately monitor the data quality?”

DAQ Overview



- The DAQ must assemble data from an “Internet of things” to form a complete picture of each muon fill.
- Expect an average data rate of 12 Hz for muon fills, plus some laser and pedestal fills.
- This provides data at a rate of 20 GB/s, which must be reduced via processing of data in GPUs.
- An art-based online data quality monitor allows us to monitor the data.

Data Flow

10 Gbps Fiber x28

1 Gbps CAT6 x18 to 10 Gbps fiber

2x10 Gbps Fiber

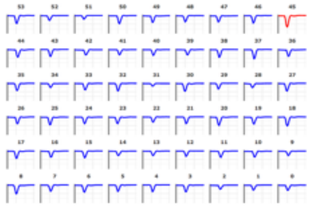
µTCA x28

20 GB/s

Frontends/GPUs

200 MB/s

DQM



FTS/SAM

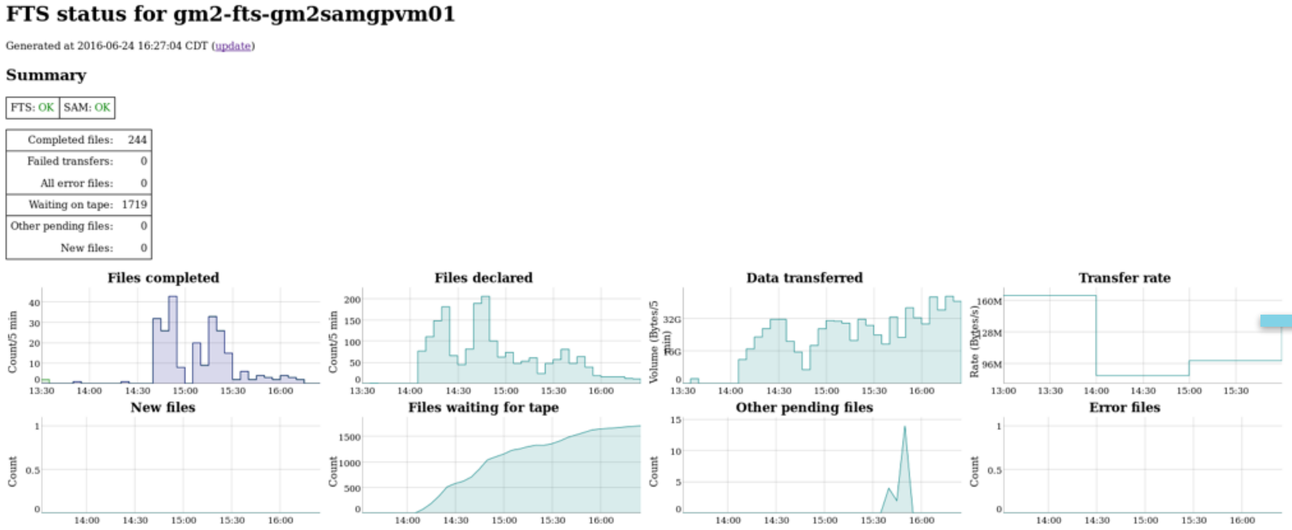
200 MB/s

Fermilab enstore tape storage

Fermilab

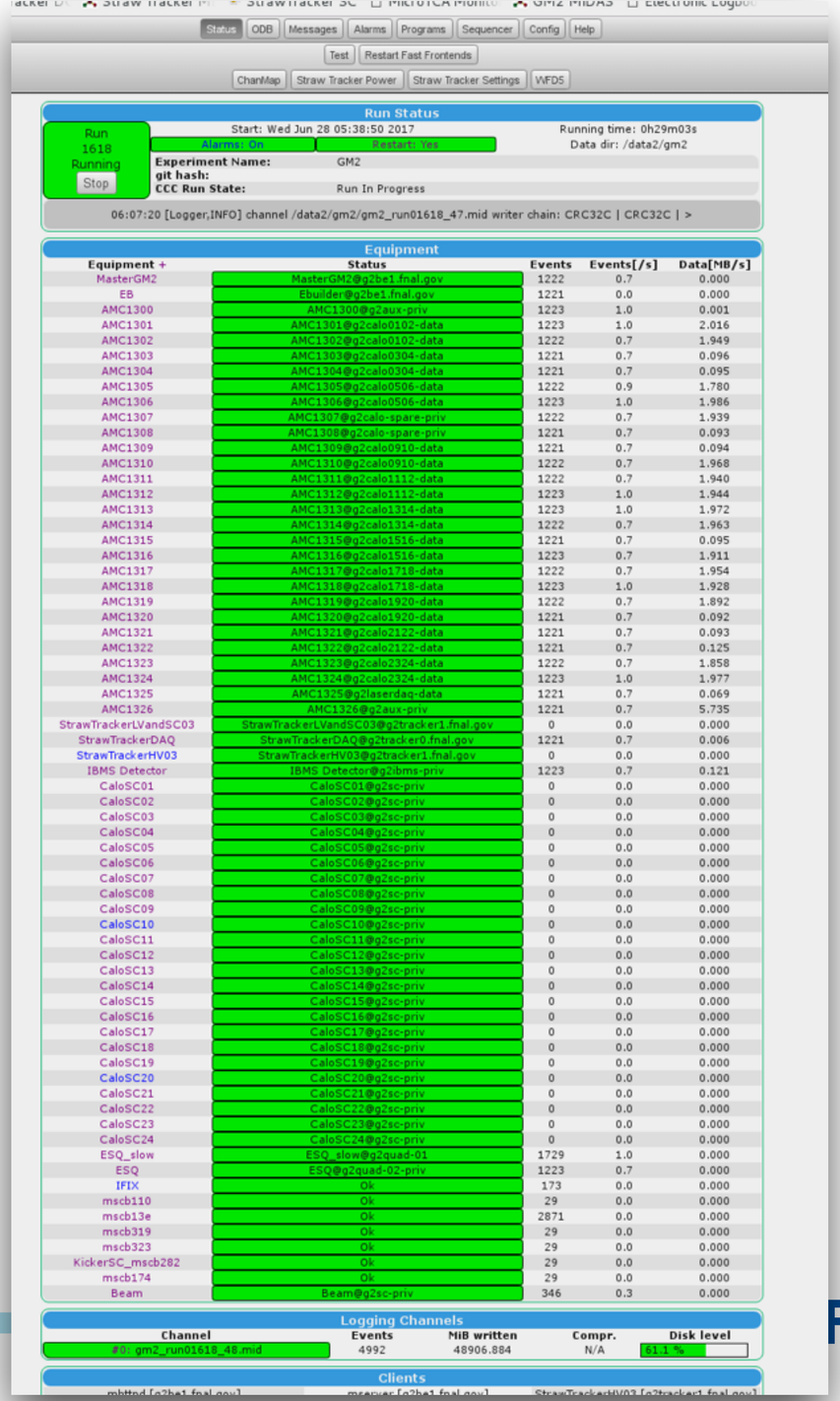
Data storage

- Data is copied from local RAID using FTS/SAM to the Fermilab enstore tape system.
- The local RAID can accommodate four days of running if the link goes down.
- Both CRC32C and enstore checksums are computed for each file.
- A python plugin extracts the SAM metadata from a MIDAS JSON configuration file after each run.
- We wrote 20 TB of data during the commissioning run.
- We expect to write 6 PB of data in total.
- Two copies of the raw data will be stored in the MIDAS format as well as derived data in the *art* format.



MIDAS configuration

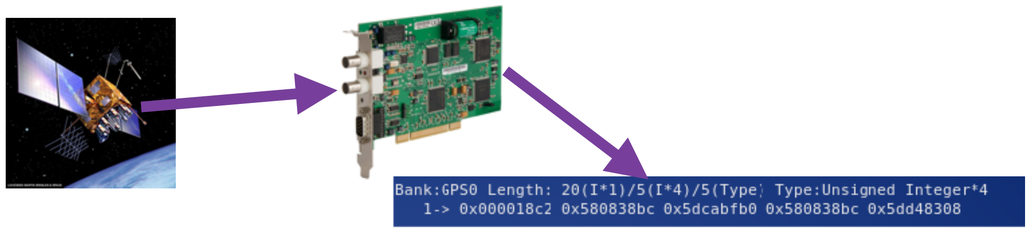
- 32 fast frontends (data at beam fill rate).
- 35 slow control frontends.
- Midas alarm system.
- Midas sequencer used for calibration runs.
- ODB dumped to JSON file and saved to Postgres database at each end of run.
- Online analyzer using *art* and javascript.
- Separate MIDAS experiment running for magnetic field DAQ.

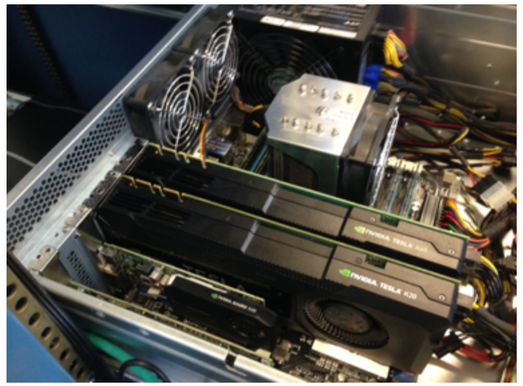


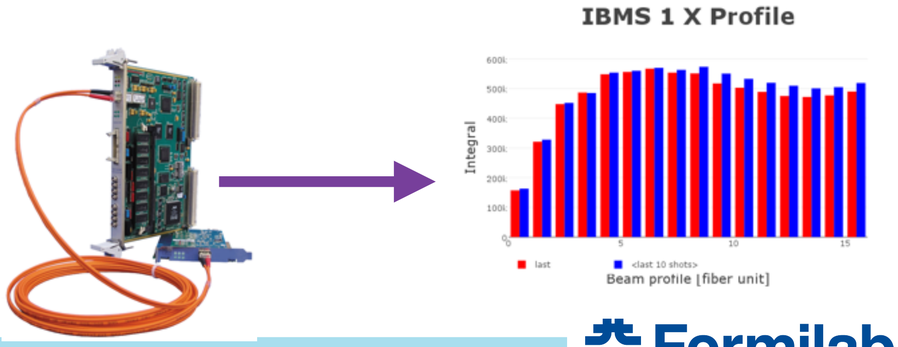
Fermilab

MIDAS Frontends

- Master:
  - Communicates with other frontends with RPCs, sets up CCC, and writes GPS timestamps from Meinberg GPS unit.
- AMC13 frontend:
  - Main frontend for processing data from calorimeters, laser, fiber harps, quads, and kickers.
  - Processes the data with Nvidia Tesla K40 GPUs
- Tracker frontend:
  - Data comes from multihit TDCs that are read via FC7 cards.
- IBMS frontend:
  - Data from the inflector beam monitoring system (IBMS) is read out via a CAEN digitizer.





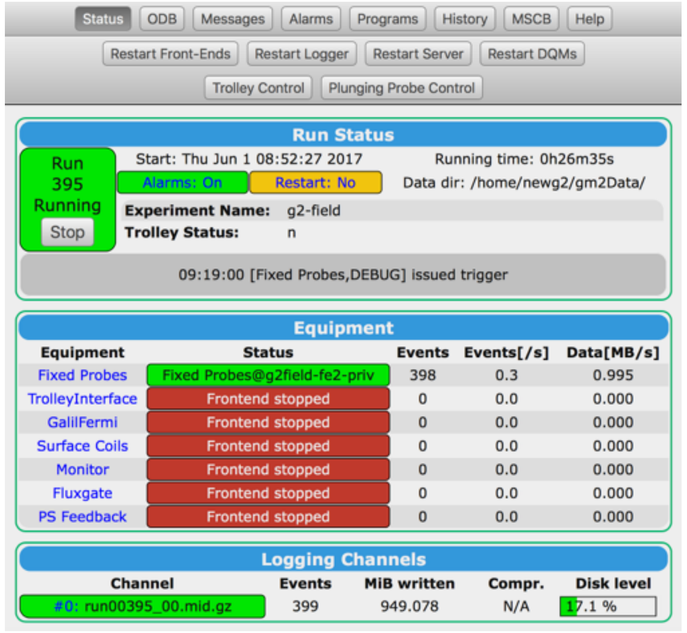


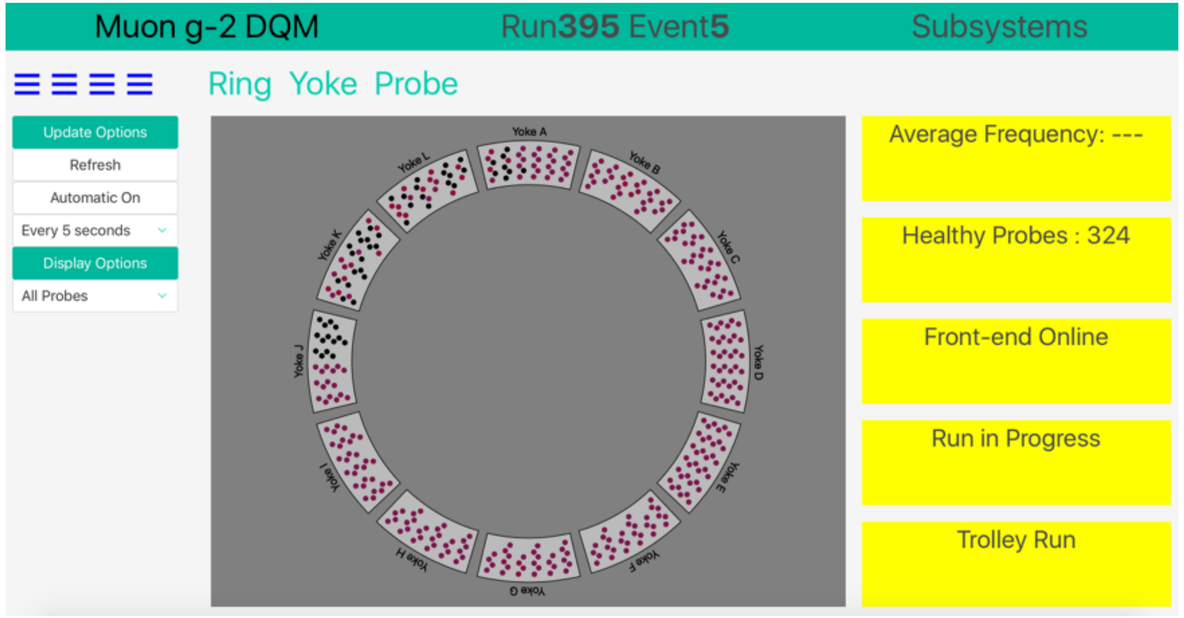
Slow Controls

- Slow control data is read from six SCS3000 mscb devices and 24 beaglebones.
- HV and LV frontends for tracker system.
- Slow frontend reading magnet properties from IFIX via an OPC client.
- Slow control data is stored in a Postgres database and displayed using a custom Django web display.

Field DAQ

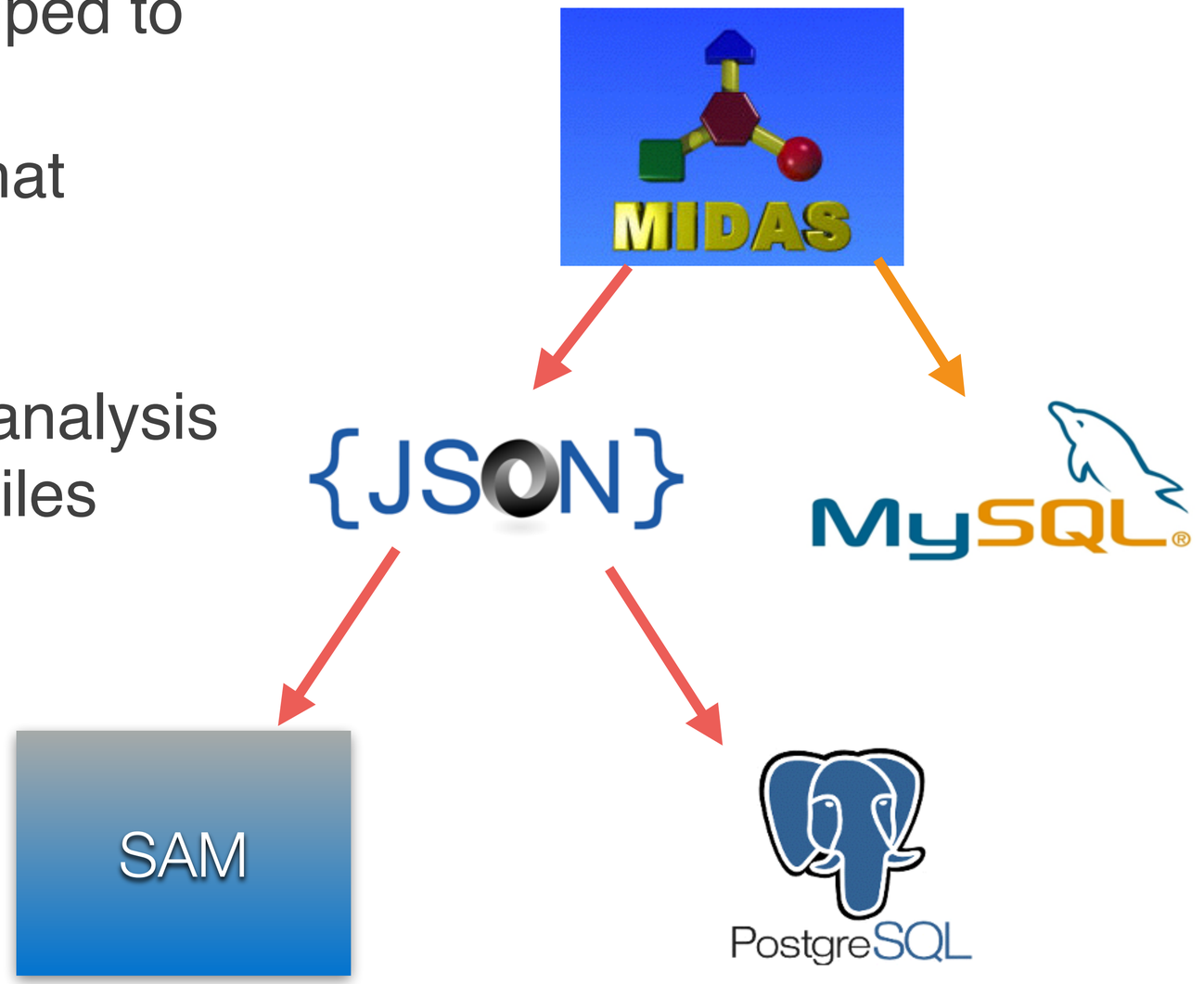
- Field DAQ runs in independent MIDAS experiment.
- Contains seven asynchronous frontends reading data from fixed magnetic field probes and from a trolley that periodically transverses the ring to perform precision measurements of magnetic field.
- Data is correlated with the fast DAQ offline using GPS timestamps.





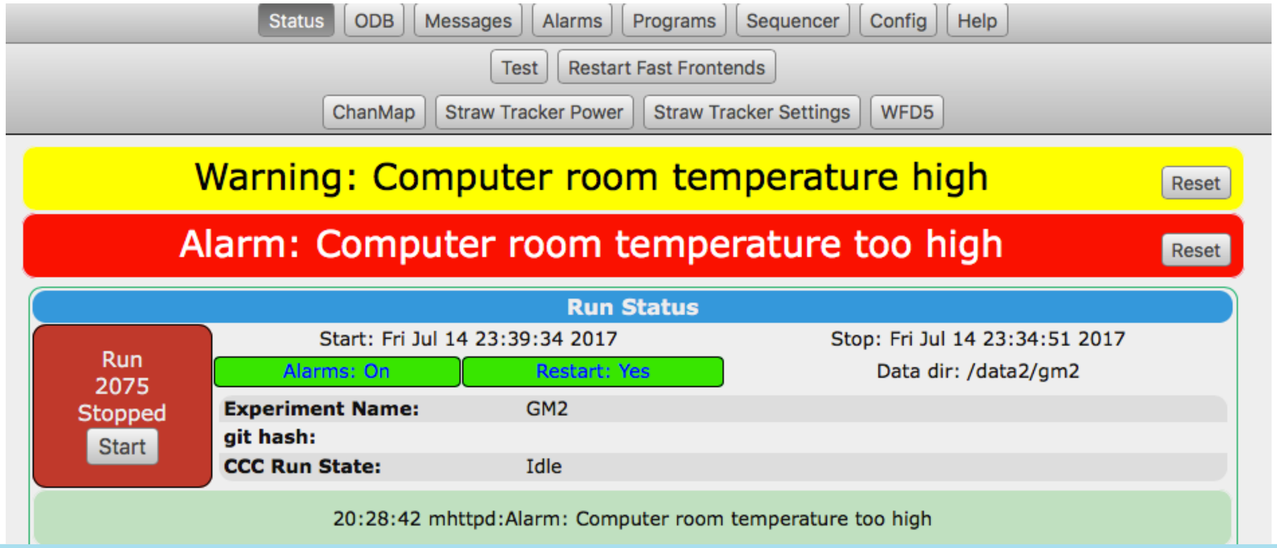
MIDAS ODB Archive

- At each end of run, the ODB is dumped to a JSON file.
- A python routine is then executed that imports the entire JSON file into a PostgreSQL database.
- Metadata that is used in the offline analysis is also extracted from these JSON files using python plugins.



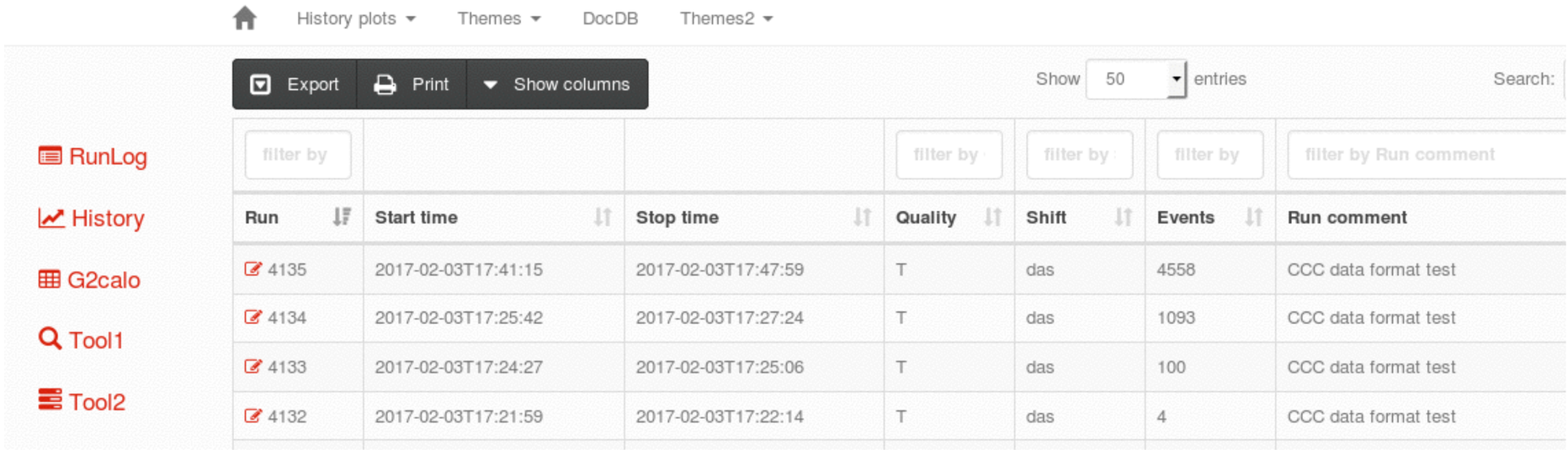
MIDAS Alarm System

- The MIDAS alarm system is used as the primary alarm system.
- Alarms are set on temperatures and voltages from slow control devices.
- Other slow frontends set alarms automatically when encountering an error.
- Periodic alarms reminded shifters to perform shift checks.
- Each alarm sounds an audible beep and speaks a message, while also displaying a banner at the top of the main DAQ page.
- Some alarms post automatically to the Fermilab electronic logbook.



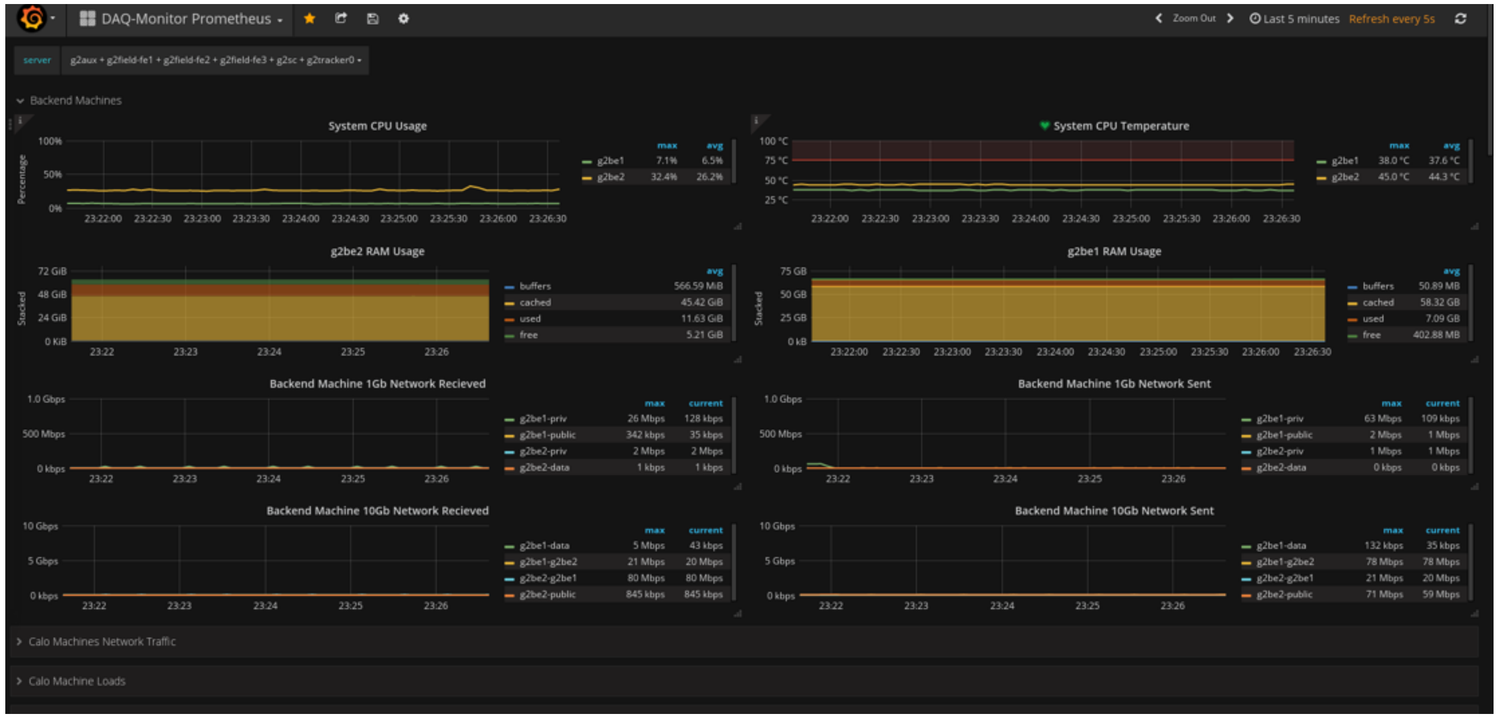
Online Databases

- Slow control data is stored in Postgres database.
- Entire MIDAS configuration is exported to separate table in Postgres database after each run.
- Postgres database is stored locally on 10 TB RAID
- Real-time backup to FNAL server.
- Plan to add redundant local database server.
- Access via custom Django-based web display.



DAQ Health Monitor

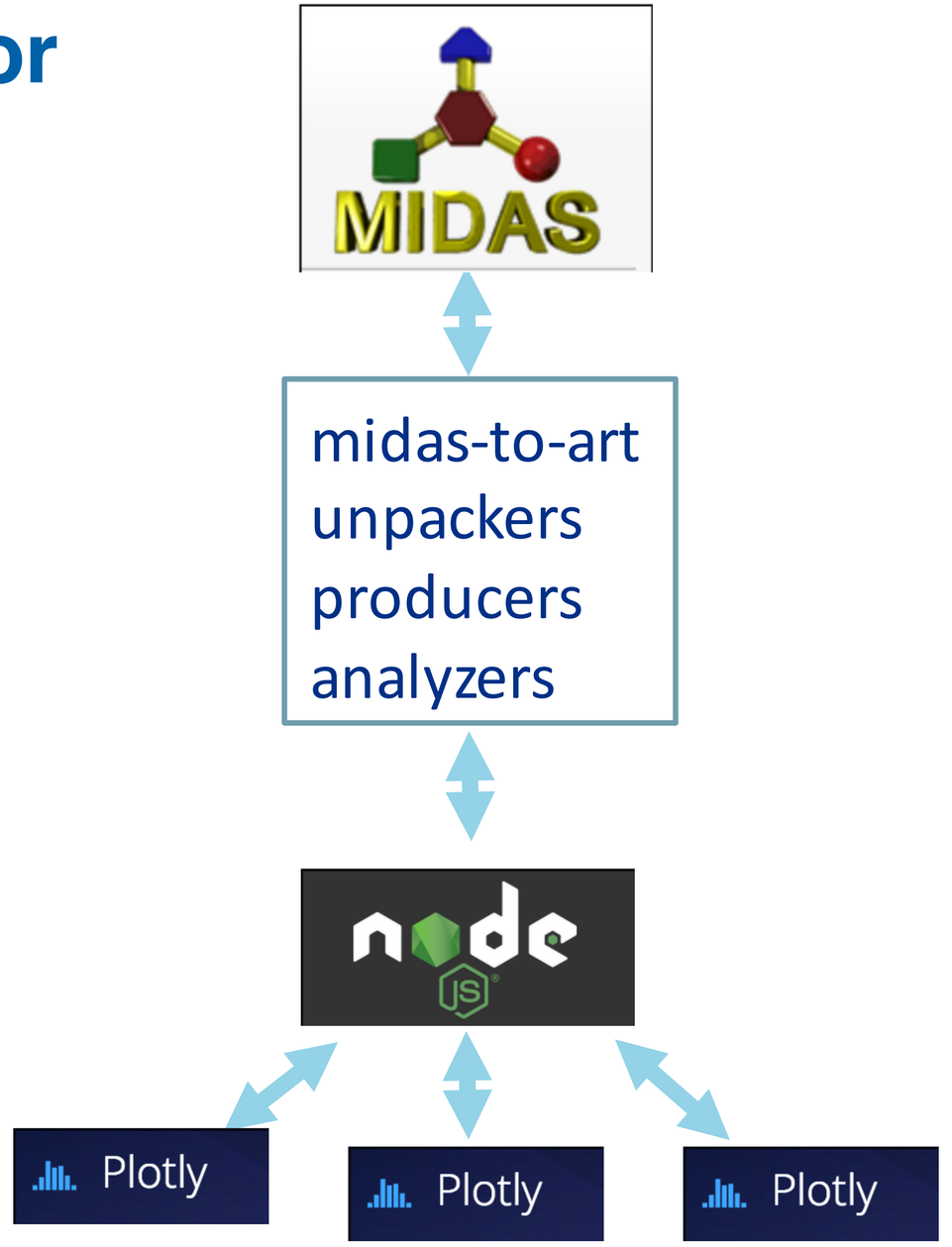
- Monitored health of DAQ systems using netdata for system monitoring, prometheus for short term data storage, and grafana to display data.



- Sets alarms and notifies experts when values go out of range.
- Satisfies recommendation from 2016 computing review.

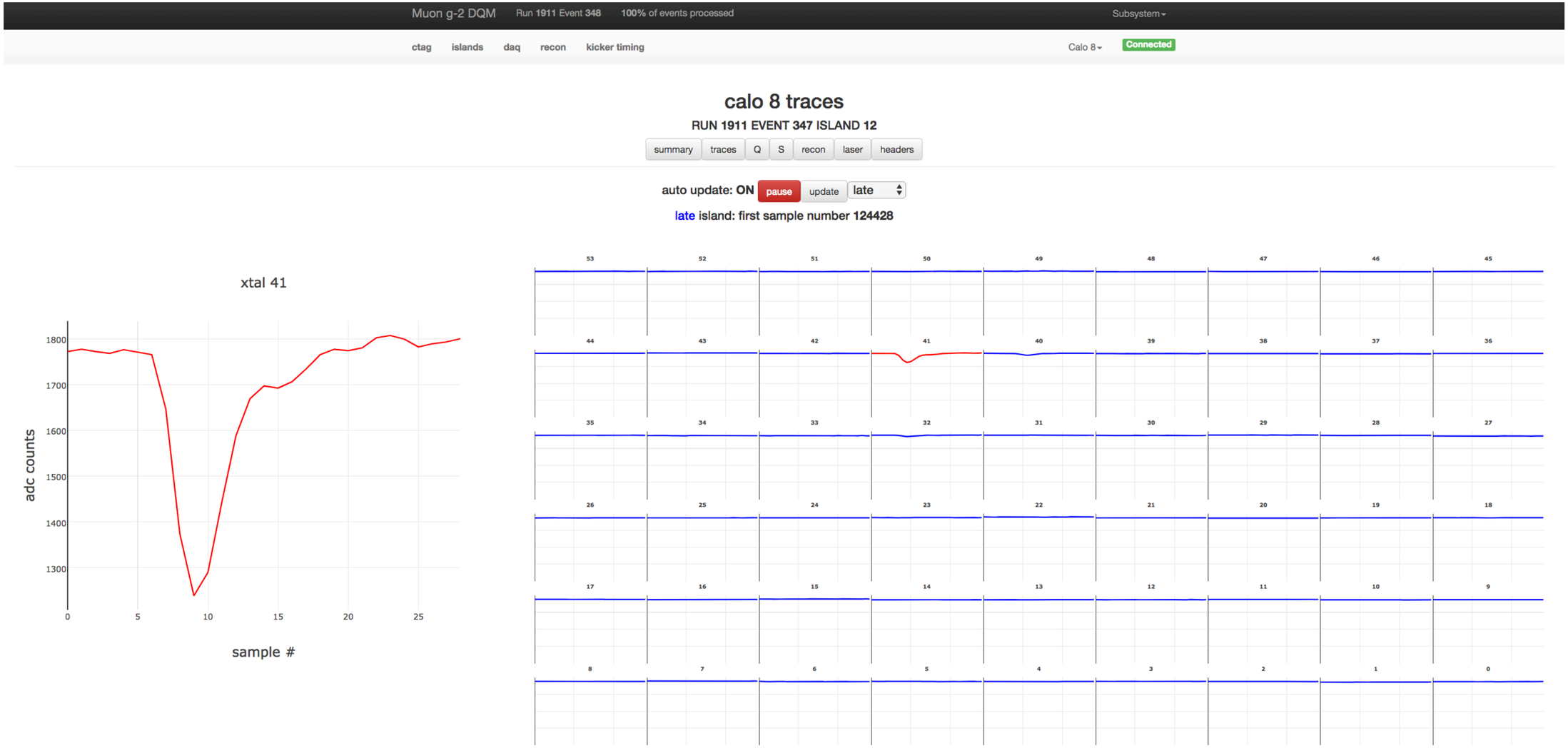
art-based Data Quality Monitor

- Online DQM streams data from MIDAS mserver.
- Uses existing art modules to process the data.
- Data is extracted from the art job using ZeroMQ and published to a web GUI using node.js and plotly.



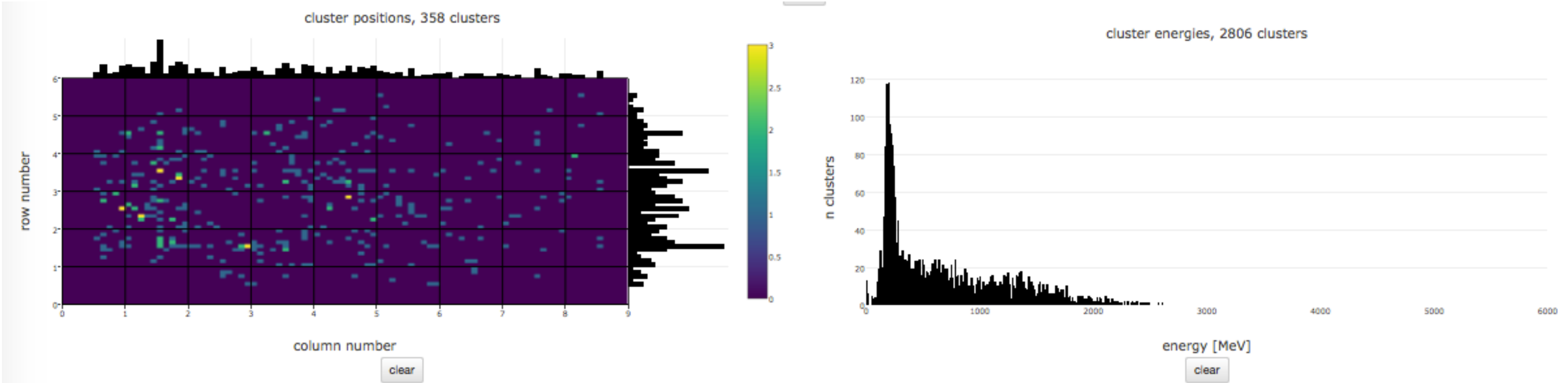
Online monitoring of the experiment

Position of pulse in calorimeter

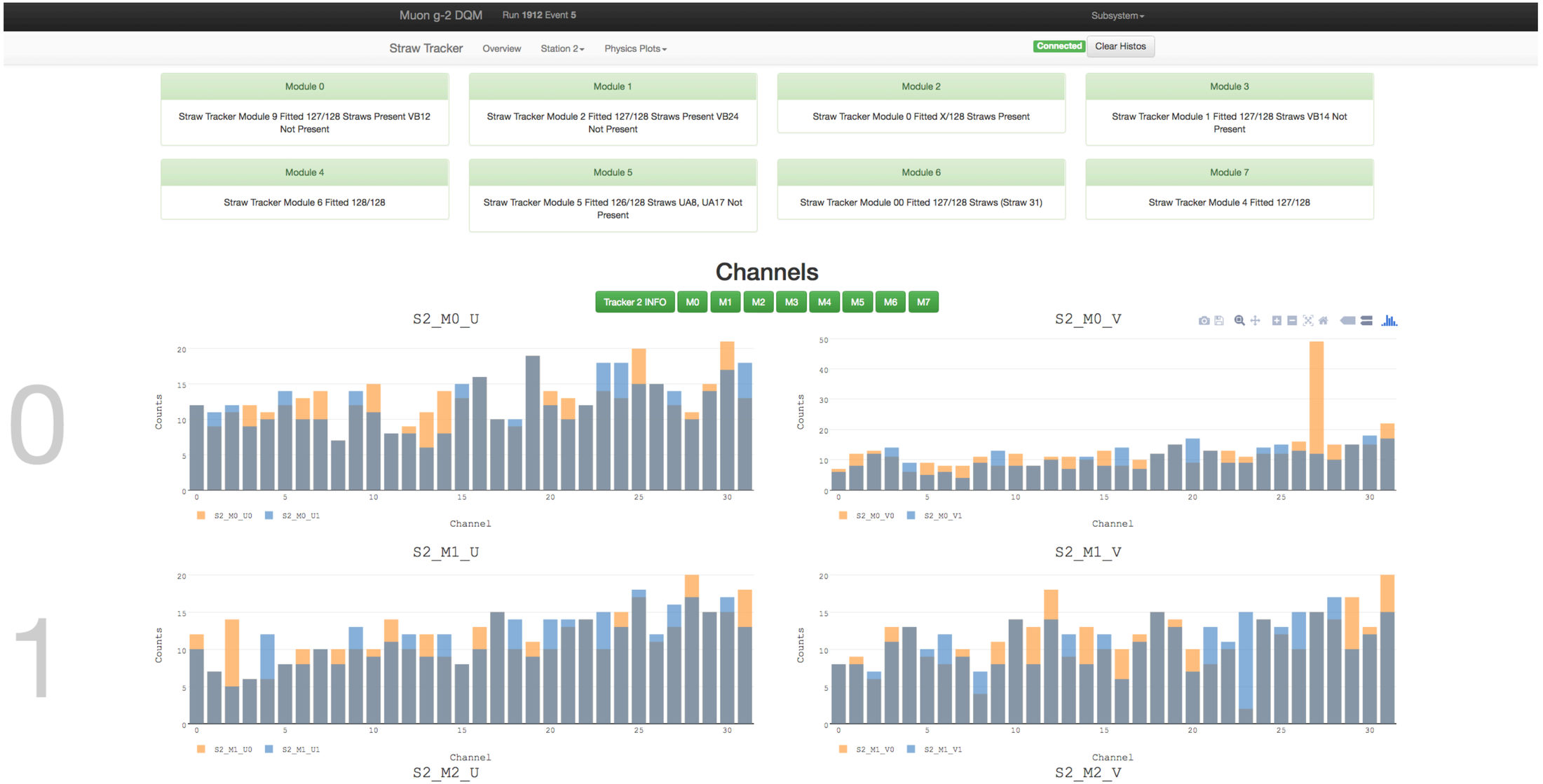


Online reconstruction

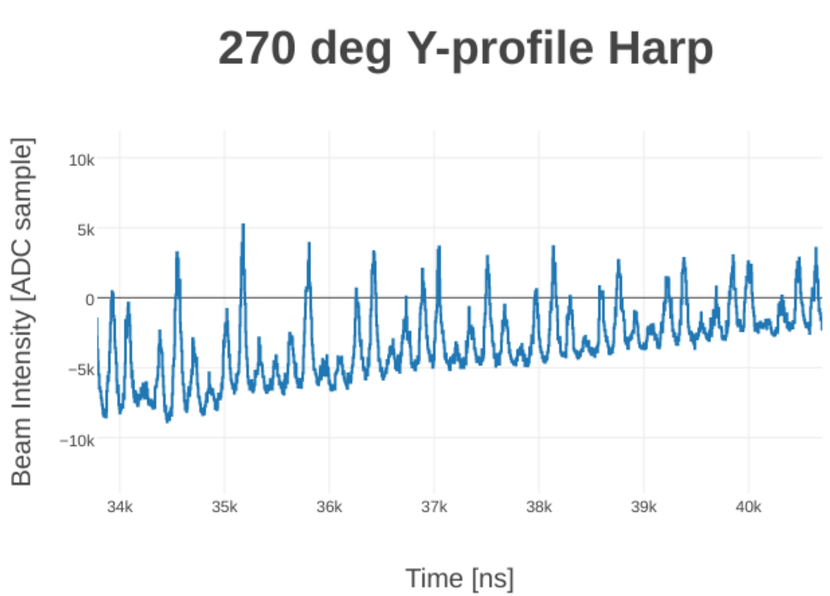
- The online DQM uses GPU-based template fits to do clustering and energy reconstruction.

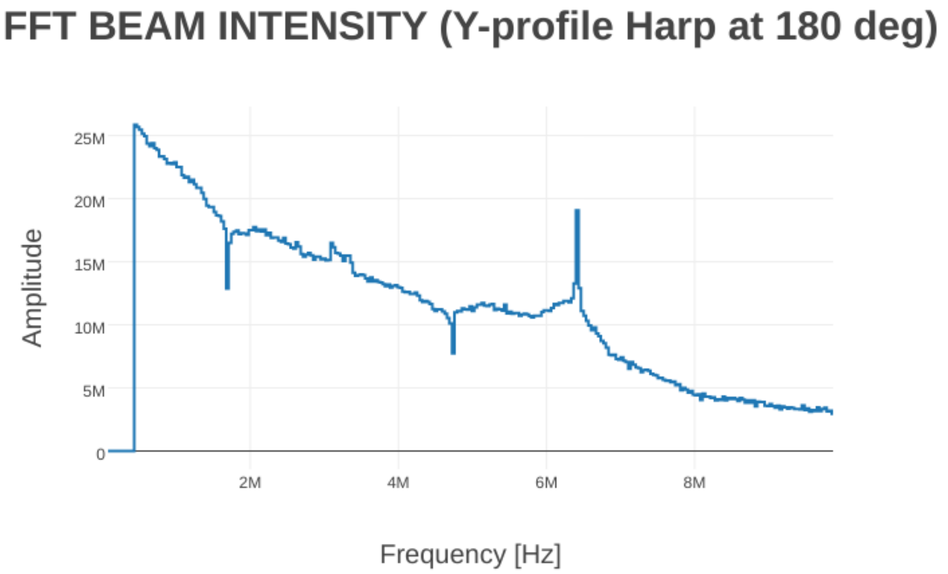


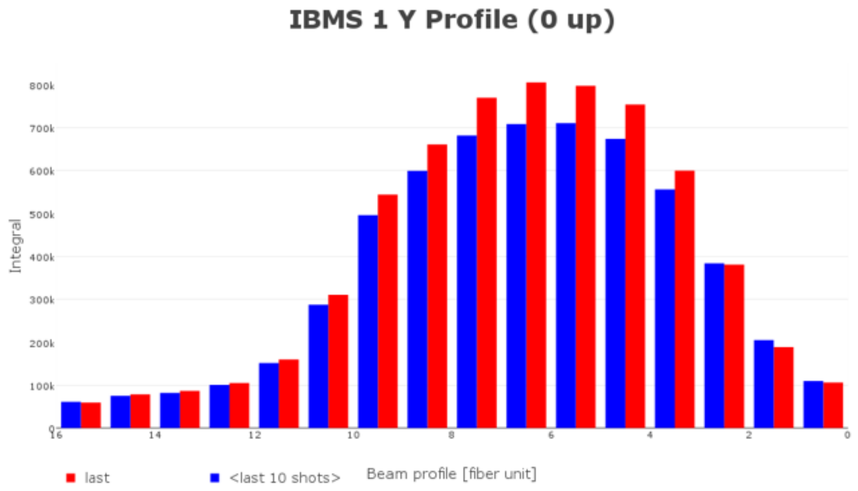
Tracker Monitor

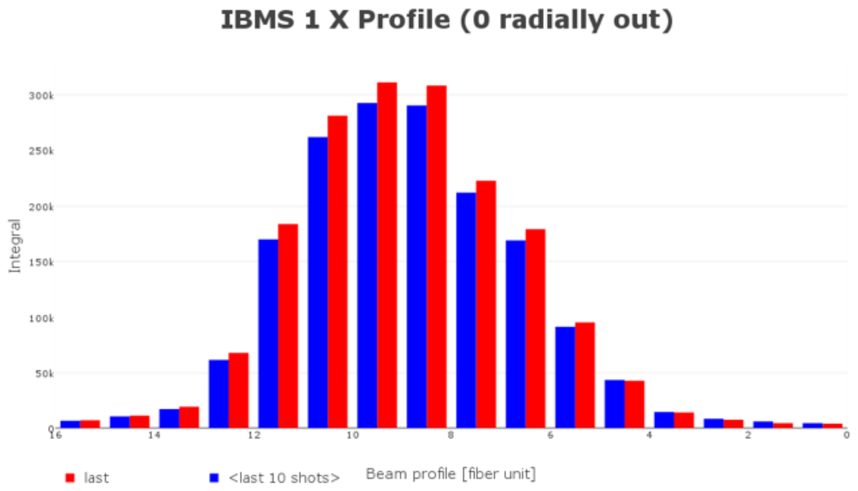


IBMS and Fiber Harps

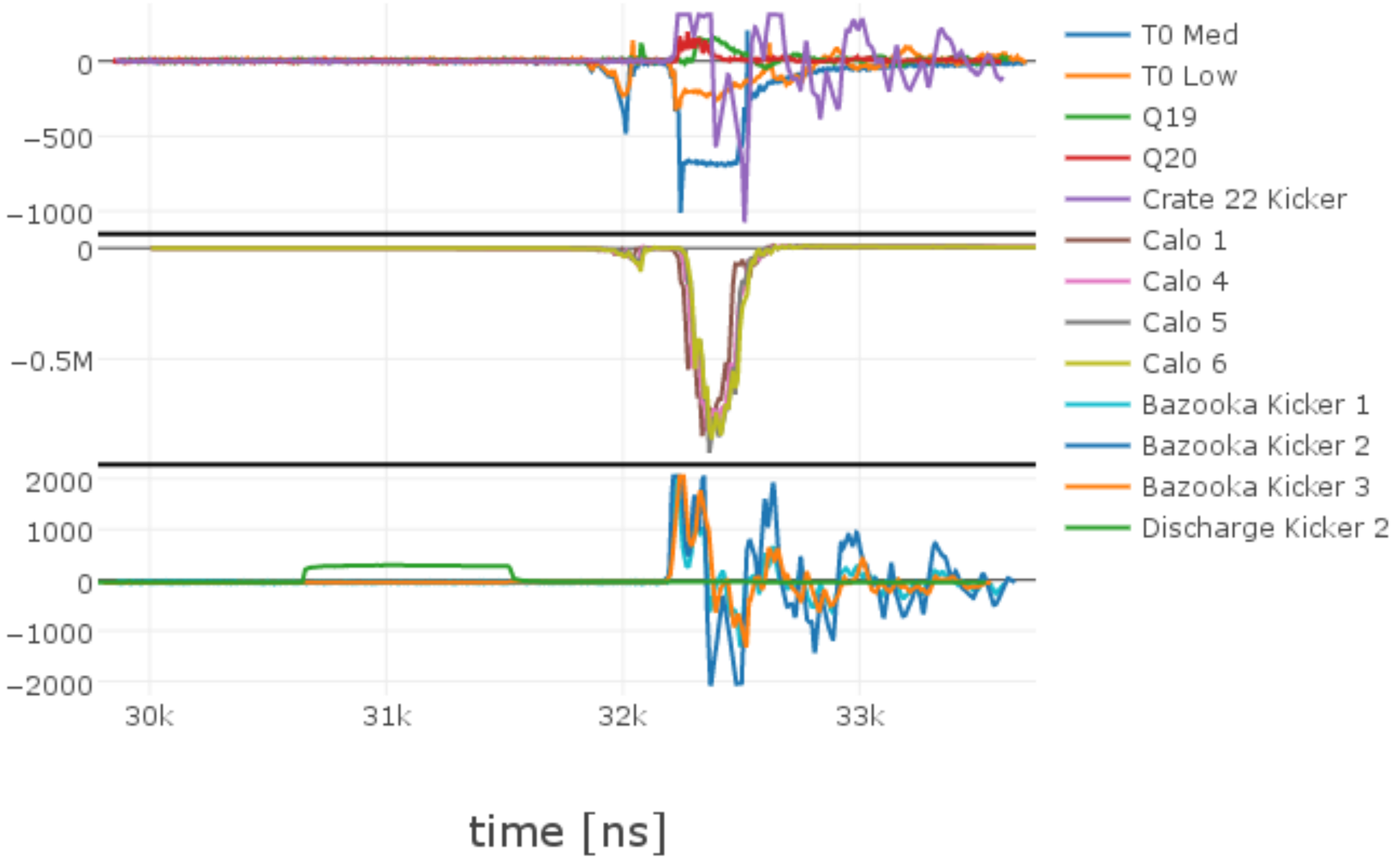






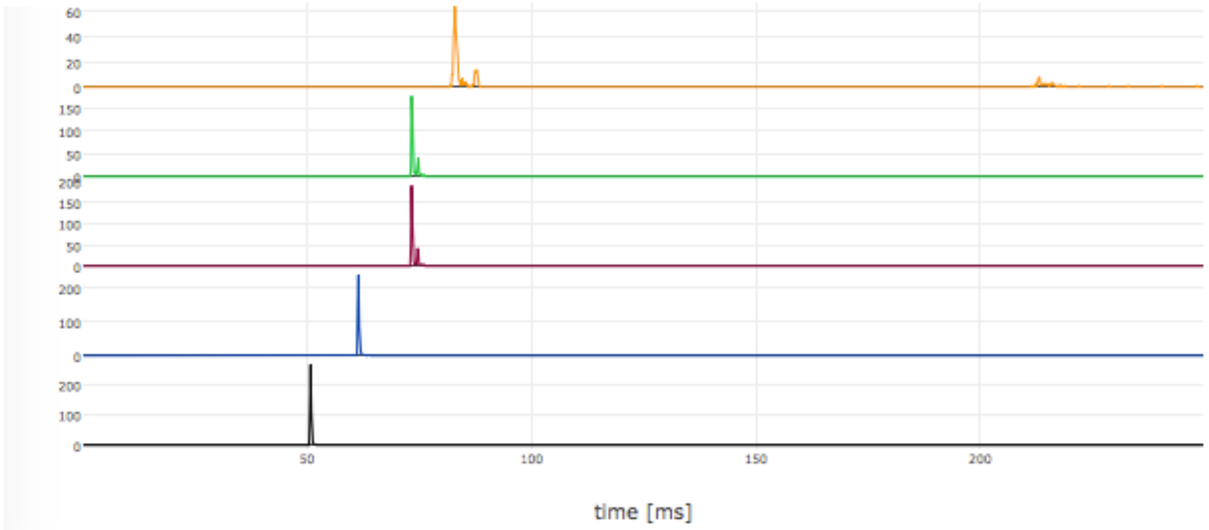


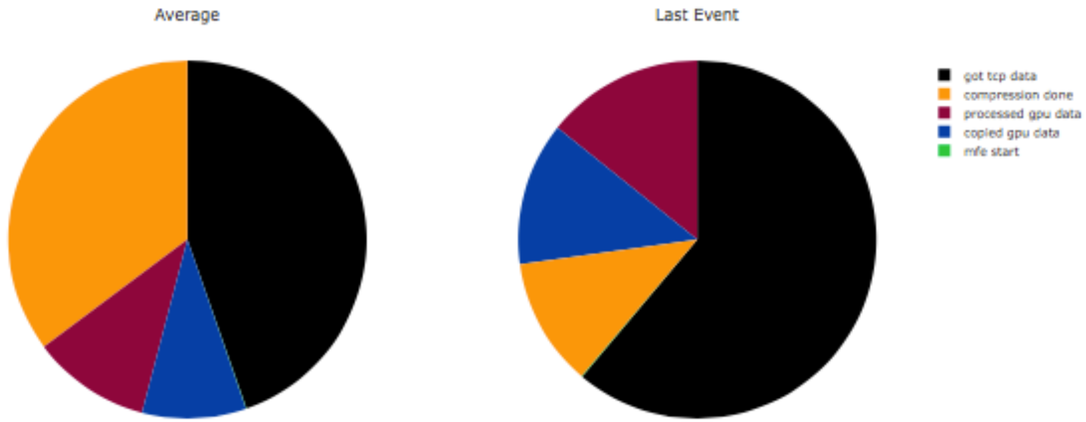
Monitoring Beam Injection



Online DAQ Profiling

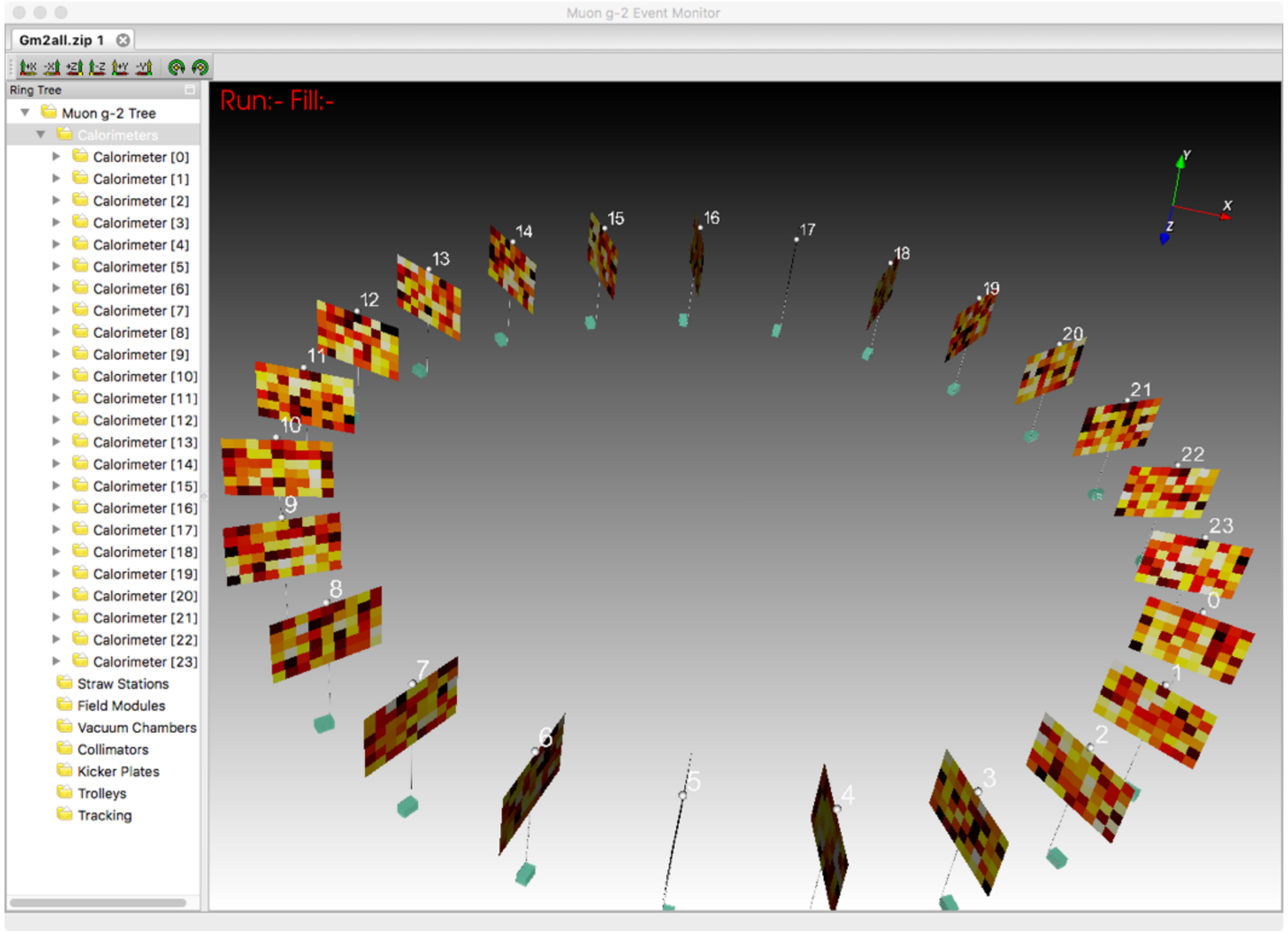
- Must process each event in 83 ms to keep up with average beam rate of 12 Hz.
- Most time is spent reading data from TCP socket and copying it to the GPU.
- Processing time in the GPU is very small.





Event Display

- Based on Paraview (vtk).
- Reads data from same *art* jobs as DQM.
- Not ready for commissioning run, but it is being implemented now.
- Currently includes only calorimeters, but plan to add more subsystems soon.

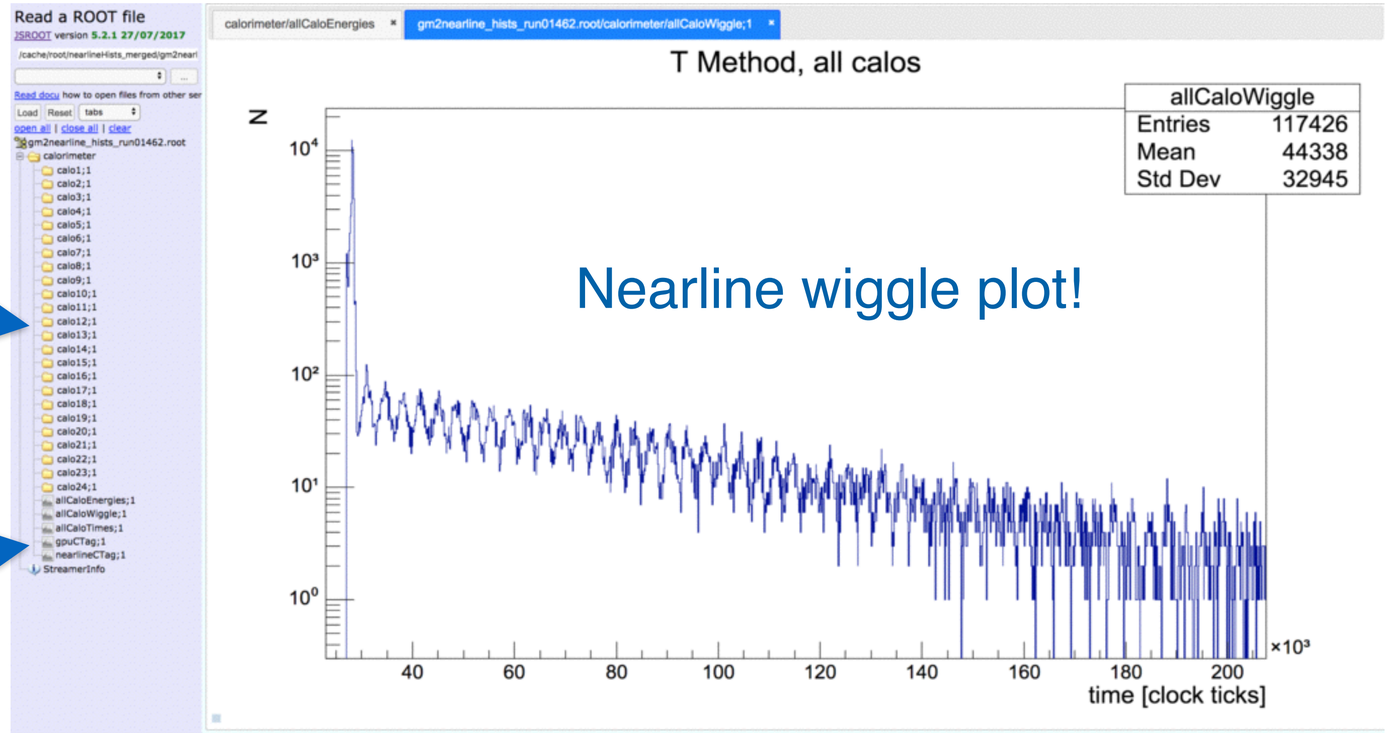


Nearline analysis

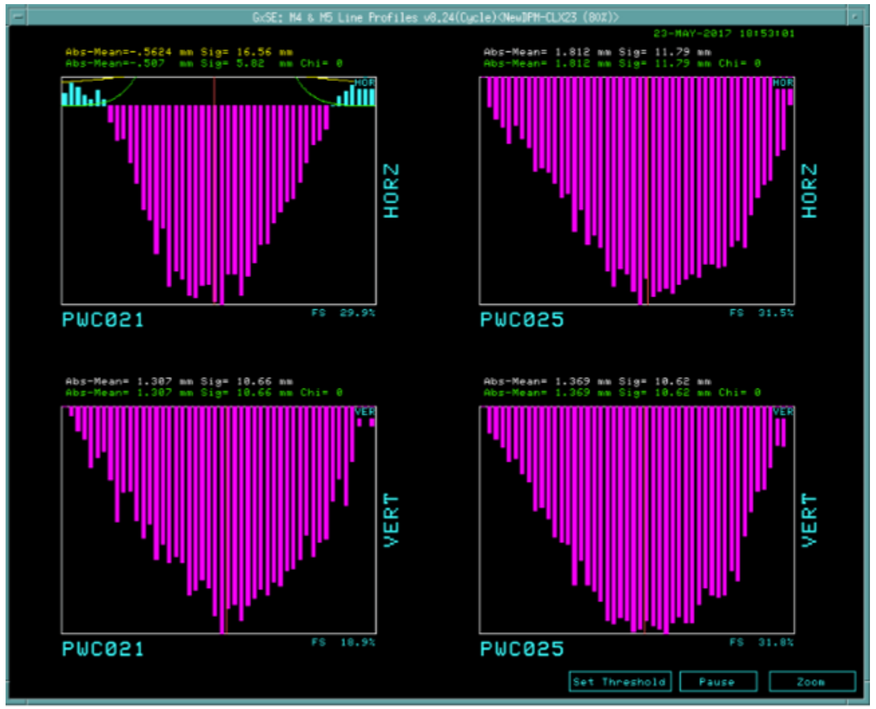
- Two dedicated machines in the control room are used for near line analysis.
- The near line analysis processes data from the most recent run, but performs the full art-based analysis on every run it processes.
- It is allowed to get behind the current run, but so far it can keep up with the live running.
- A set of histograms is made available for each processed run via the Django based web display utilizing JSRoot.

Select run, detector, and plot from menu

Plots combining data from multiple detectors.



Beam monitoring

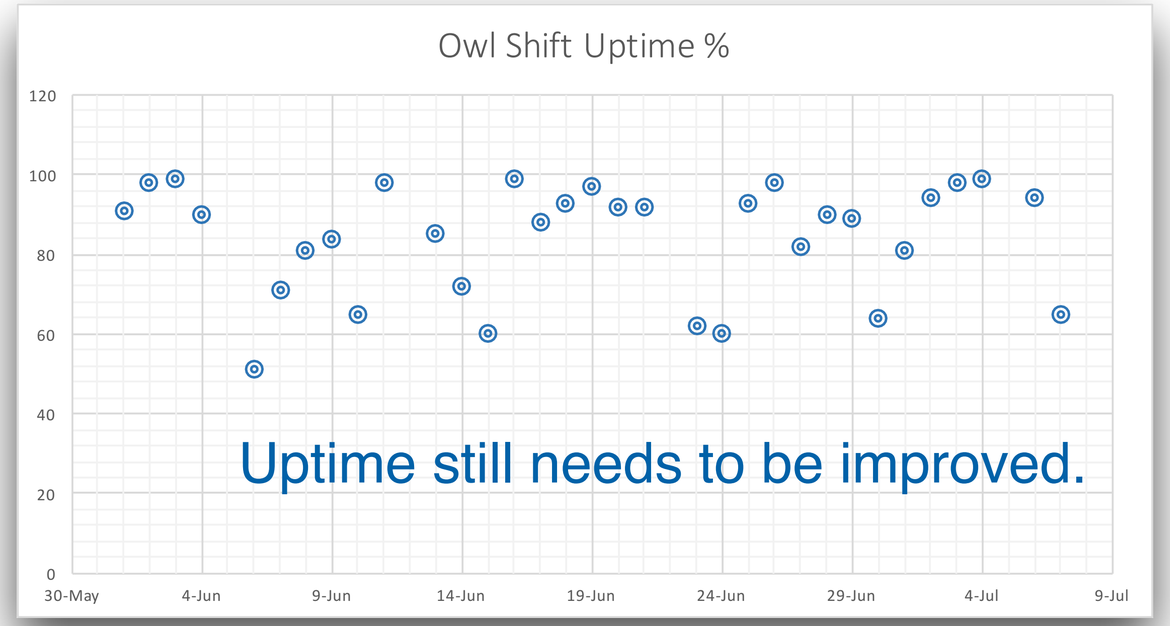


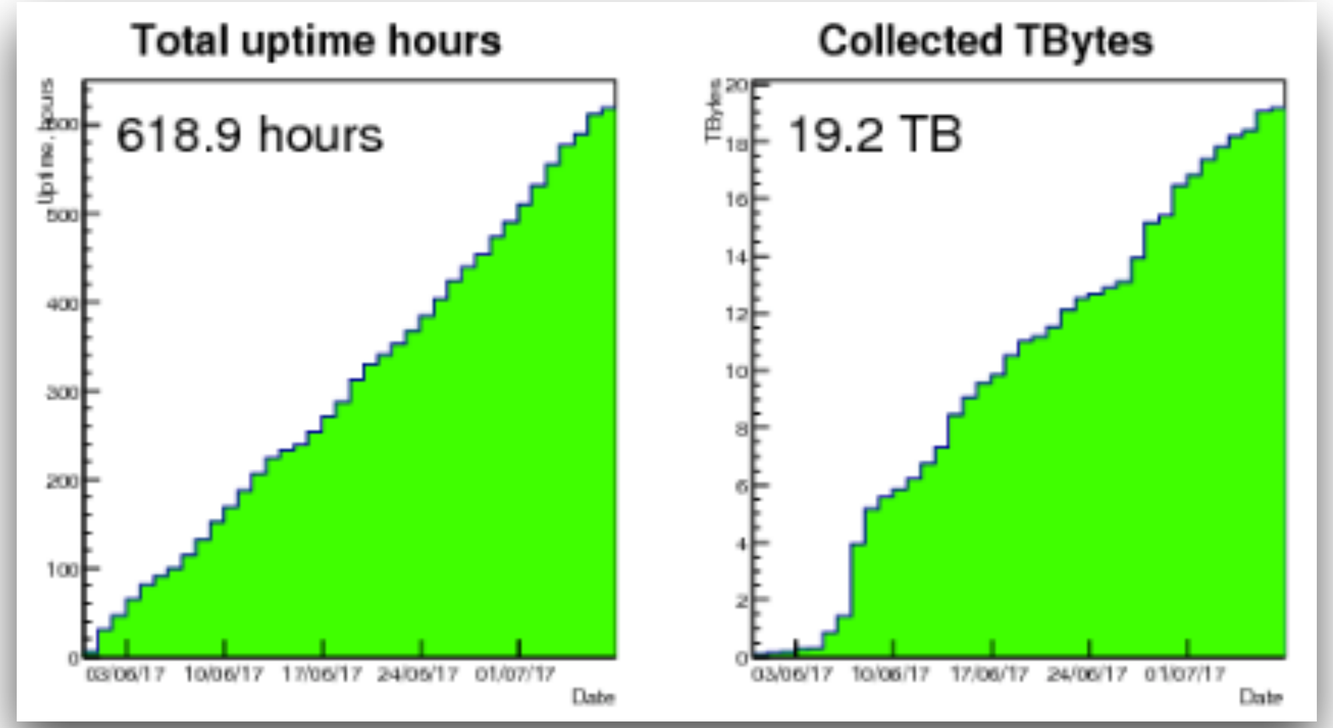
- ACNET
  - A dedicated ACNET terminal is in the control room as is used by shifters to observe the status of beam line components.
- IFBeam
  - An IFBeam display is kept visible for the shifter on one of the control room monitors to view trend plots of beam line measurements such as protons on target.
- MIDAS Beam frontend
  - A dedicated MIDAS frontend reads data from the IFBeam service and writes it in the data stream.

To-do before next run: Update MIDAS frontend to account for relative timing between beam line components.

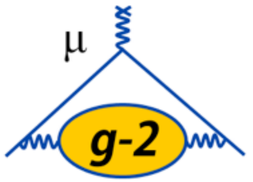
DAQ Performance During Commissioning Run

- The MIDAS DAQ performed well during our first run.
- Day shifts were mostly dedicated to beam line commissioning, so production data was taken at night.



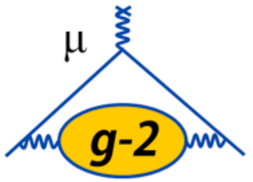


Hardware redundancy plan



- Working with SLAM to develop a detailed plan for the replacement of any machine in the DAQ system.
  - Hot spare for GPU-based frontend machine for calorimeter readout (hot spare was successfully used during commissioning!)
  - Redundant backend systems that could share load temporarily if necessary.
  - Plan to add second database server.
  - IBMS requires custom hardware — only one machine now that can handle it.
- All computers are under warranty.
- Agreement negotiated with Dell on-site support to provide routine maintenance for hardware issues to reduce the likelihood of sending a machine off-site for repair.

Planned hardware upgrades before next run



- Setting up test stand for software development.
- Adding redundant database server.
- Adding dedicated web server with SSO.
- Improving control room hardware (double memory and add SSDs).
- Adding additional control room machine.

Contributing Institutions and Responsibilities

Institution	Responsibilities	Personnel
University of Kentucky	AMC13 Readout, GPS, DAQ Hardware, IFIX integration, Online Systems Management	PI + PD (on site) + GS (on site)
University College London	Tracker readout, Online Management	PI (on site) + PD (on site) + GS (on site)
University of Washington	DQM, IBMS readout, Field, Nearline	GS + 2x GS + PD
Northern Illinois University	Slow controls	PI
Argonne National Lab	Field DAQ	PD (on site)
JINR, Dubna	Web interface, Event display	2x Scientist
Novosibirsk	Django Web Display	PD
University of Michigan	Field	GS
Shanghai	Database management	PD
Italy (Frascati + Napoli)	Laser system integration	PI + PD (on site) + PD + GS
Cornell	Auxiliary detector integration, Clock	PD + GS (on site)
Fermilab	Beamline integration	Scientist

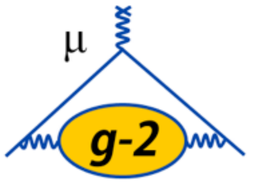
~24 total

RED = DAQ on-call

DAQ On-site support

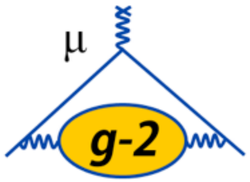
Level 1	Shifter tries to debug the problem.	← 2 shifters in control room at all times
Level 2	DAQ On-calls are called.	← 2 DAQ experts on-call at all times
Level 3	Local online experts (Wes and Becky) are called.	
Level 4	SLAM is paged by Wes or Becky.	← SLAM provides 24/7 support
Level 5	Dell on-site hardware support is contacted.	← Maintenance agreement to have computer hardware serviced locally.

High-rate tests



- During the commissioning run, we had data at ~0.1 Hz instead of 12 Hz, and the DAQ performed very well.
- We are using the laser calibration system to test the DAQ at 12 Hz.
- This data is being used for fine tuning our optimum DAQ settings for data collection such as the Q-method decimation and flush rate.
- We have observed that the DAQ typically runs for ~5 hours without human intervention, punctuated by TCP/IP errors and WFD5 errors that are fixed by restarting the offending process.
- Efforts to resolve these errors and improve uptime are in progress.

Summary



- MIDAS is working very well as the DAQ software for Muon g-2.
- The DAQ hardware includes 17 frontend machines, 5 backend machines, 2 dedicated near line analysis machines, 3 computers for slow control, 3 servers, and 24 beagle bones running 67 MIDAS frontends.
- We take an input data rate of 20 GB/s, and reduces that to < 200 MB/s in the event builder via GPU processing and lossless compression.
- We have a clear plan for monitoring the beam and data quality and the associated infrastructure has been tested.

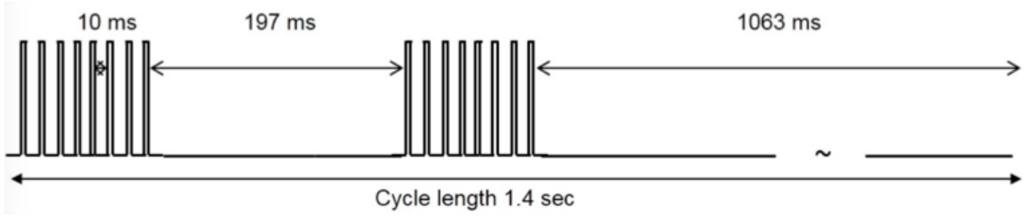
backup

Outline

- Overview of DAQ/DQM Systems (and nearline?)
- Alarm system
- Data path and storage
- Slow controls/databases
- DAQ health monitoring
- What works
- What we have done to fix the things that didn’t
- Interface between online/offline systems
- High rate tests
- Hardware redundancy plan + DAQ maintenance agreement
- Hardware upgrades planned before next run
- Personnel needed and SLAM contribution

Rate requirements

- Accommodate 12 Hz average rate of muon fills that consist of sequences of eight successive 700 μs fills with 10 ms fill-separations.

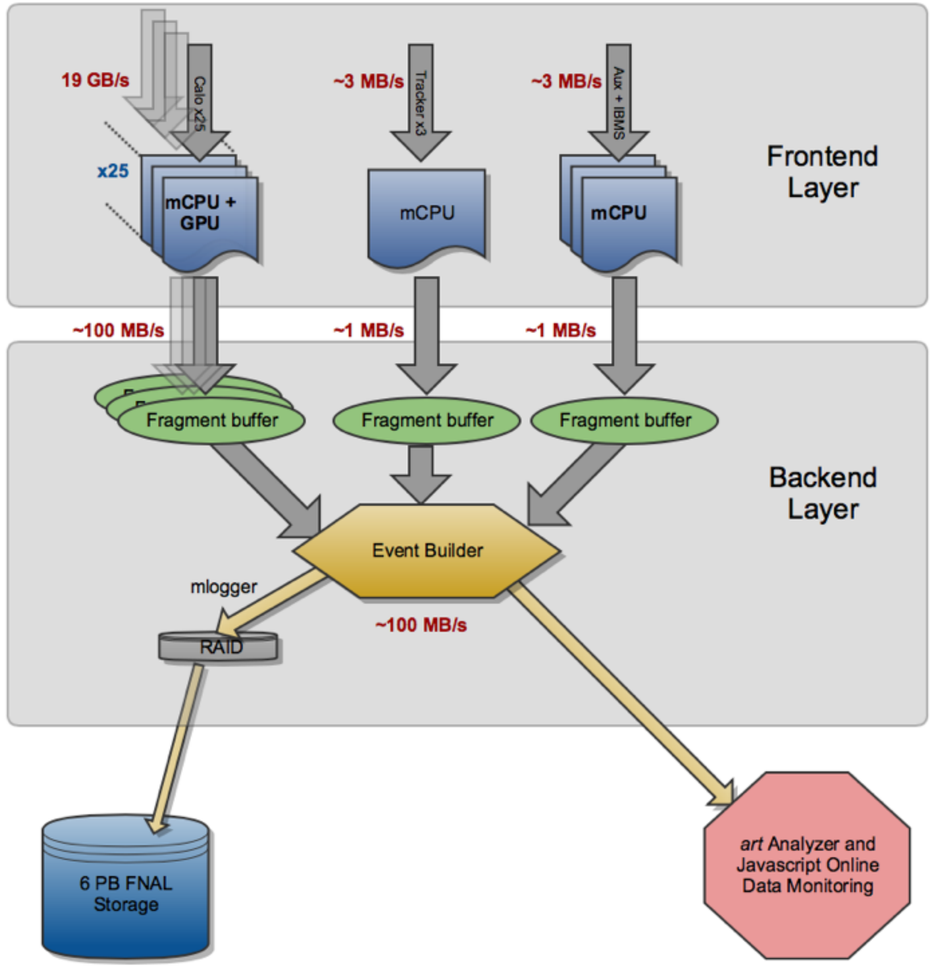


- Time-averaged rate of raw ADC samples is 20 GB/s, which must be reduced by a factor of 100.
- Data is processed in GPUs to accomplish this task.
- Total data on tape after 2 years of running will be 7 PB.

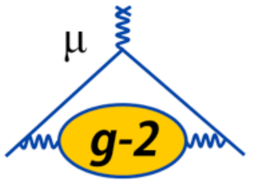
Source	MB Per Fill	MB Per Second
Raw data	1,600	19,400
T-Method	9.4	112.5
Q-Method	4.0	48.5
Prescaled Raw	1.6	19.4
Tracker	0.75	9
Laser Monitor	0.08	1
Auxiliary	0.33	4
Event Builder:	**16.2**	**194.4**

DAQ Design

- Layered array of commodity, networked processors
- Frontend layer for readout of detectors.
- Backend layer for assembly of event fragments.
- Slow control layer.
- Online analysis layer using *art*+JS.
- Field DAQ operates independently, but with a similar design.



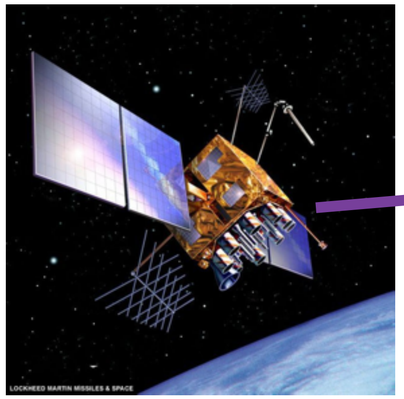
Input sources



- Digitization is performed in custom uTCA based waveform digitizers.
- Each digitizer runs at 800 MSPS, so each time bin is 1.25 ns, and a 700 us fill is 560,000 clock ticks.
- Each uTCA crate contains 12 WFD5s or 60 channels of digitization.
  - Crate 0 reads data from the clock and control center (CCC)
  - Crates 1-24 each read data from one calorimeter (+ spare channels)
  - Crate 25 reads data from the laser system
  - Crate 26 reads data from the Auxiliary detectors (Harps, Quads, and Kickers)
  - Crate 27 reads data from the three tracker detectors.
- Data from each crate is sent to a DAQ computer via a dedicated 10 Gb fiber. The total data rate is 20 GB/s.
- The data is then processed in Nvidia K40 GPUs.

Master frontend

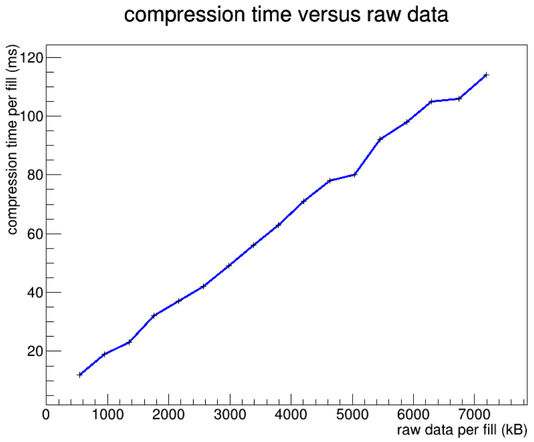
- Communicates with other frontends using RPC calls.
- Provides begin of run and end of run RPCs to all frontends.
- Provides end of fill trigger to synchronous frontends.
- Configures clock and control system.
- Reads trigger times from Meinberg GPS unit and writes them to a MIDAS bank.

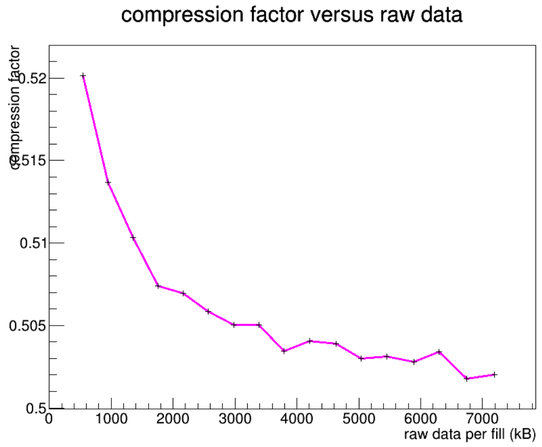


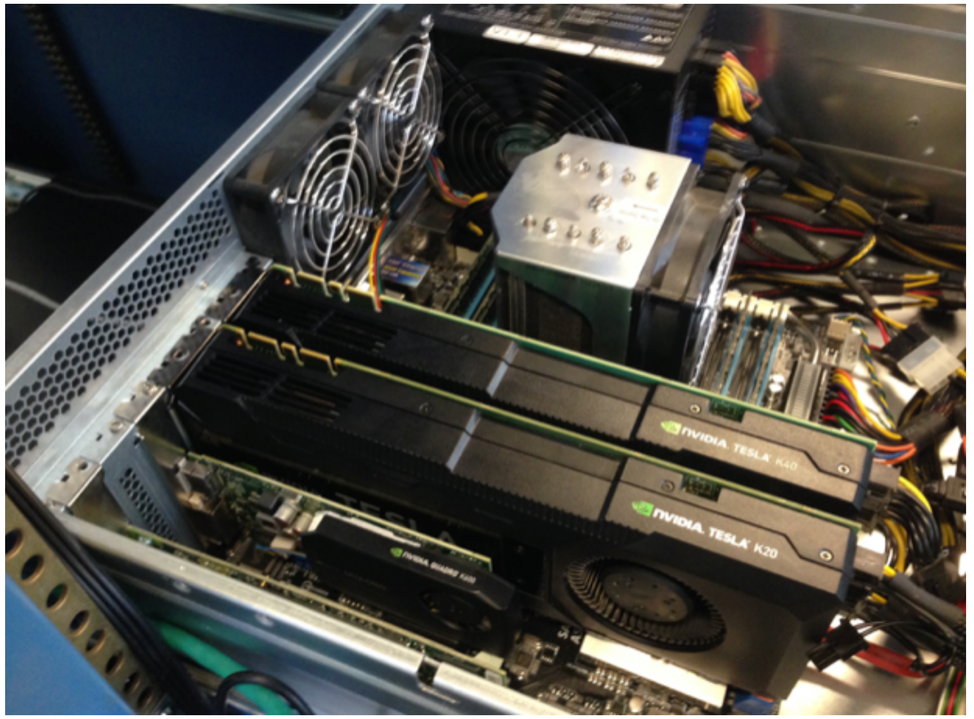
```
Bank:GPS0 Length: 20(I*1)/5(I*4)/5(Type) Type:Unsigned Integer*4
   1-> 0x000018c2 0x580838bc 0x5dcabfb0 0x580838bc 0x5dd48308
```

AMC13 Frontend

- Main frontend for processing data from calorimeters, laser, fiber harps, quads, and kickers.
- Each frontend process reads data from one uTCA crate over 10 Gb ethernet with TCPIP.
- Data is processed in Nvidia Tesla K40 GPUs using CUDA code that is integrated into the frontend.
- Midas banks are losslessly compressed using zlib.
- Full configuration of the uTCA crate is performed via the MIDAS ODB.

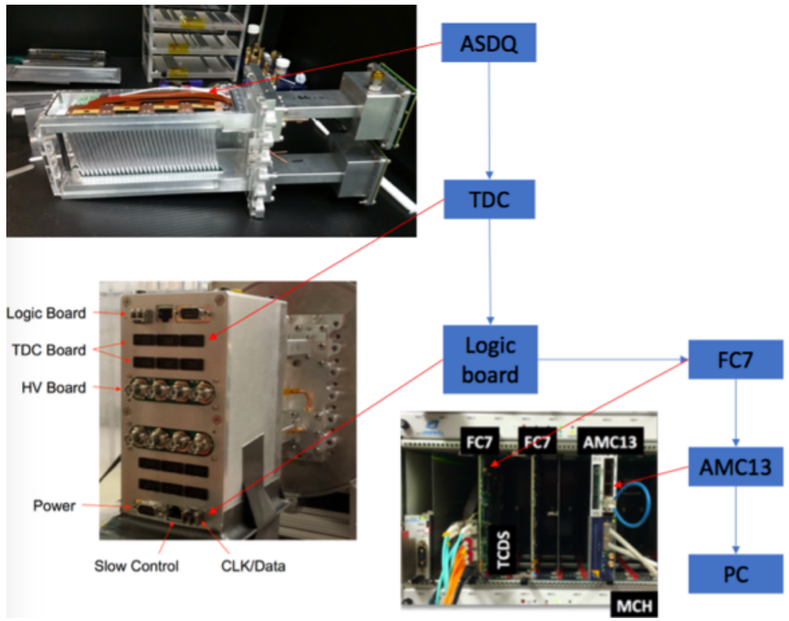






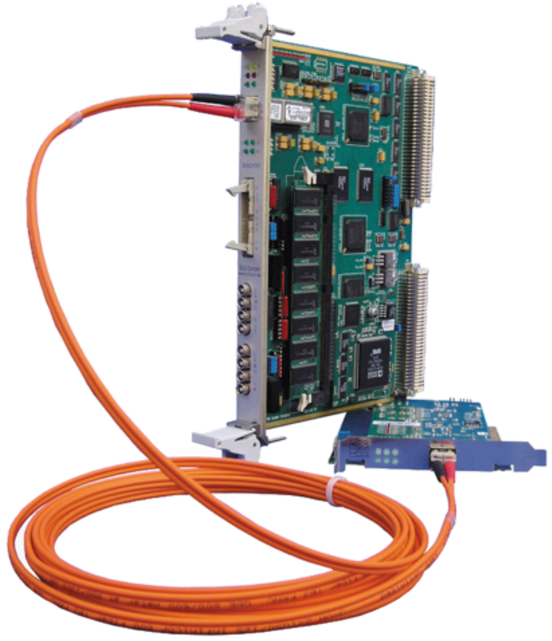
Tracker Frontend

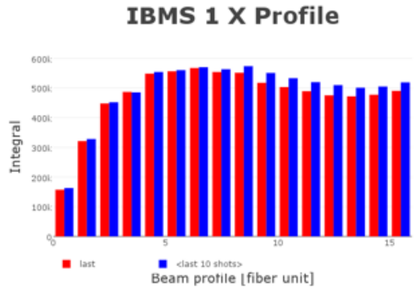
- Three tracker stations will be read via one uTCA crate.
- Reads data from AMC13.
- Instead of digitizers, data comes from multihit TDCs that are read via FC7 cards.

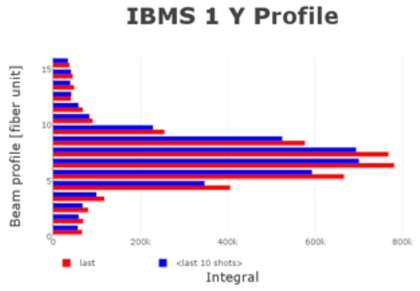


IBMS Frontend

- Data from the inflector beam monitoring system (IBMS) is read out via a CAEN digitizer.
- A custom MIDAS frontend was written to integrate this detector into the DAQ.



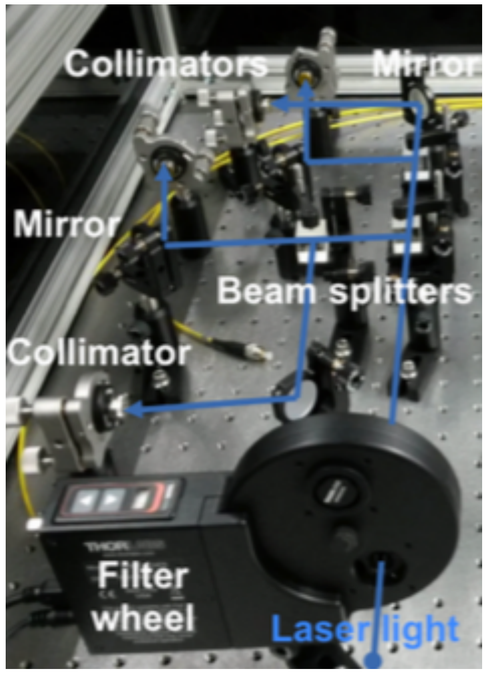




MIDAS Sequencer

- The sequencer was used extensively for calibration runs such as filter wheel scans and bias voltage scans.

- A typical sequence would be:
  - Execute script to move wheel.
  - Update ODB values
  - Take data for 10 minutes
  - Repeat



Progress

Sequence is finished

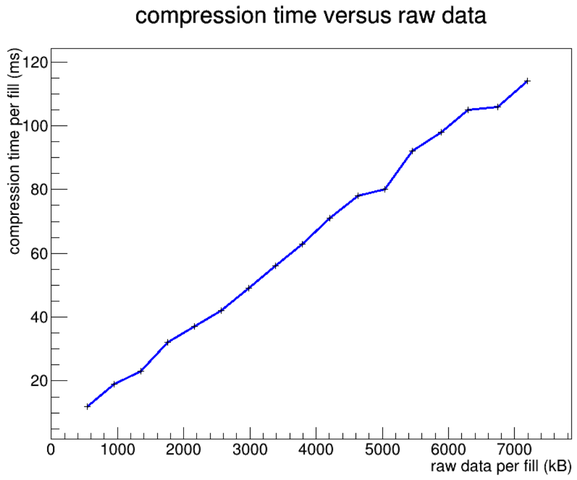
Sequencer File

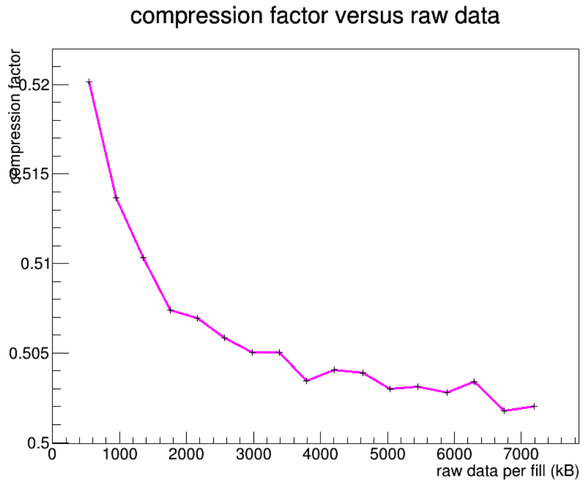
Filename:wheel_setting.msl Show XML

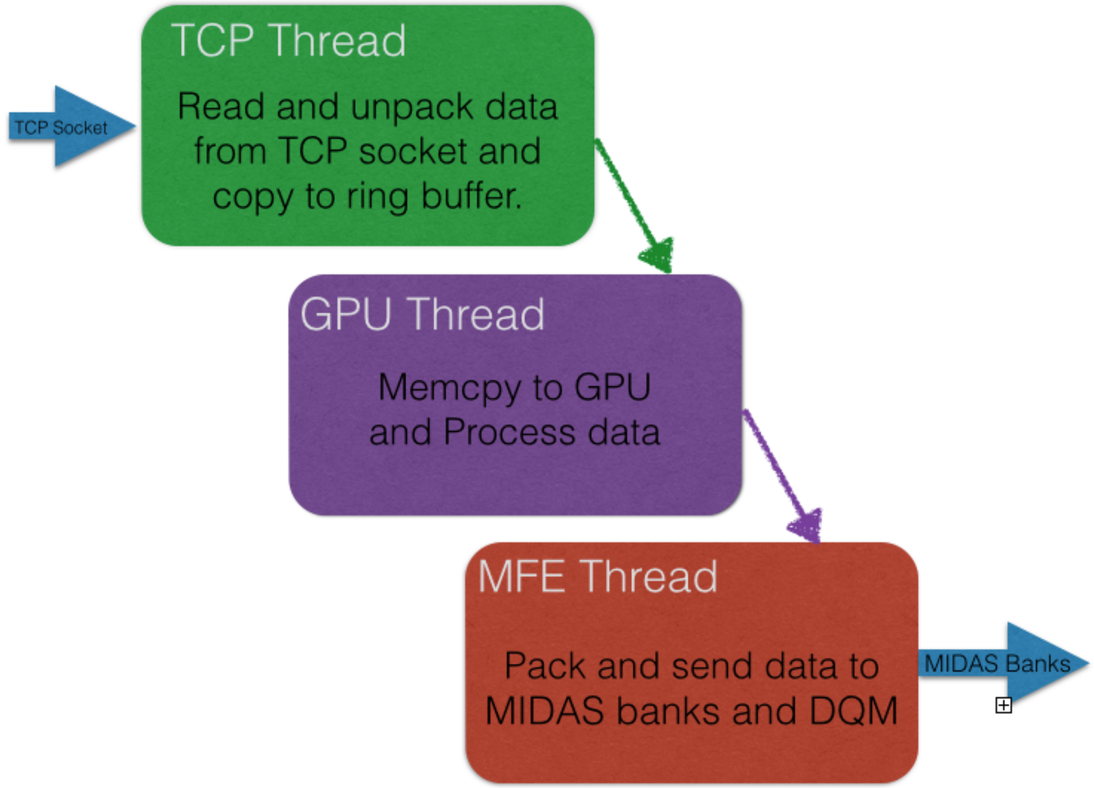
```
 1 COMMENT "Run a sequence of laser runs"
 2 RUNDESCRIPTION "Calibration run"
 3
 4 LOOP angle, 1, 2, 3, 4, 5, 6, 7, 1
 5       ODBSET "/Experiment/Edit on start/Comment (256 max)", "Automated run"
 6       ODBSET "/Experiment/Edit on start/Quality YNCT", "C"
 7       SCRIPT /home/daq/test_wheel.sh, $angle
 8       WAIT Seconds 2
 9       TRANSITION START
10       WAIT Seconds 200
11       TRANSITION STOP
12 ENDLOOP
13 ODBSET "/Experiment/Edit on start/Comment (256 max)", "Enter comment"
14 ODBSET "/Experiment/Edit on start/Quality YNCT", "T"
```

Calorimeter-GPU Frontend

- Each frontend process reads data from one uTCA crate over 10 Gb ethernet with TCPIP.
- Frontend is multithreaded with mutex locks.
- Data is processed in Nvidia Tesla K40 GPUs using CUDA code that is integrated into the frontend.
- Midas banks are losslessly compressed using zlib.
- Full configuration of the uTCA crate is performed via the MIDAS ODB.





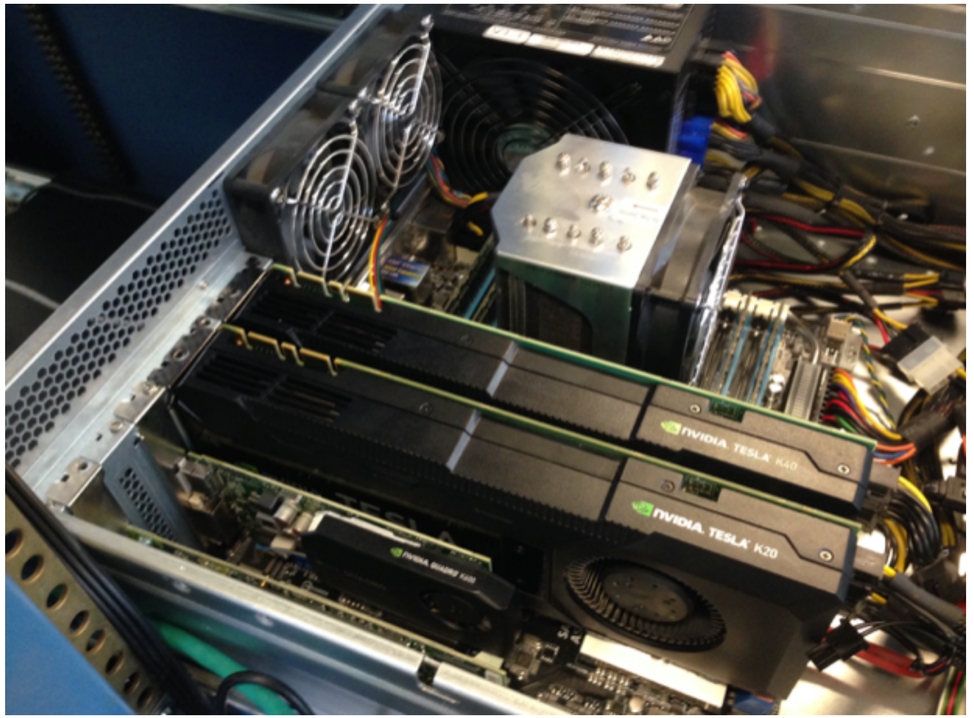


Why GPUs?

- The GPUs dramatically improve performance by parallelizing processing.
- Technology was developed for commercial applications, so it is well supported.
- Easier to code in C++ than to learn specialized language.
- Without GPUs, we could not keep up with our data rates.

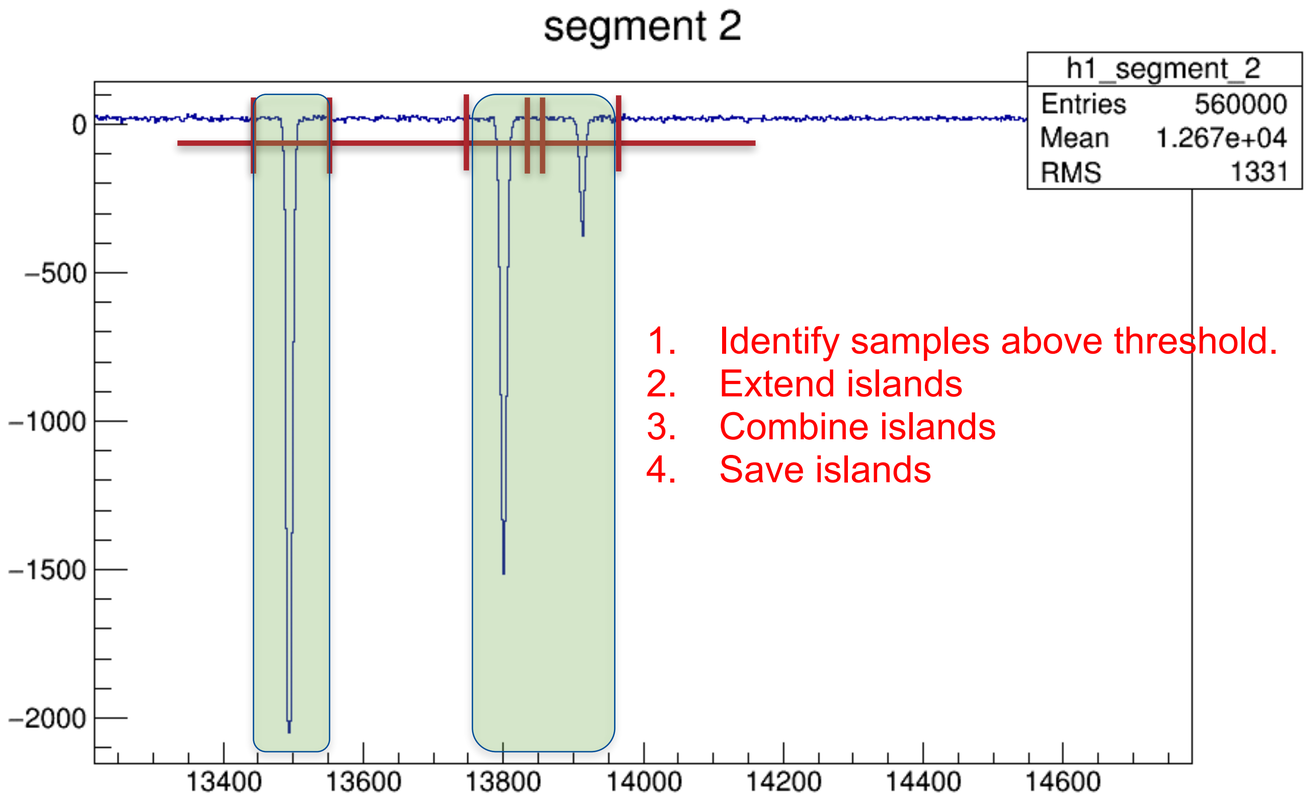
GPU Processing

- The frontend includes CUDA routines for data processing.
- Each GPU processes data from one calorimeter.
- Raw fill is copied to GPU memory, where it is reduced using T-method (island chopping), Q-method (histogramming), pedestal calculation, and template fitting.
- The output of each process is written in one MIDAS bank.



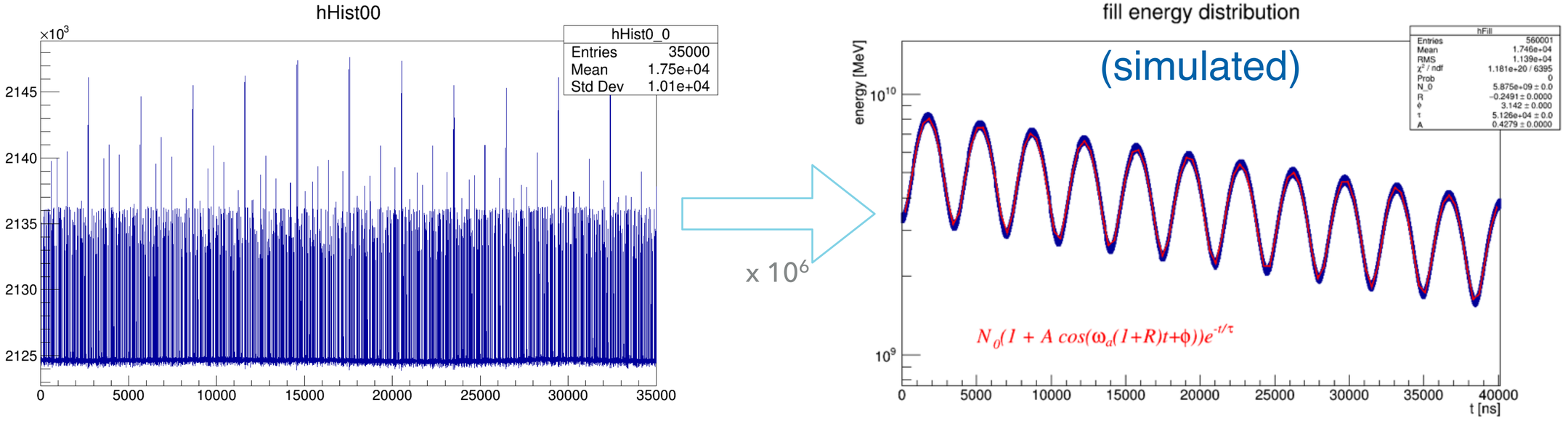
T-Method

- Identify and save regions of the waveform containing positron hits.
- A typical waveform will have ~180 islands.



Q-method

- Full waveforms are decimated in time and summed over many fills to create a histogram that is saved in the data file.
  - i.e. If we decimate in time by 10 and flush every 100 fills, we reduce the data rate by a factor of 1000, so from 20 GB/s to 20 MB/s.
- Use smaller bins at lower times and wider bins at later times to insure that we can extract the pedestal.



GPU Template Fits

- Fit templates are loaded into the GPU memory and used to fit each peak above a certain threshold.
- A bank containing the fit results will be saved for each fill in addition to the chopped islands.
- Reduces the processing necessary in the online DQM.