Midas DAQ for Muon g-2 at Fermilab

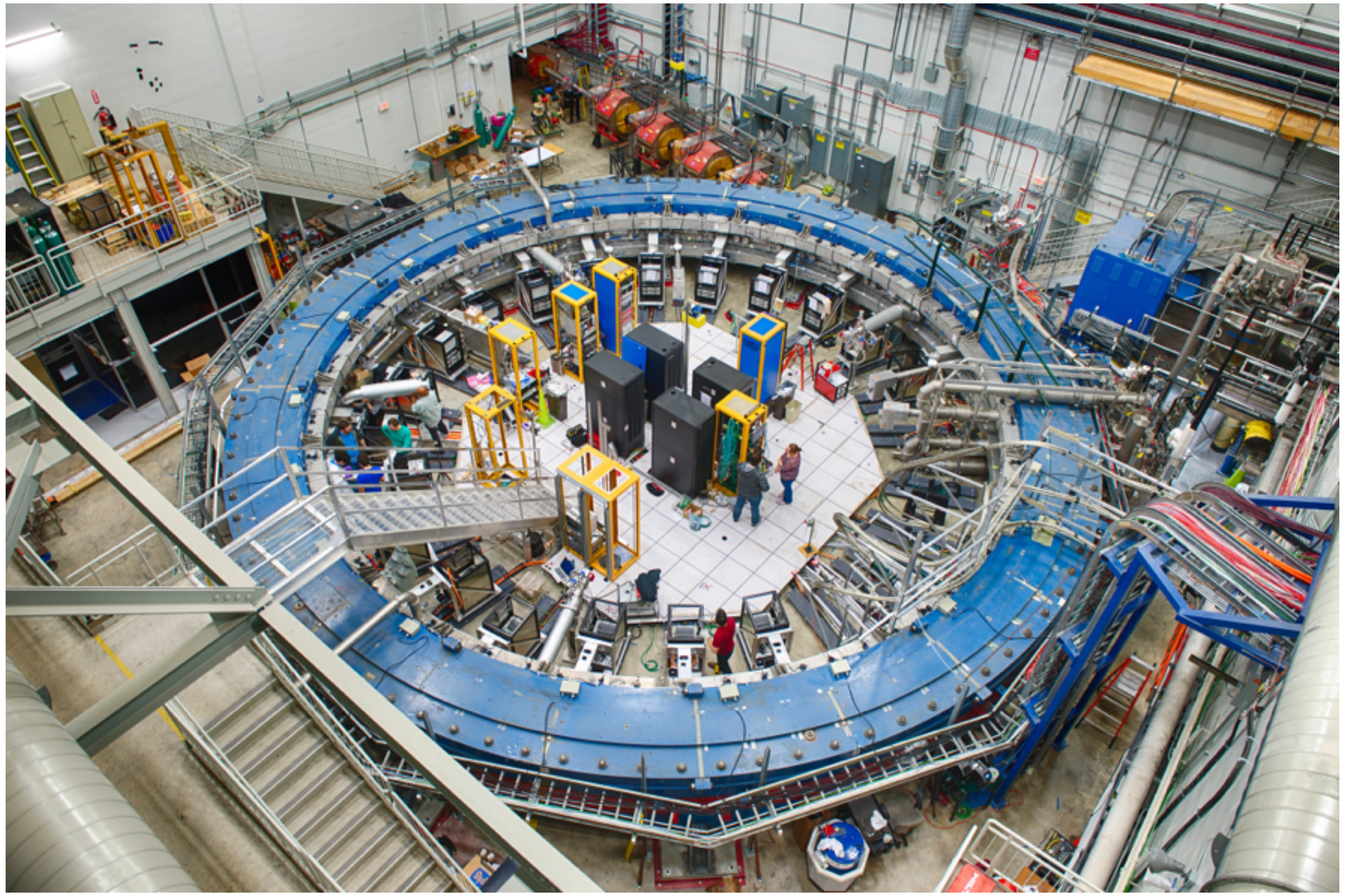
Wes Gohn

TRIUMF Midas Seminar

26 July 2017

Muon g-2 Experiment Overview

- Goal is to measure the anomalous magnetic moment of the muon to 140 ppb, which is a factor of 4 better than has been previously measured.
- Muon fills are injected into the ring at a rate of 12 Hz.
- The precession frequency of the muons is measured by detecting decay positrons in 24 segmented calorimeters inside the ring.
- We have just finished a five-week commissioning run.



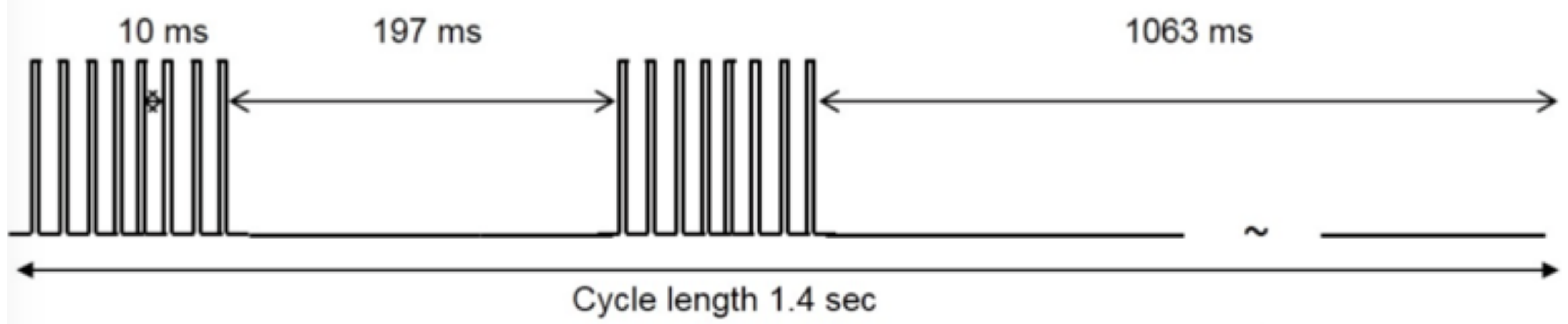
Fermilab

DAQ Input Sources

- 24 Calorimeters
  - 1 uTCA crate for each calorimeter
  - 54 channels * 24 calos = 1296 channels of digitized data.
  - Data processed by 12 Cornell WFD5s and sent from AMC13.
- 4 Fiber Harps
  - 7 channels * 4 harps = 28 channels
  - Data processed by Cornell WFD5s
- Quads and Kickers
  - Write 4 quad channels and 15 kicker channels
  - Data processed by Cornell WFD5s
- 3 Trackers
  - Data from Multihit TDCs sent from FC7s in a uTCA crate
- IBMS and quads
  - Running on CAEN digitizers

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.

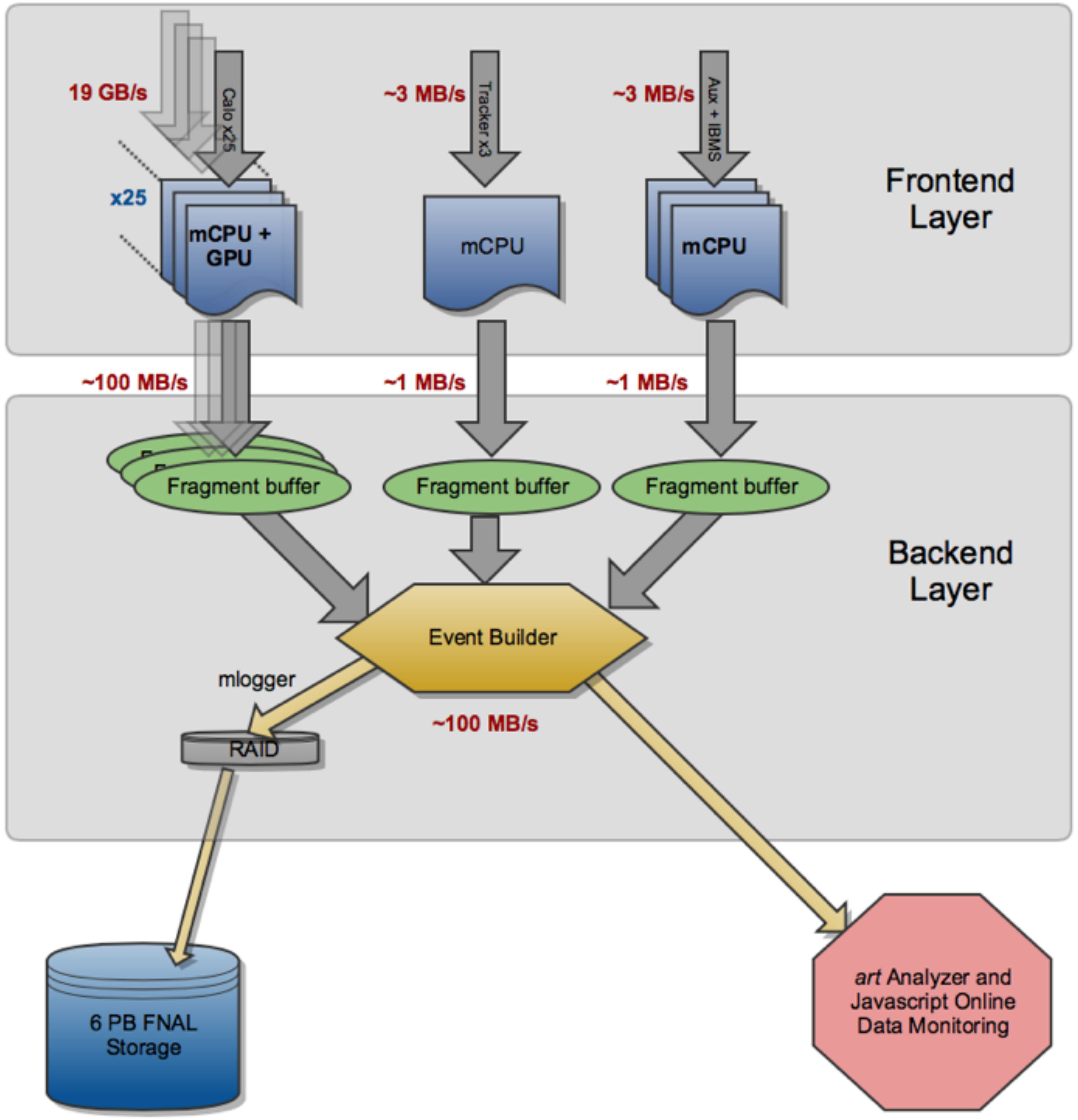


- Time-averaged rate of raw ADC samples is 20 GB/s, which must be reduced by a factor of 100.
- Data is processed in GPUs to accomplish this task.
- Total data on tape after 2 years of running will be 10 PB.

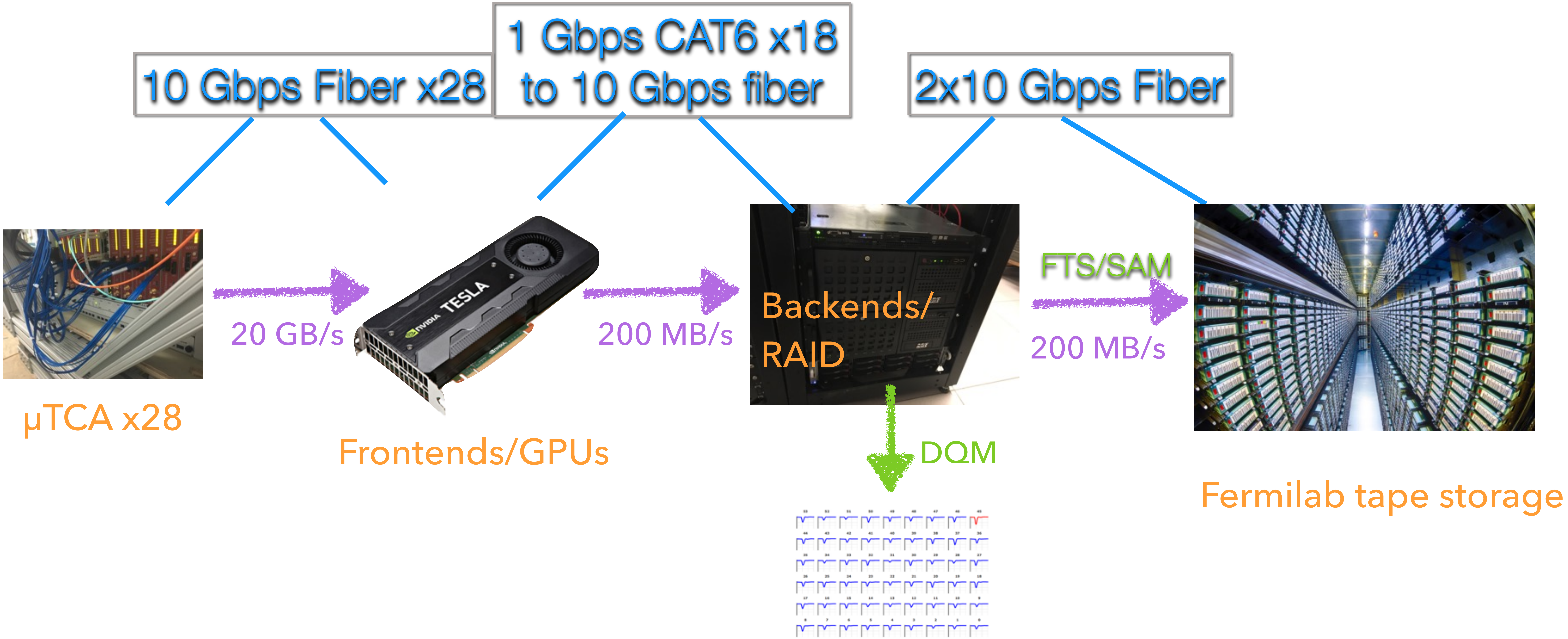
Source	MB Per Fill	MB Per Second
Raw data	1,600	19,400
T-Method	9.4	112.5
Q-Method	4.0	48.5
Prescaled Raw	1.6	19.4
Tracker	0.75	9
Laser Monitor	0.08	1
Auxiliary	0.33	4
Event Builder:	**16.2**	**194.4**

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.

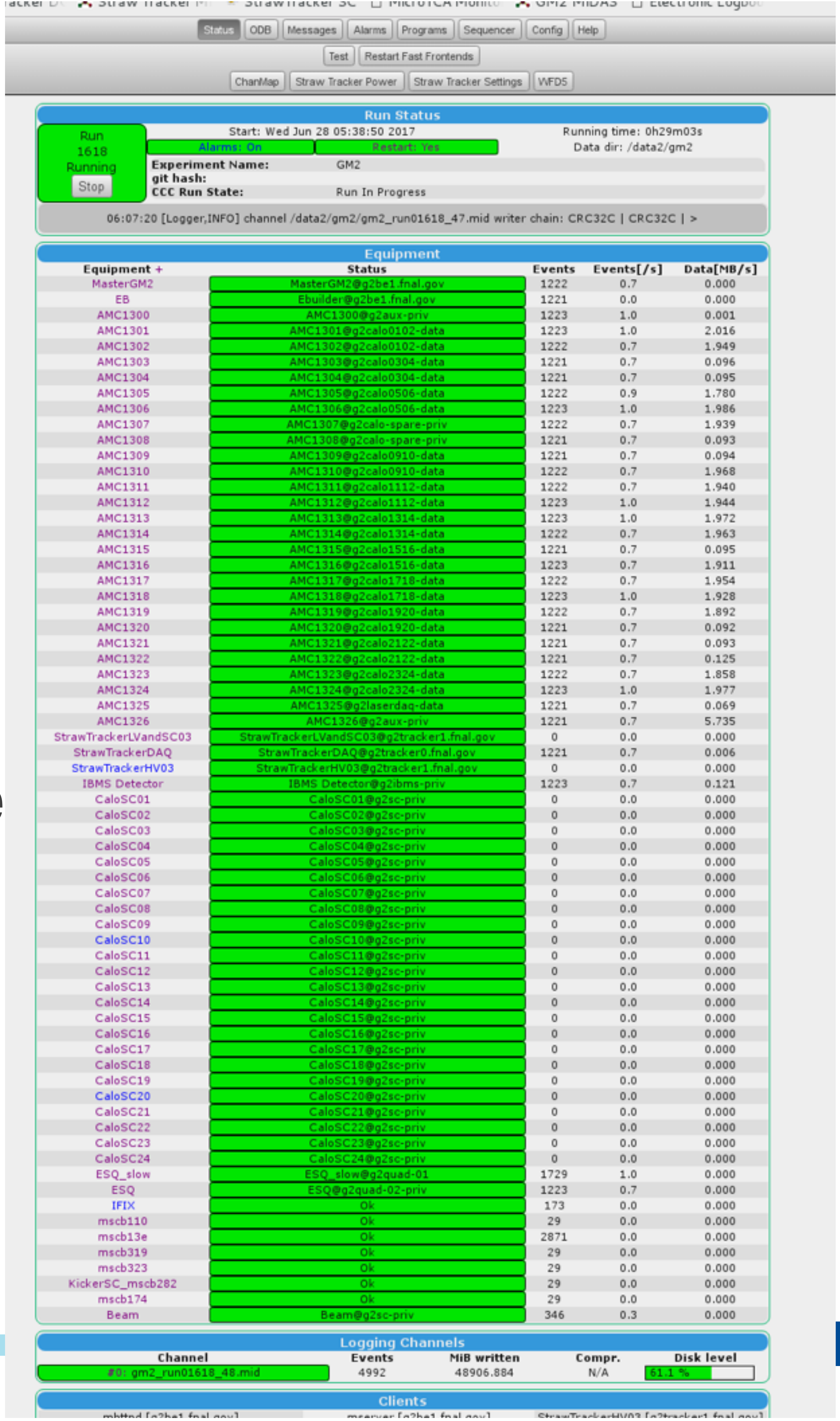


DAQ Architecture



MIDAS configuration

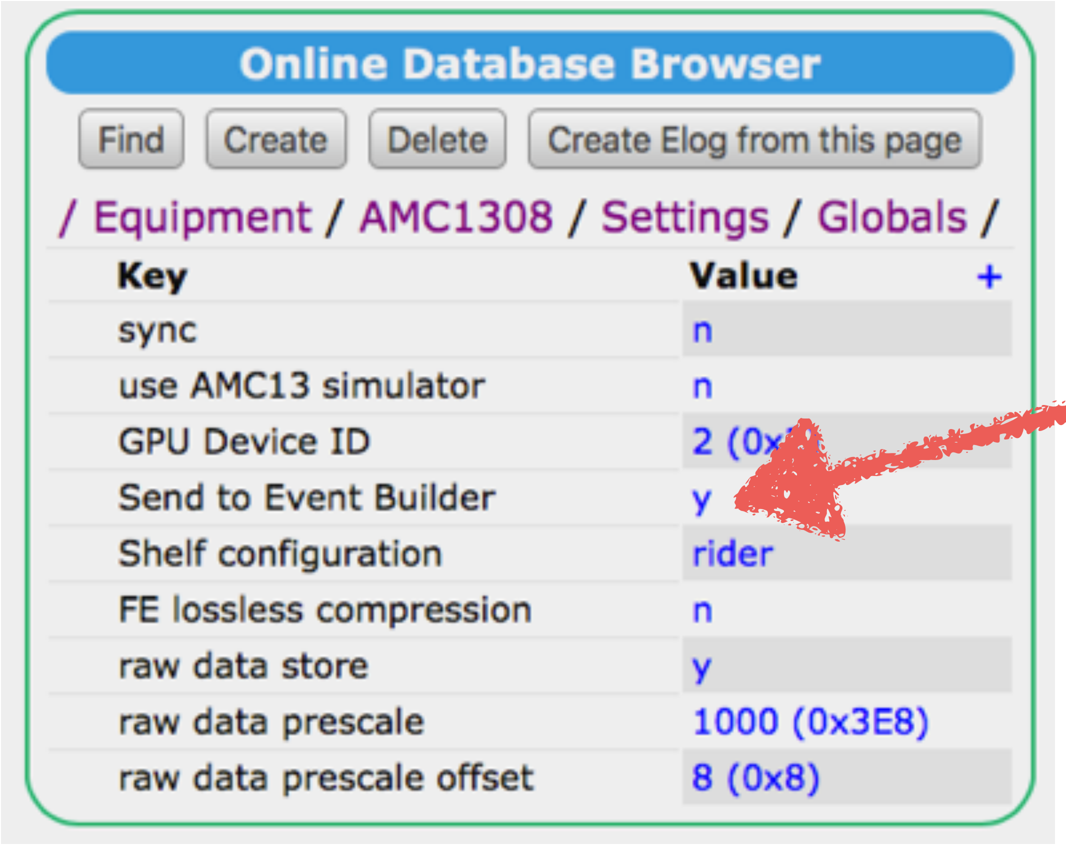
- 32 fast frontends (data at beam fill rate).
- 35 slow control frontends.
- Midas alarm system.
- Midas sequencer used for calibration runs.
- ODB dumped to JSON file and saved to Postgres database at each end of run.
- Online analyzer using *art* and javascript (se talk by A. Fienberg).
- Separate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



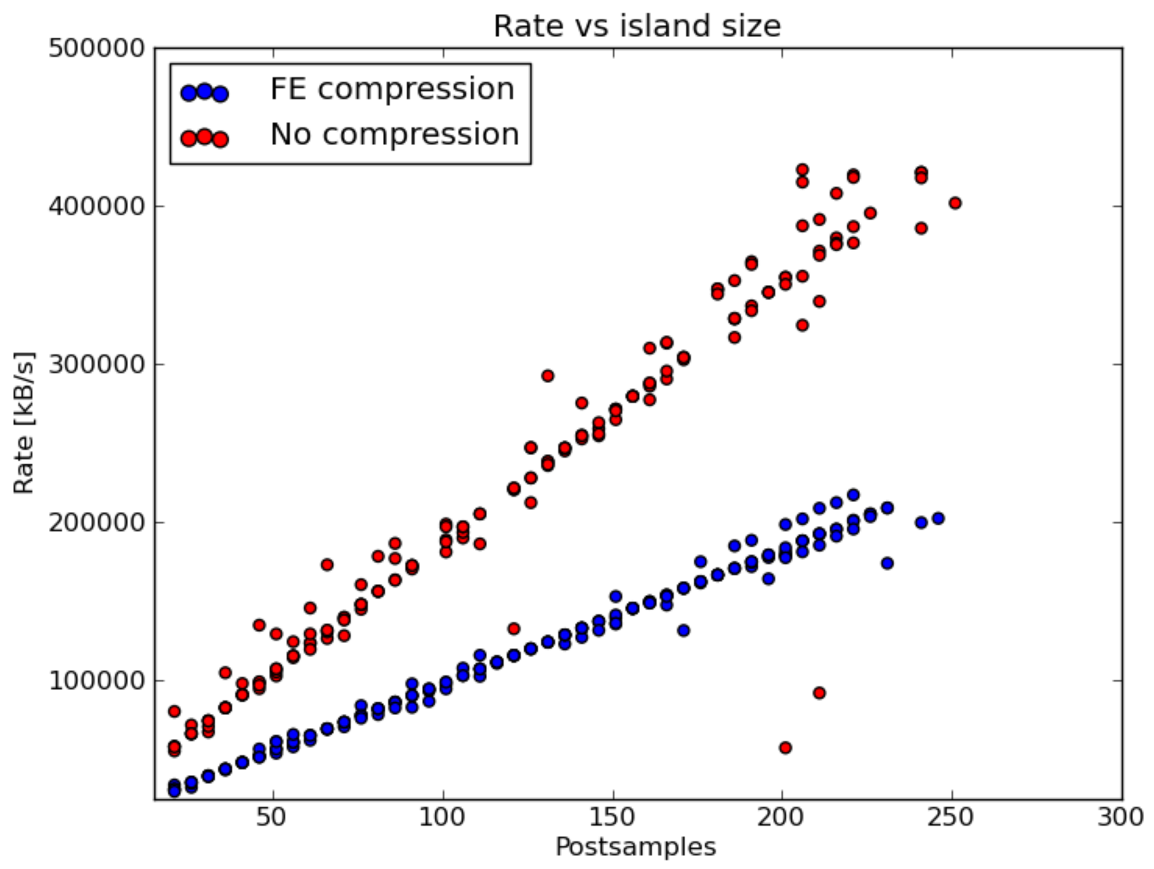
Fermilab

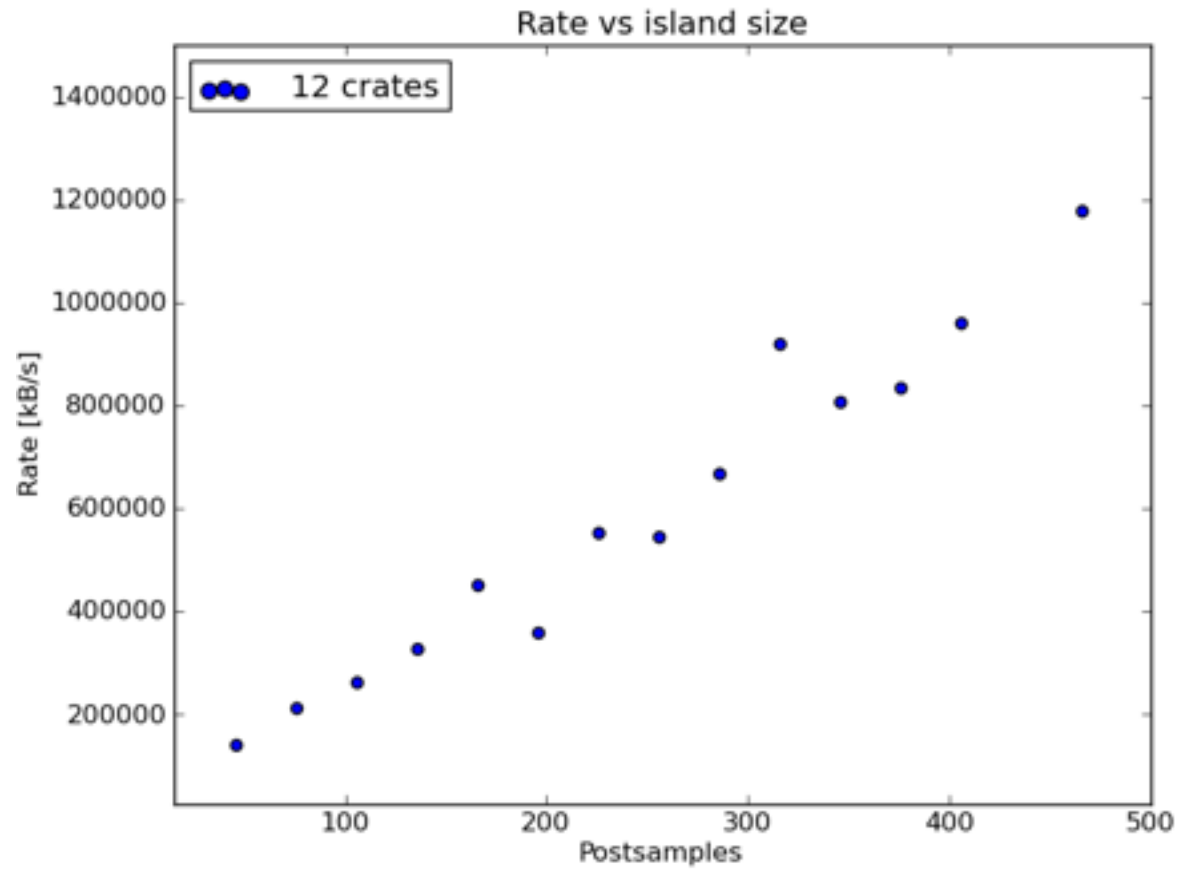
Event builder

- Modified event builder to change how it is enabled — now done entirely in ODB Equipment of each frontend.



Enable here!

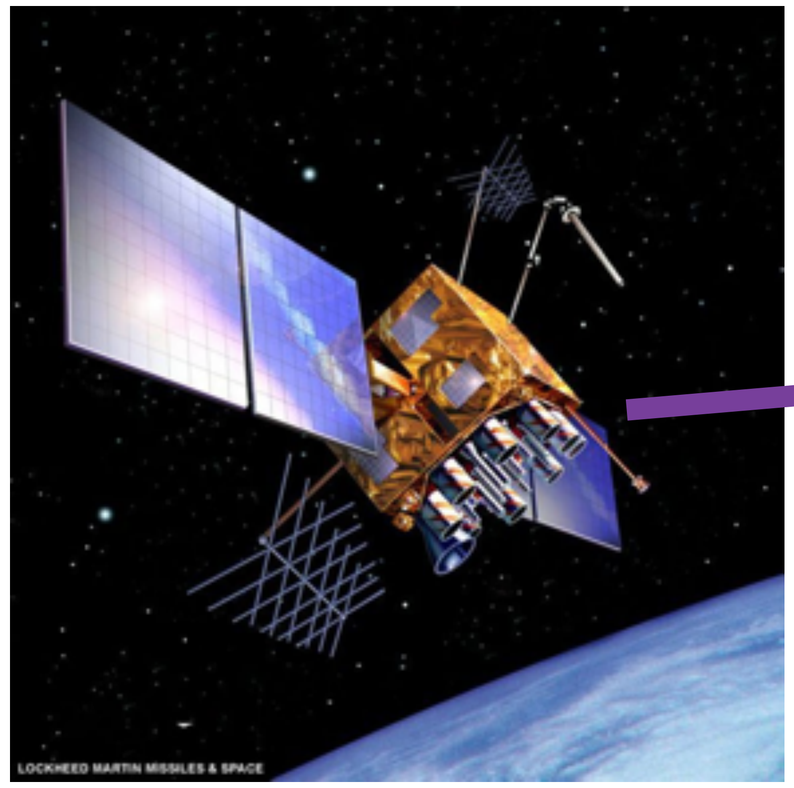




- Total EB rate maxed out at > 1.2 GB/s (limited by network bandwidth)
- The event builder combines up to 270 banks for each event.

MasterGM2 frontend

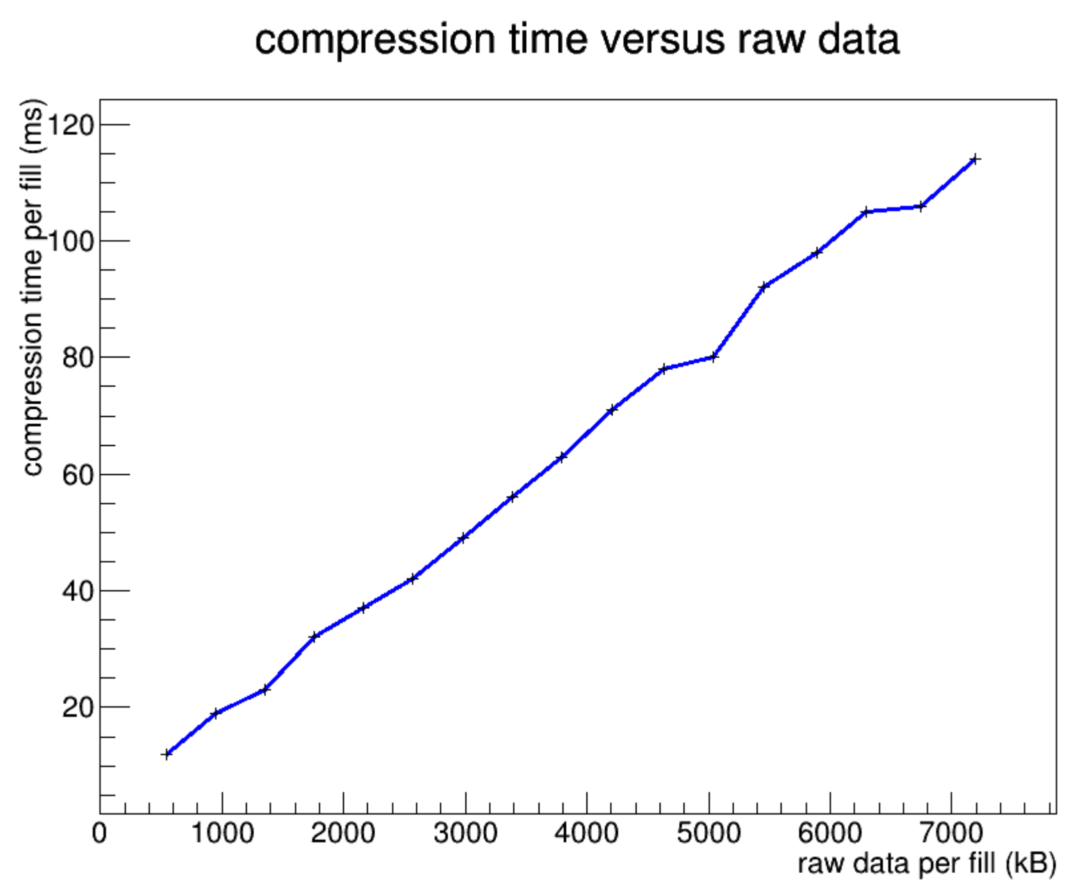
- Communicates with other frontends using RPC calls.
- Provides begin of run and end of run RPCs to all frontends.
- Provides end of fill trigger to synchronous frontends.
- Configures clock and control system.
- Reads trigger times from Meinberg GPS unit and writes them to a MIDAS bank.

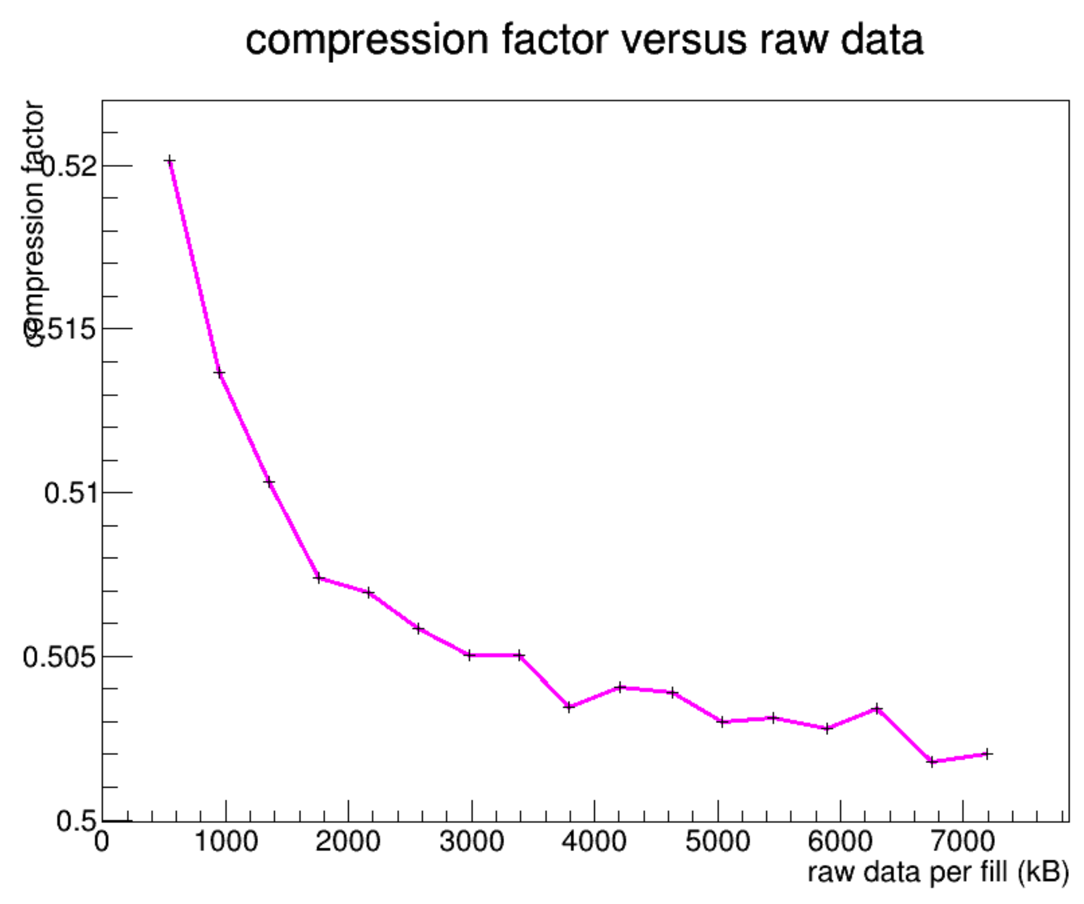


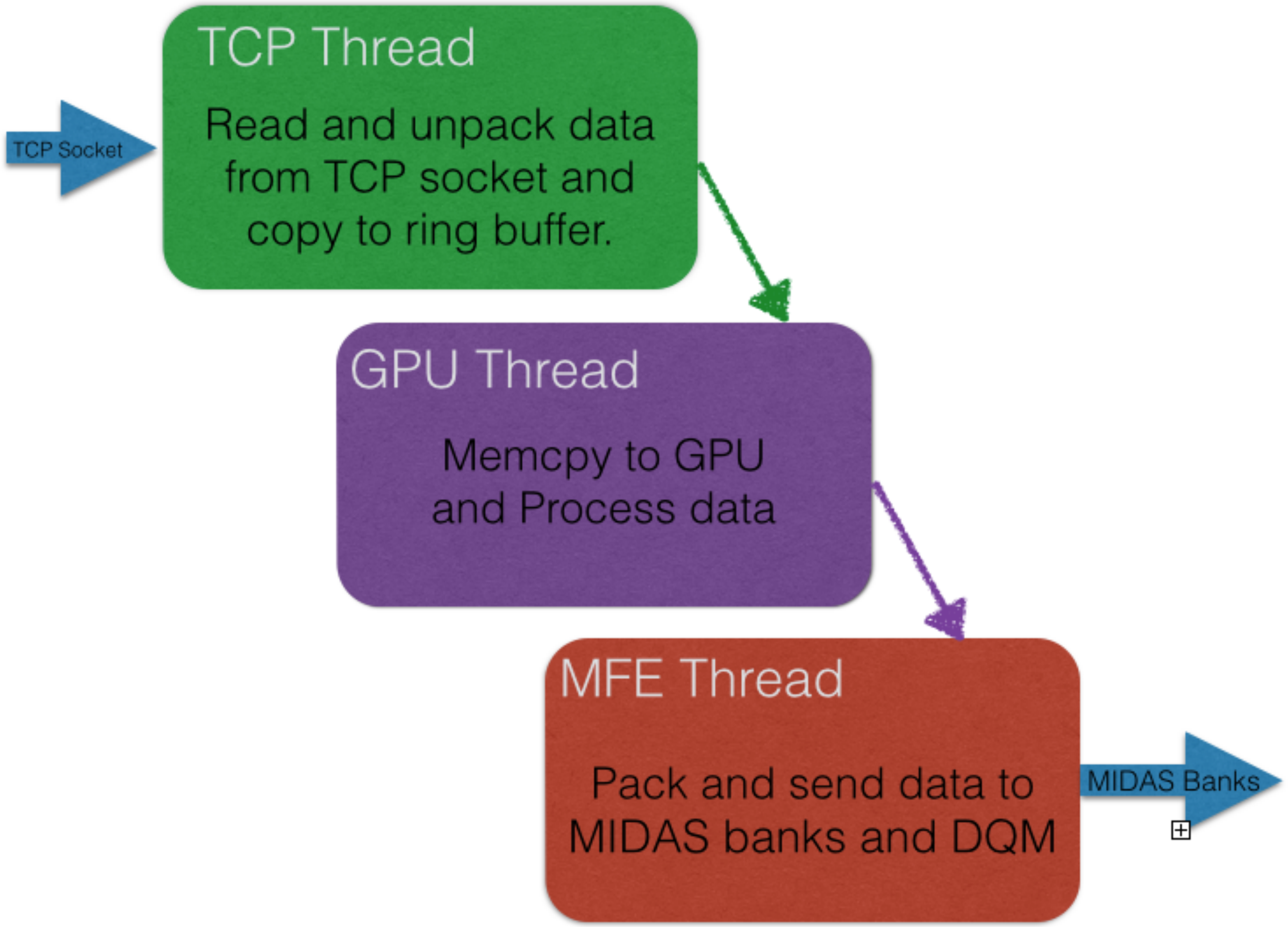
```
Bank:GPS0 Length: 20(I*1)/5(I*4)/5(Type) Type:Unsigned Integer*4
   1-> 0x000018c2 0x580838bc 0x5dcabfb0 0x580838bc 0x5dd48308
```

AMC13 Frontend

- 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.
- Frontend is multithreaded with mutex locks.
- Full configuration of the uTCA crate is performed via the MIDAS ODB.

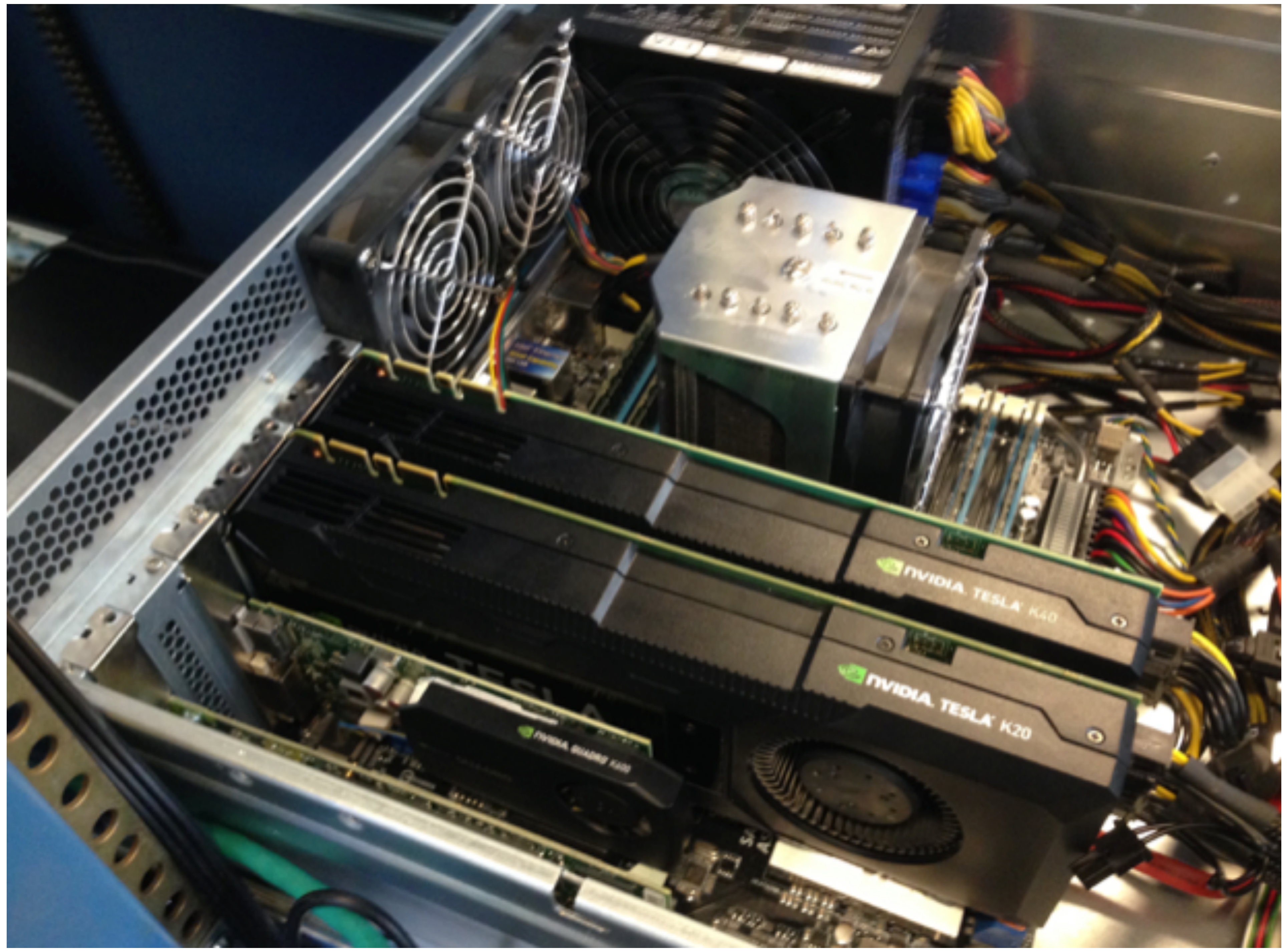






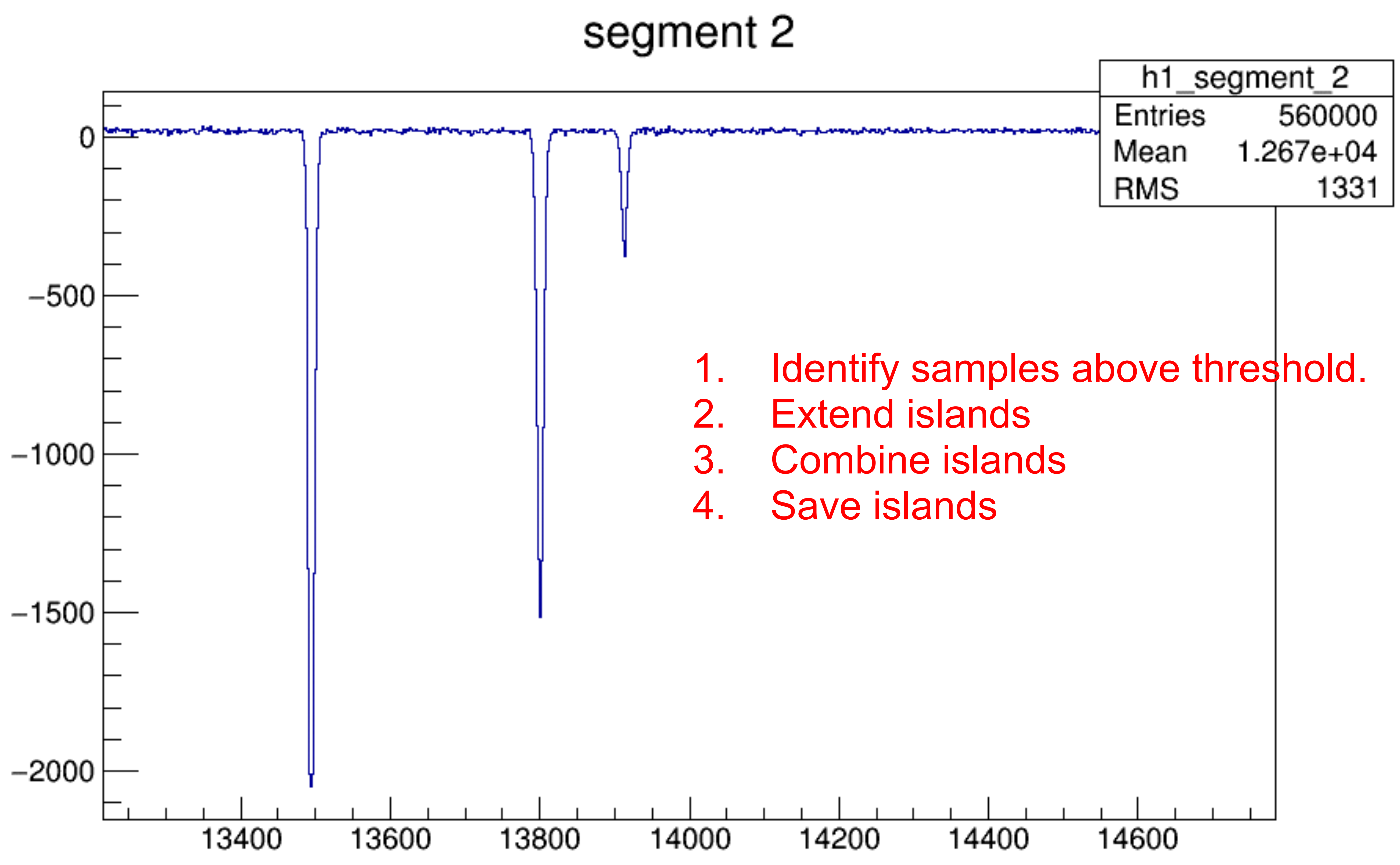
GPU Processing

- 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.



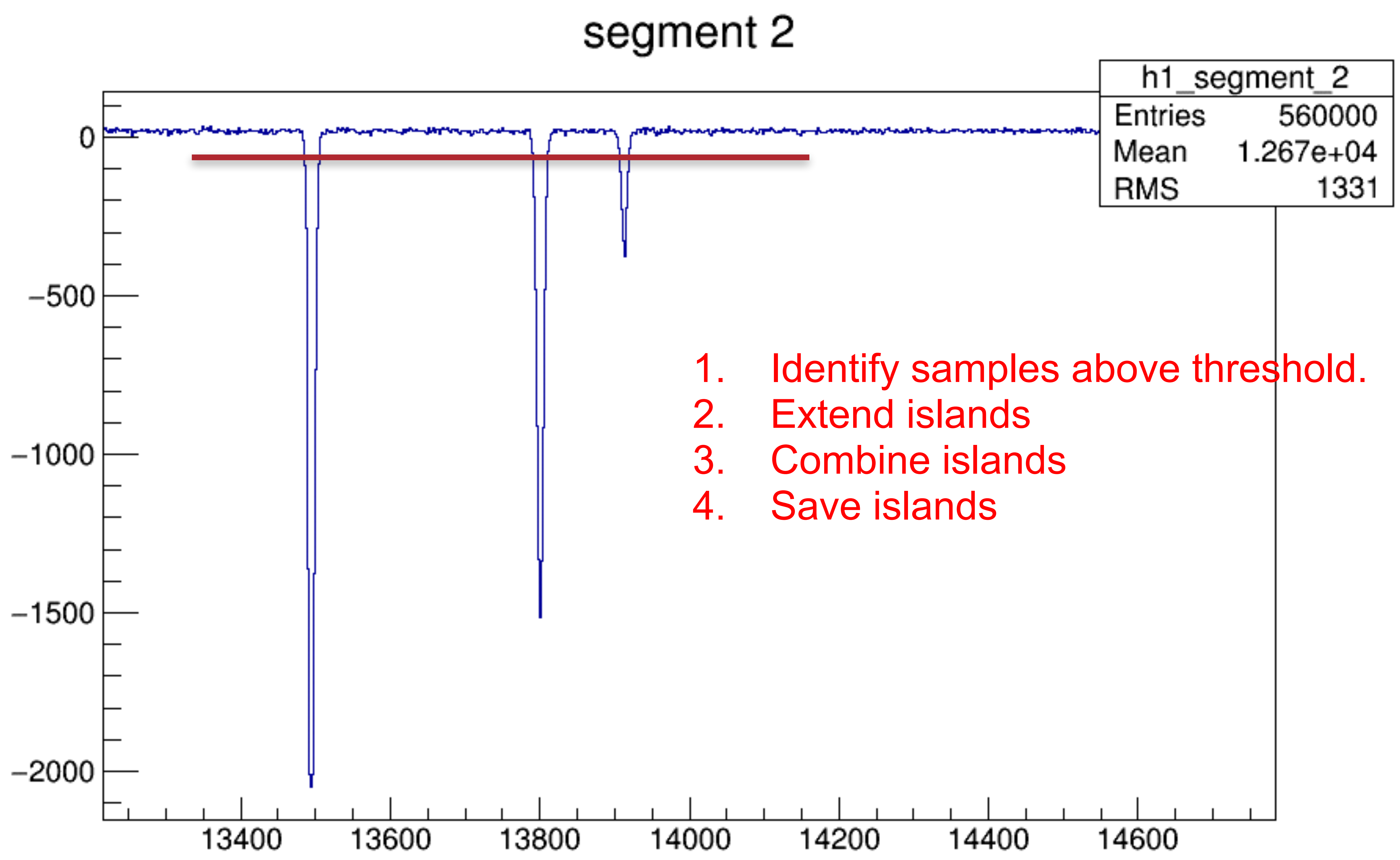
T-Method

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



T-Method

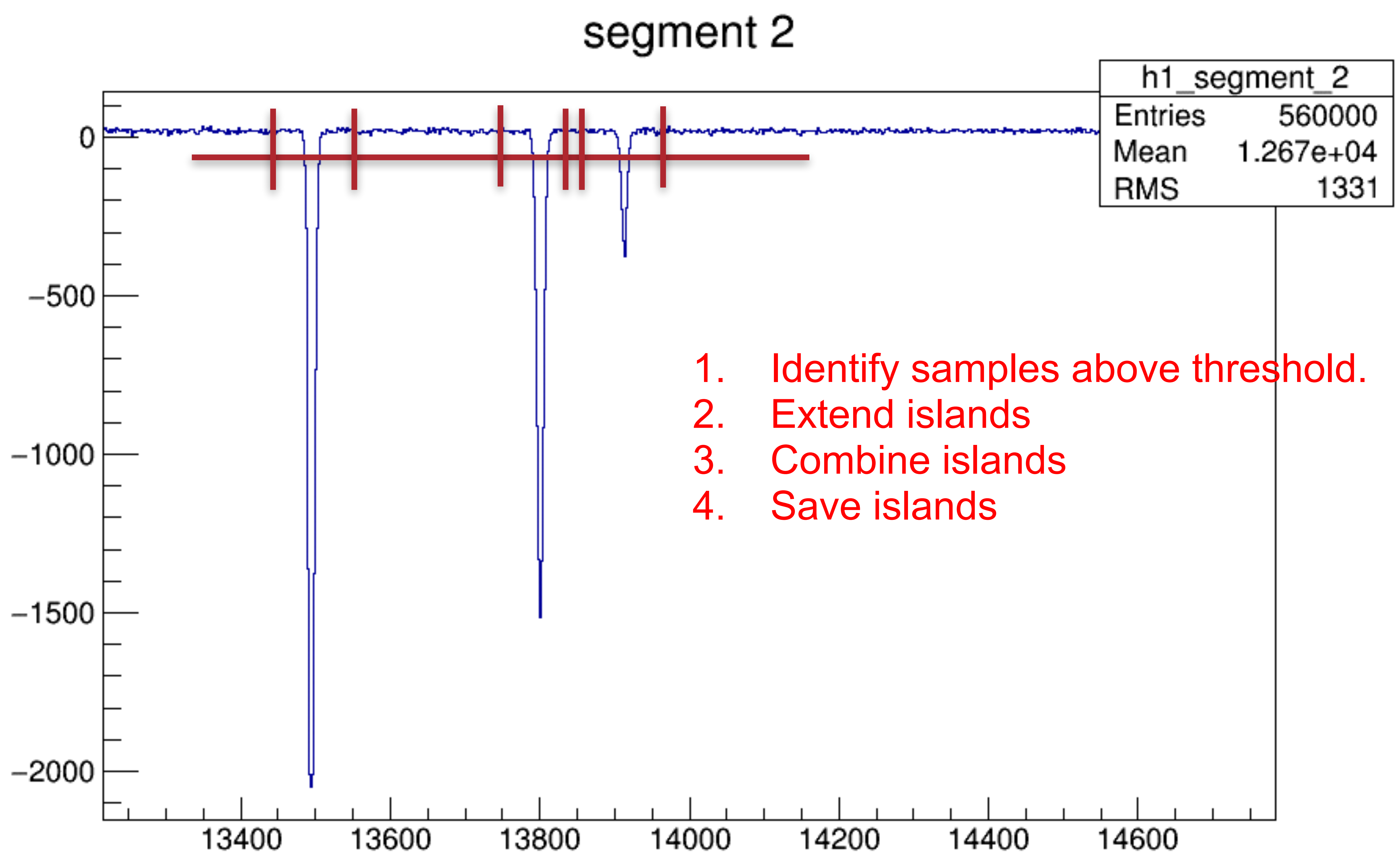
- Identify and save regions of the waveform containing positron hits.
- A typical waveform will have ~180 islands.



1. Identify samples above threshold.
2. Extend islands
3. Combine islands
4. Save islands

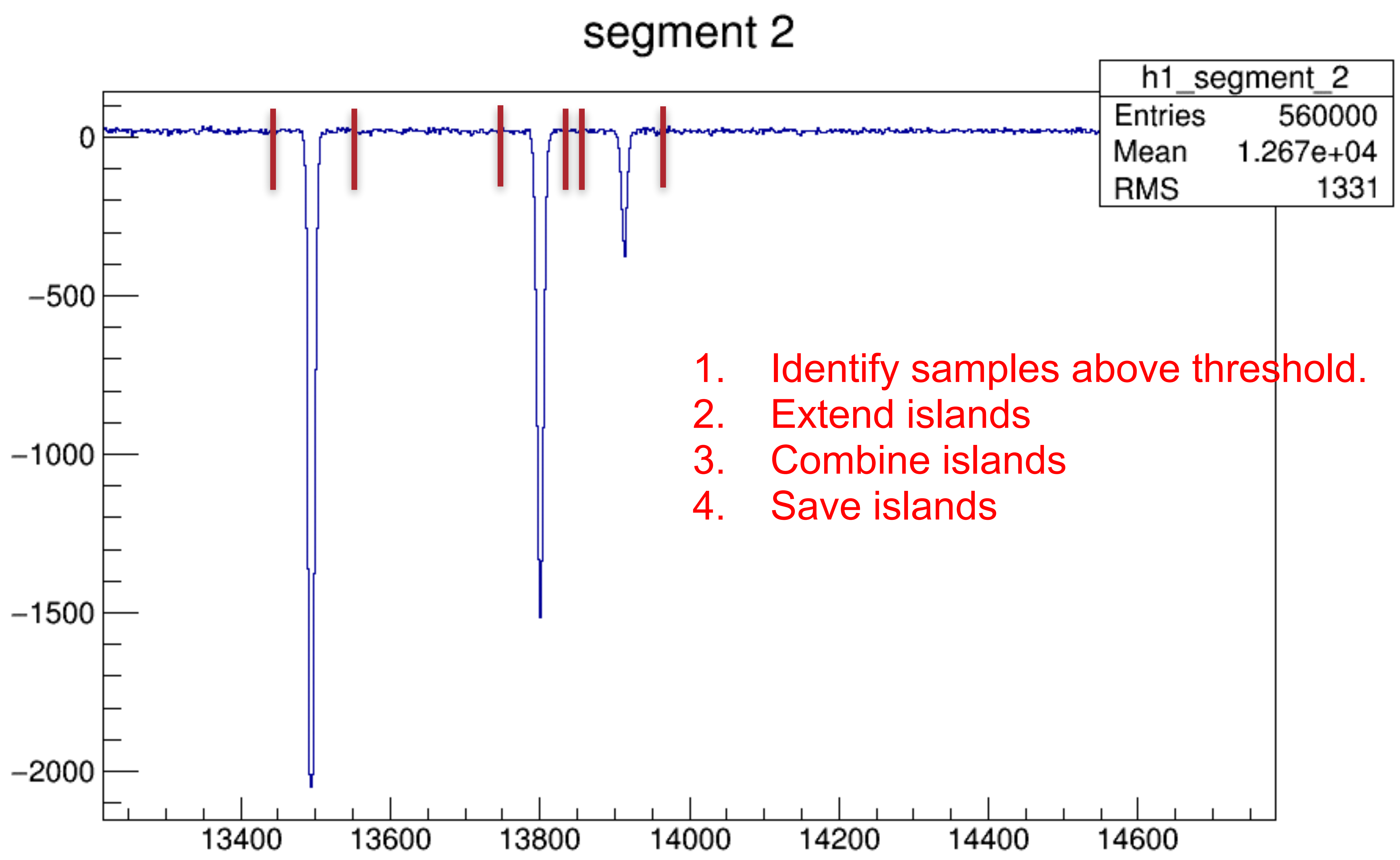
T-Method

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



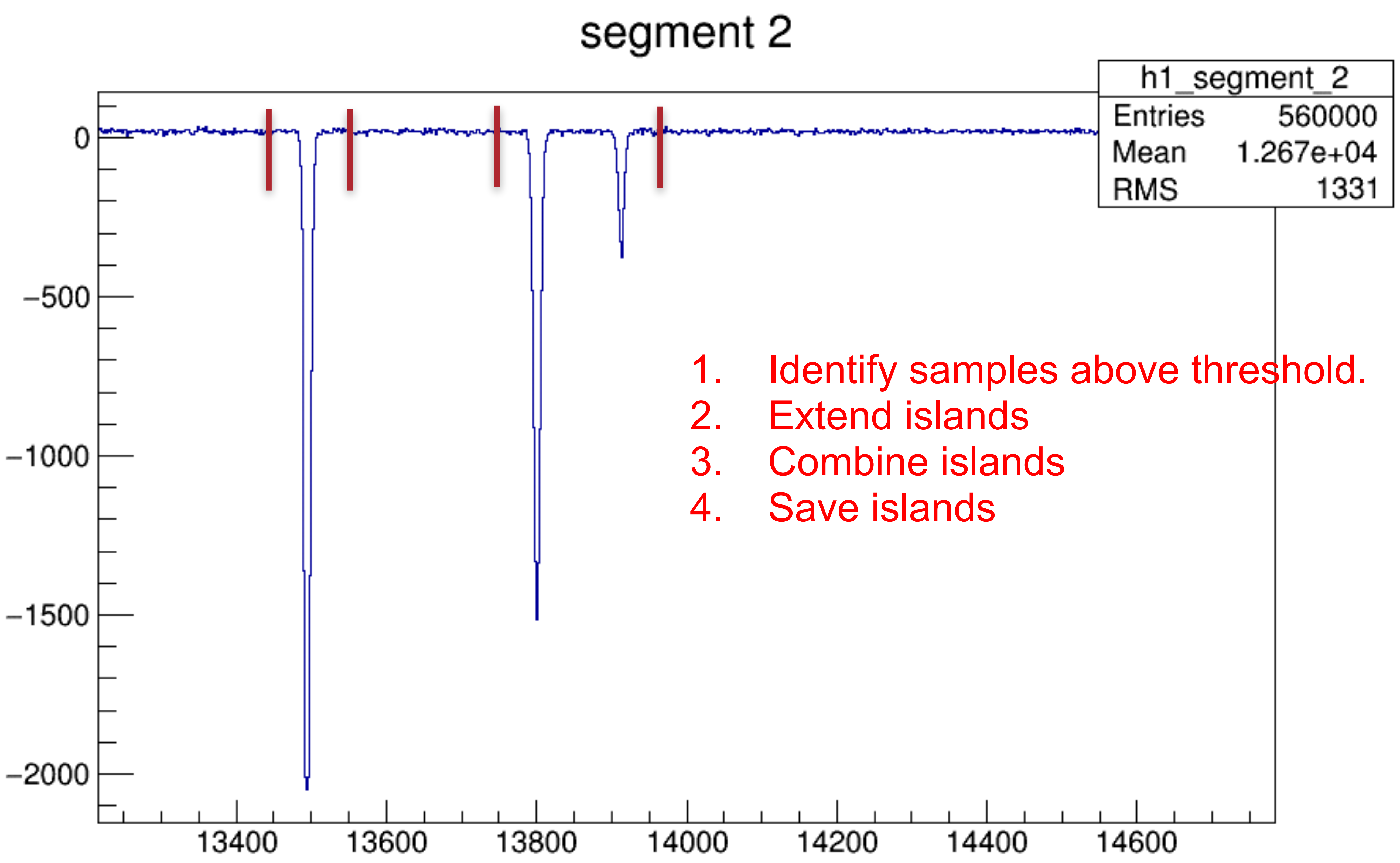
T-Method

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



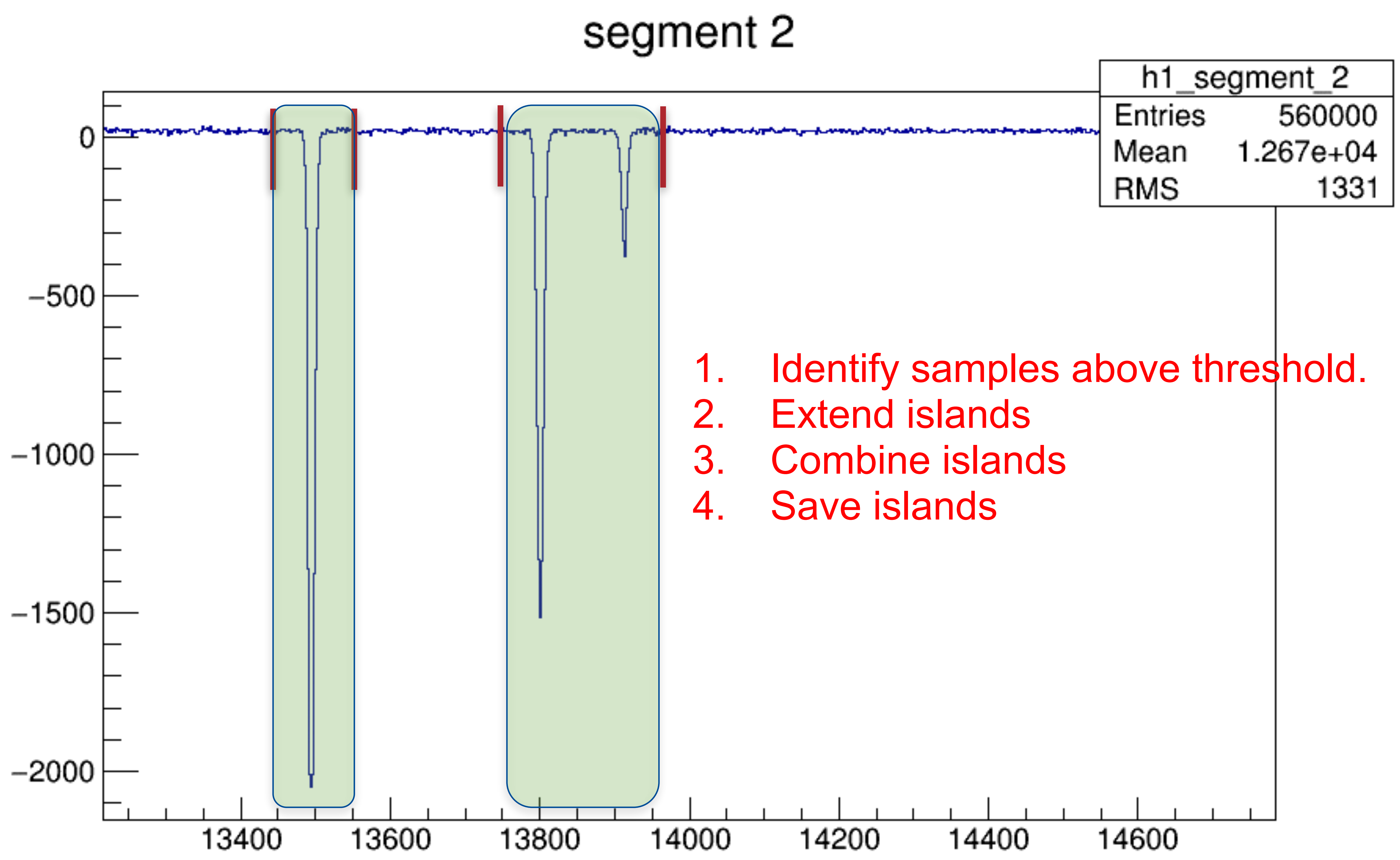
T-Method

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



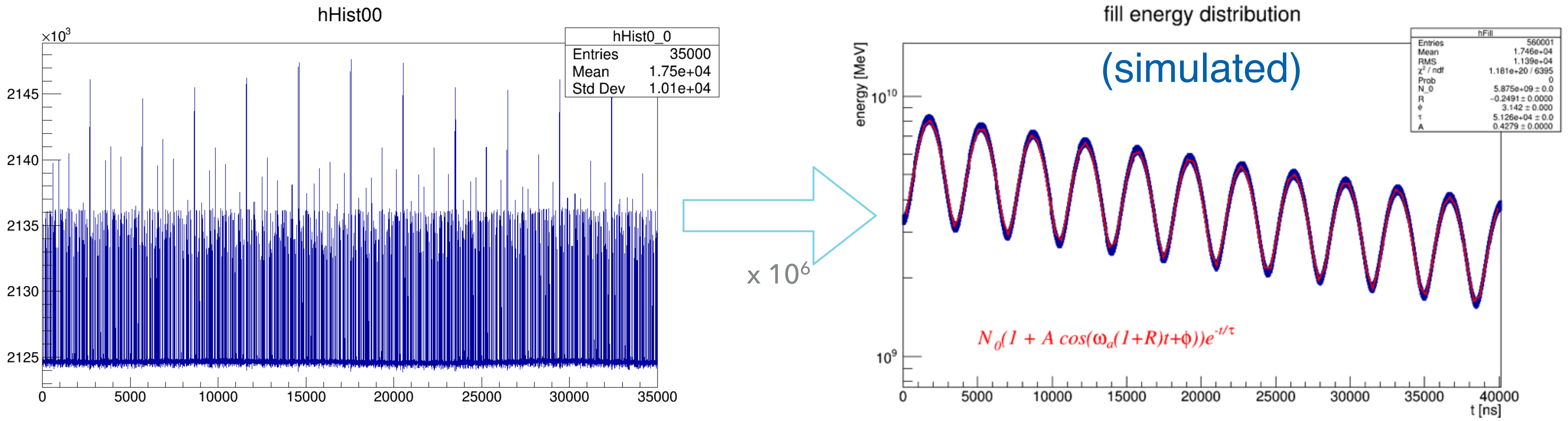
T-Method

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



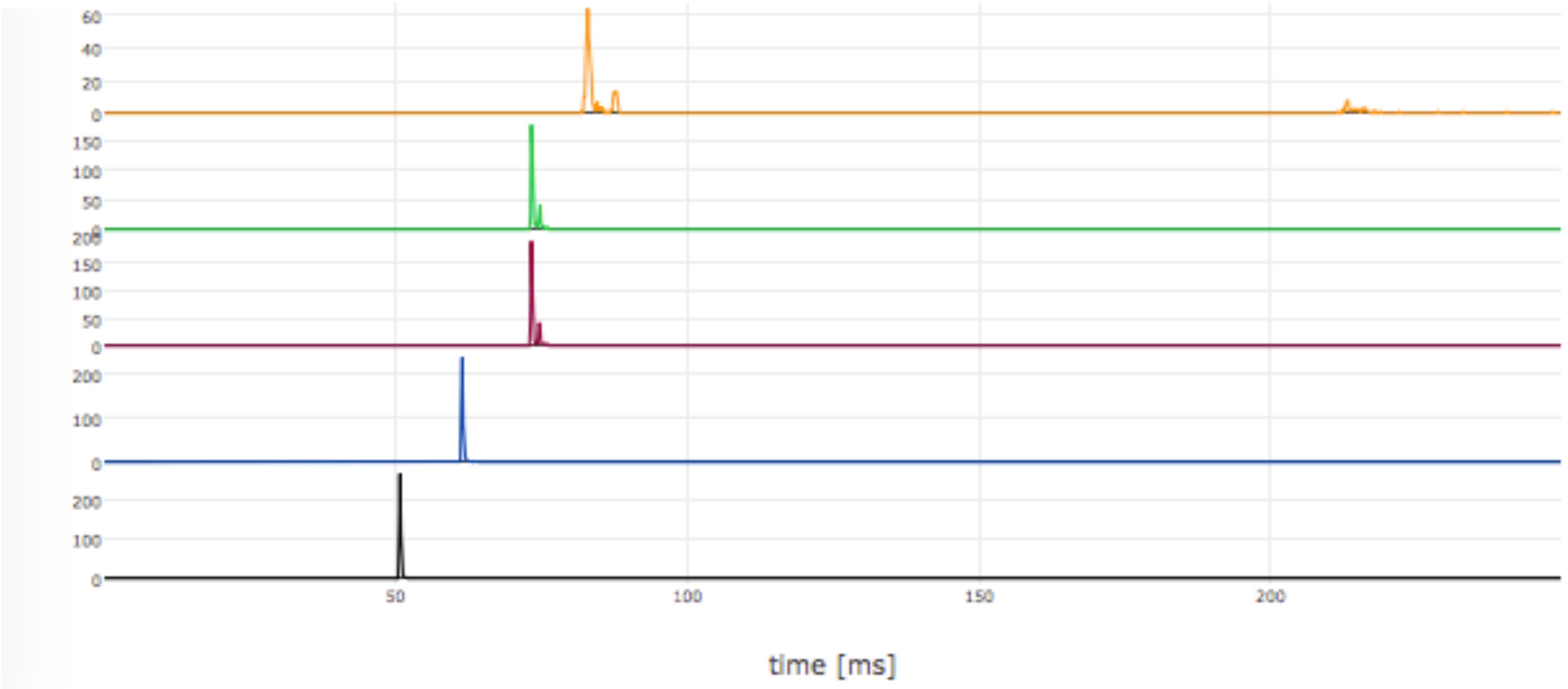
Q-method

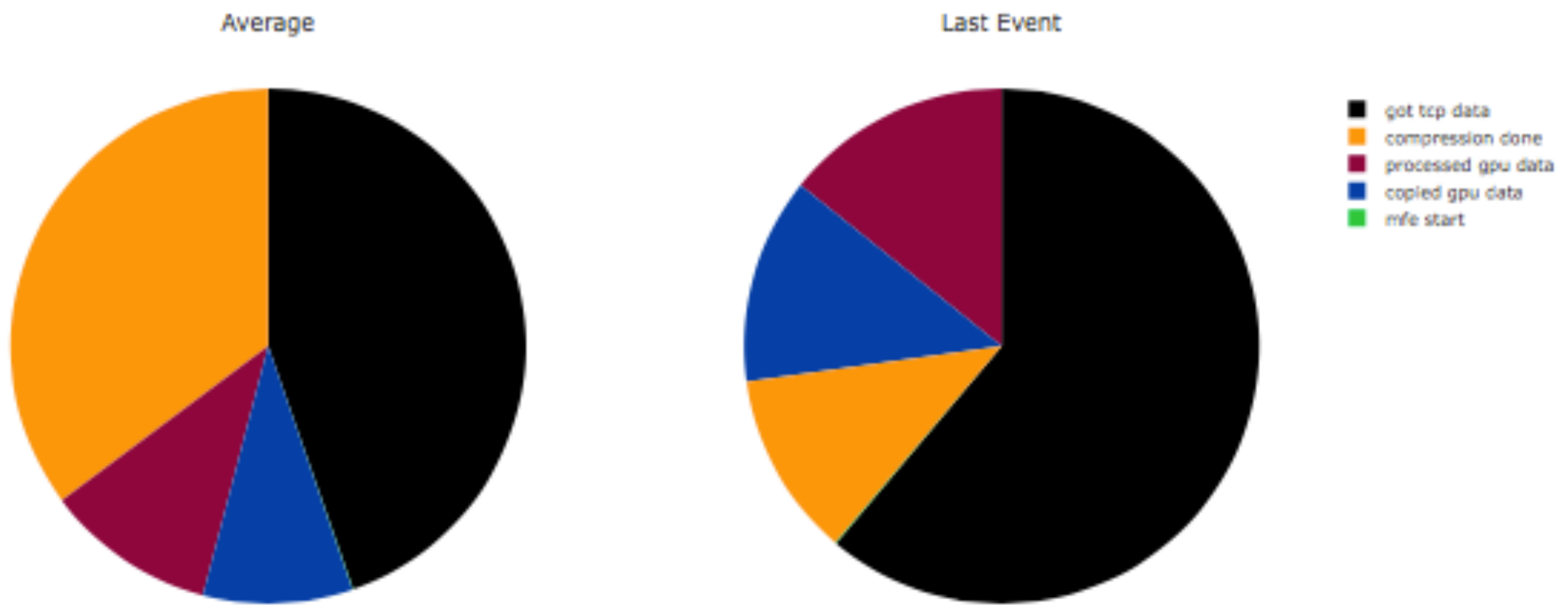
- Full waveforms are decimated in time and summed over many fills to create a histogram that is saved in the data file.
  - i.e. If we decimate in time by 10 and flush every 100 fills, we reduce the data rate by a factor of 1000, so from 20 GB/s to 20 MB/s.
- Use smaller bins at lower times and wider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



Processing time

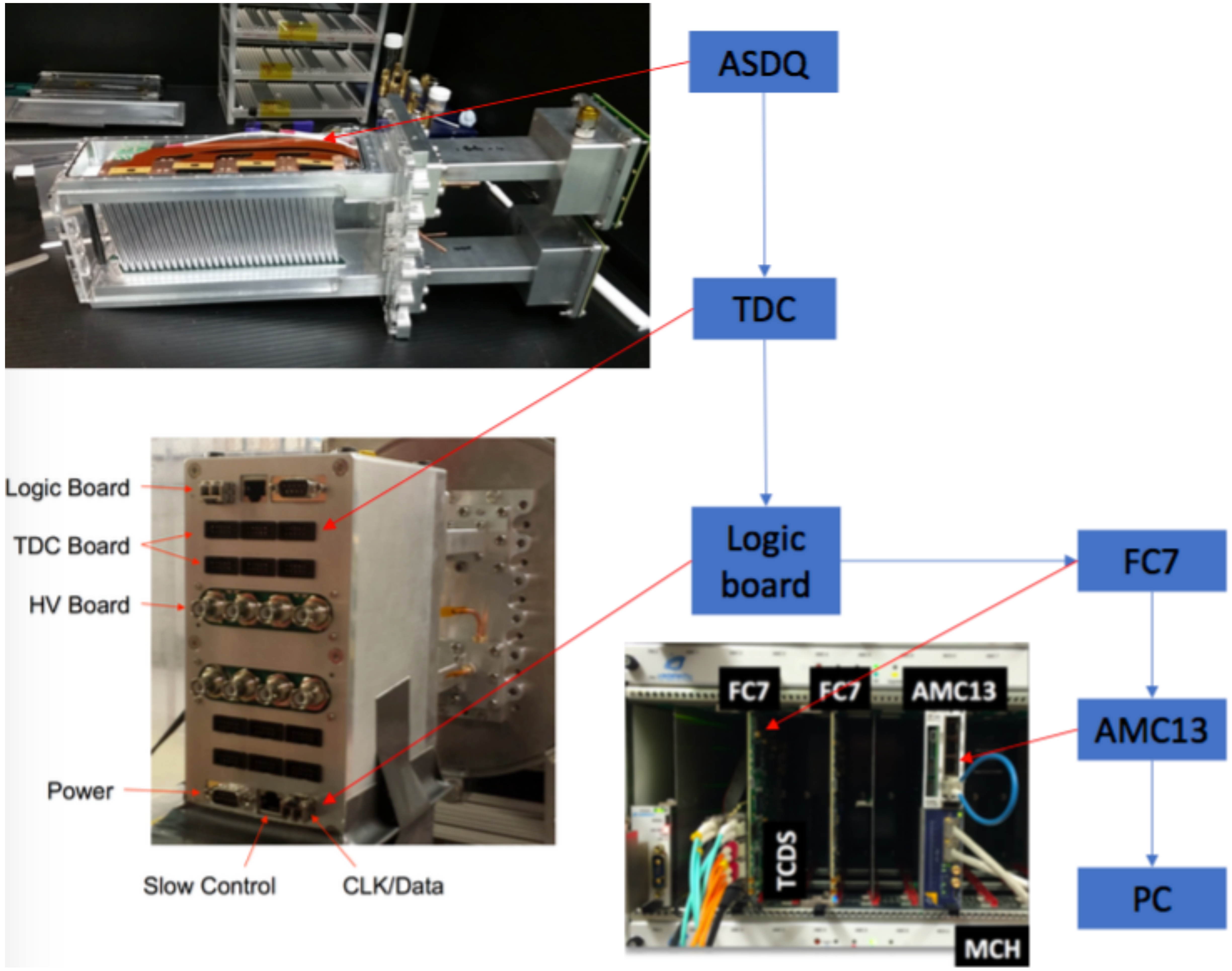
- Must process each event in 83 ms to keep up with average beam rate of 12 Hz.
- Most time is spent reading data from TCP socket and copying it to the GPU.
- Processing time in the GPU is very small.





Tracker Frontend

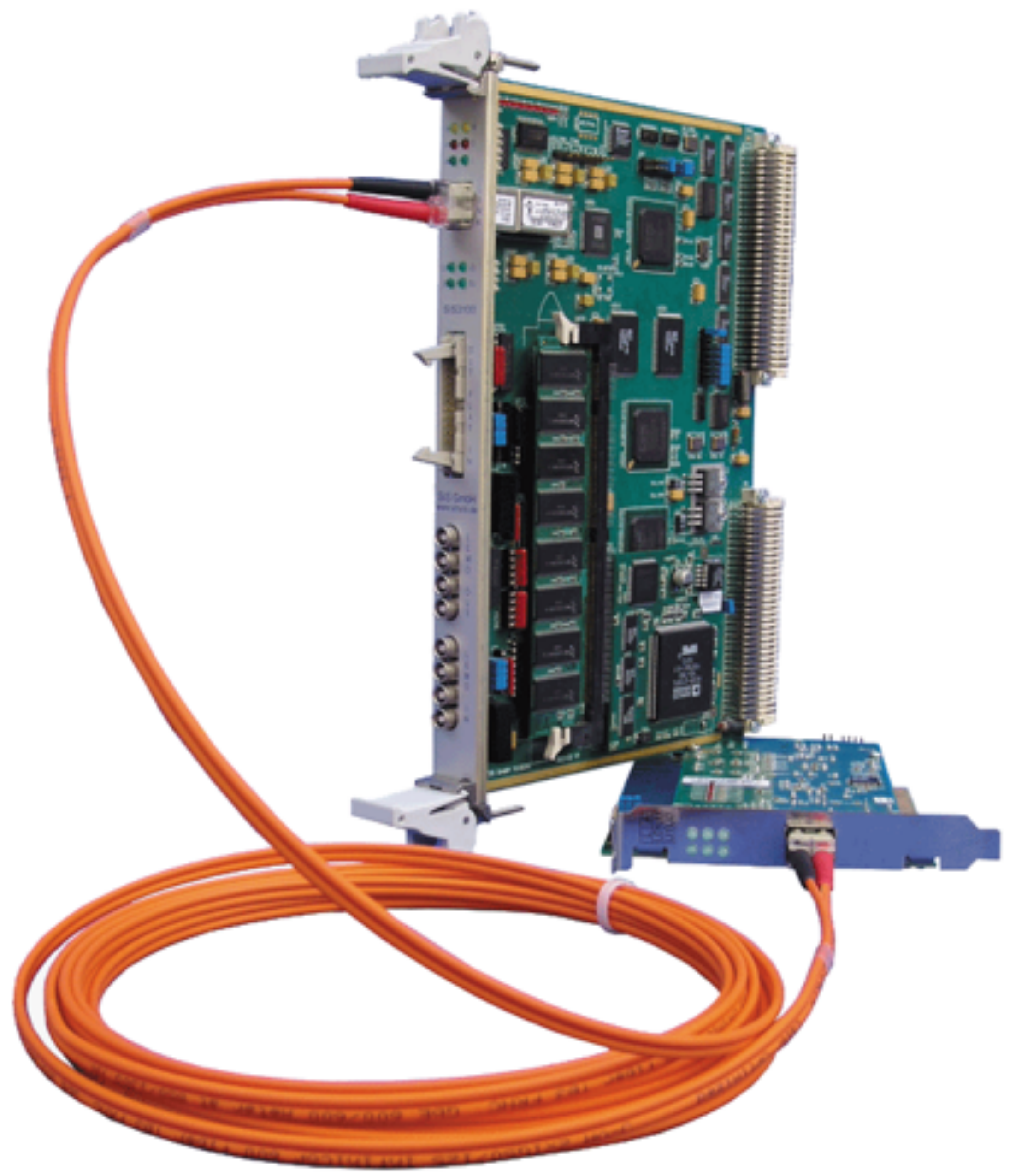
- Three tracker stations will be read via one uTCA crate.
- Reads data from AMC13.
- Instead of digitizers, data comes from multihit TDCs that are read via FC7 cards.

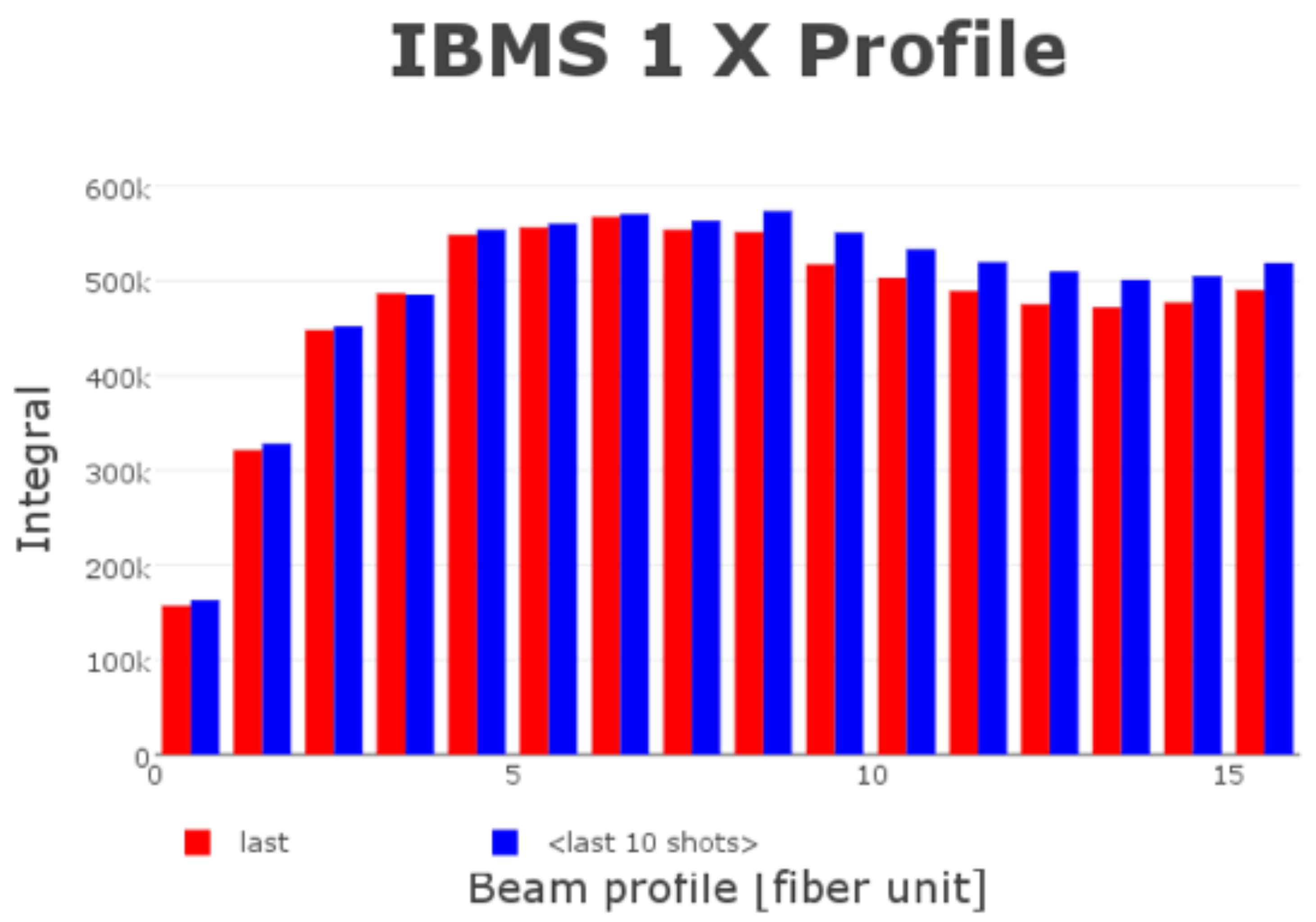


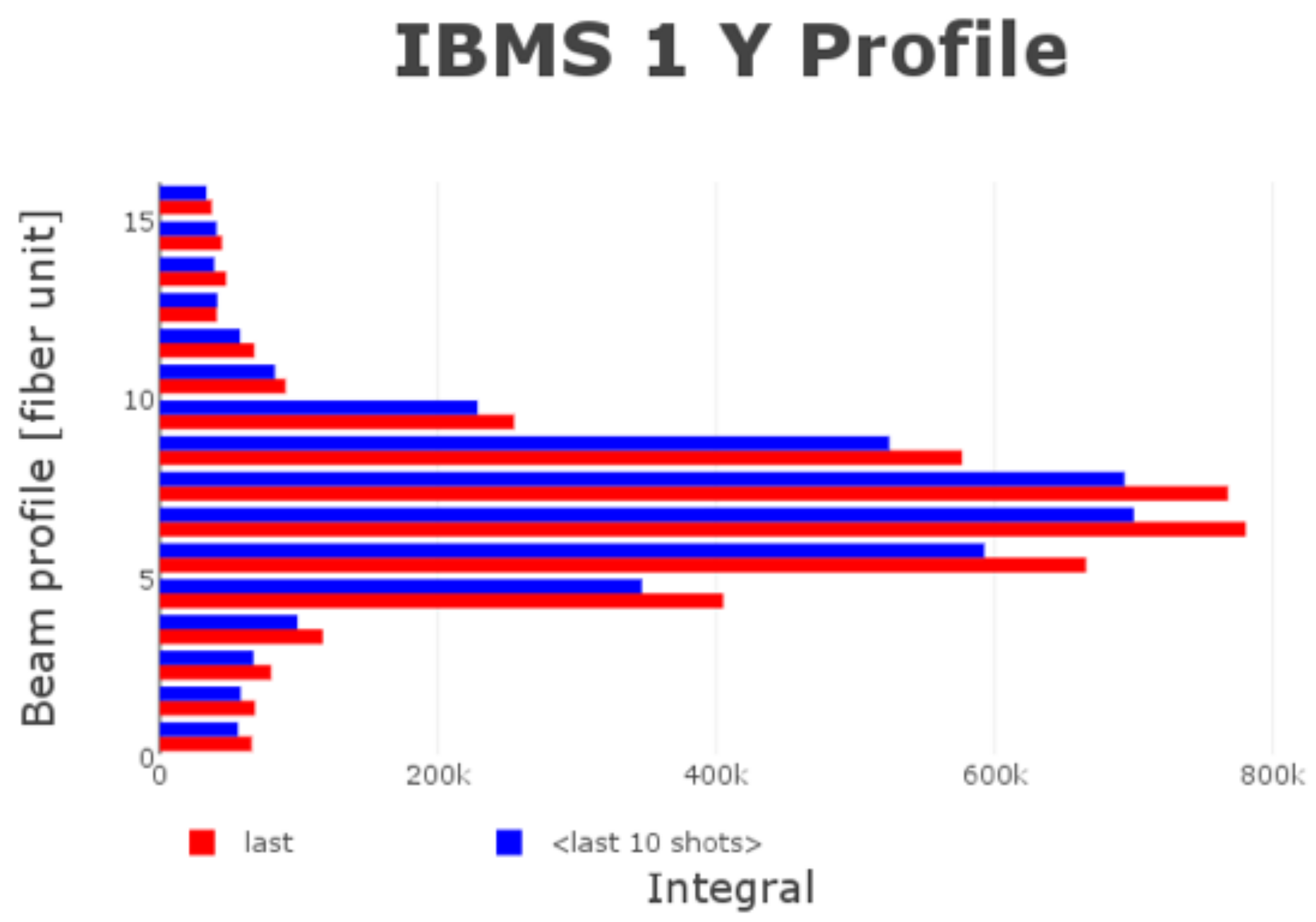
(Thanks R. Chislett for the diagram)

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.





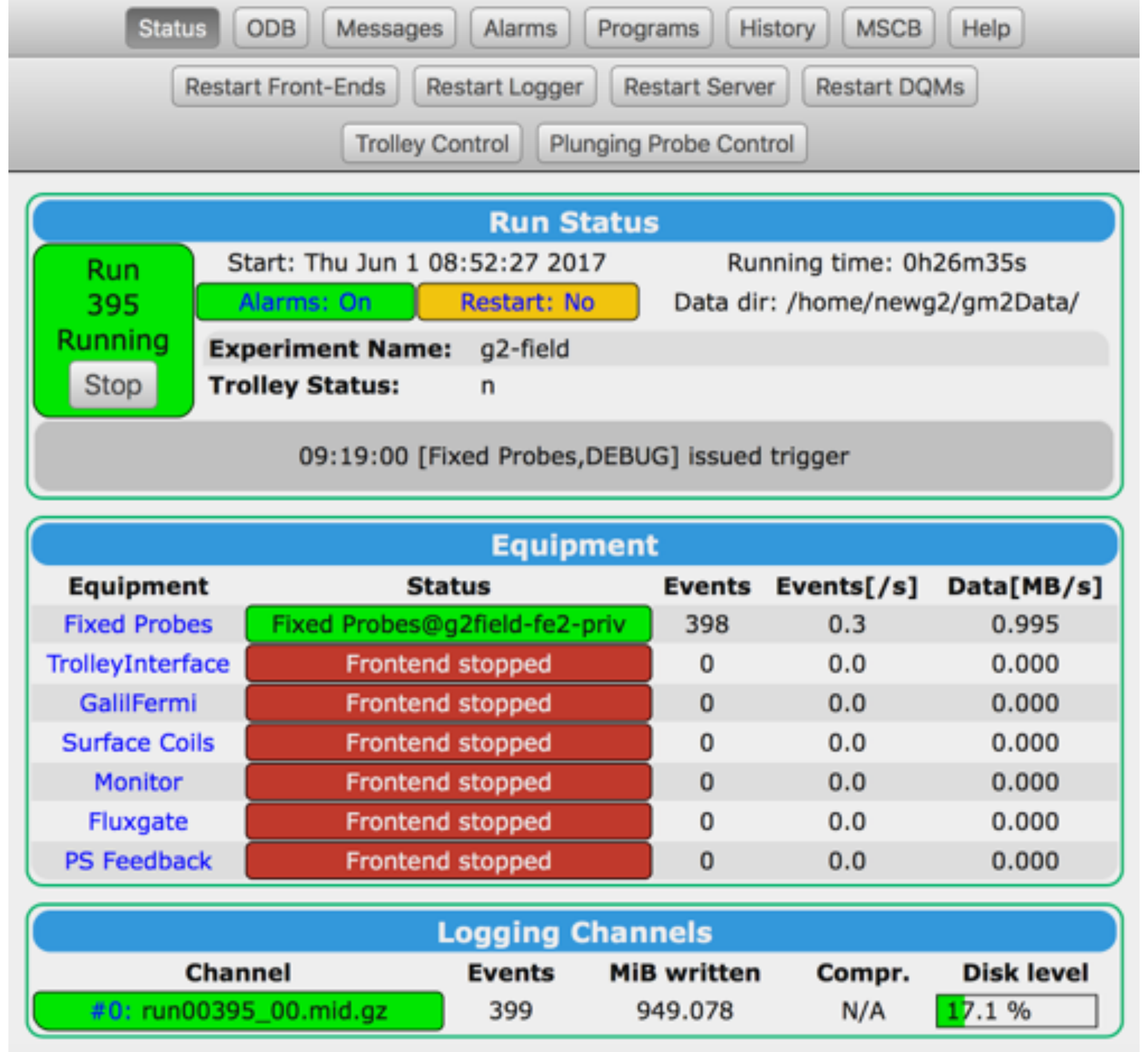


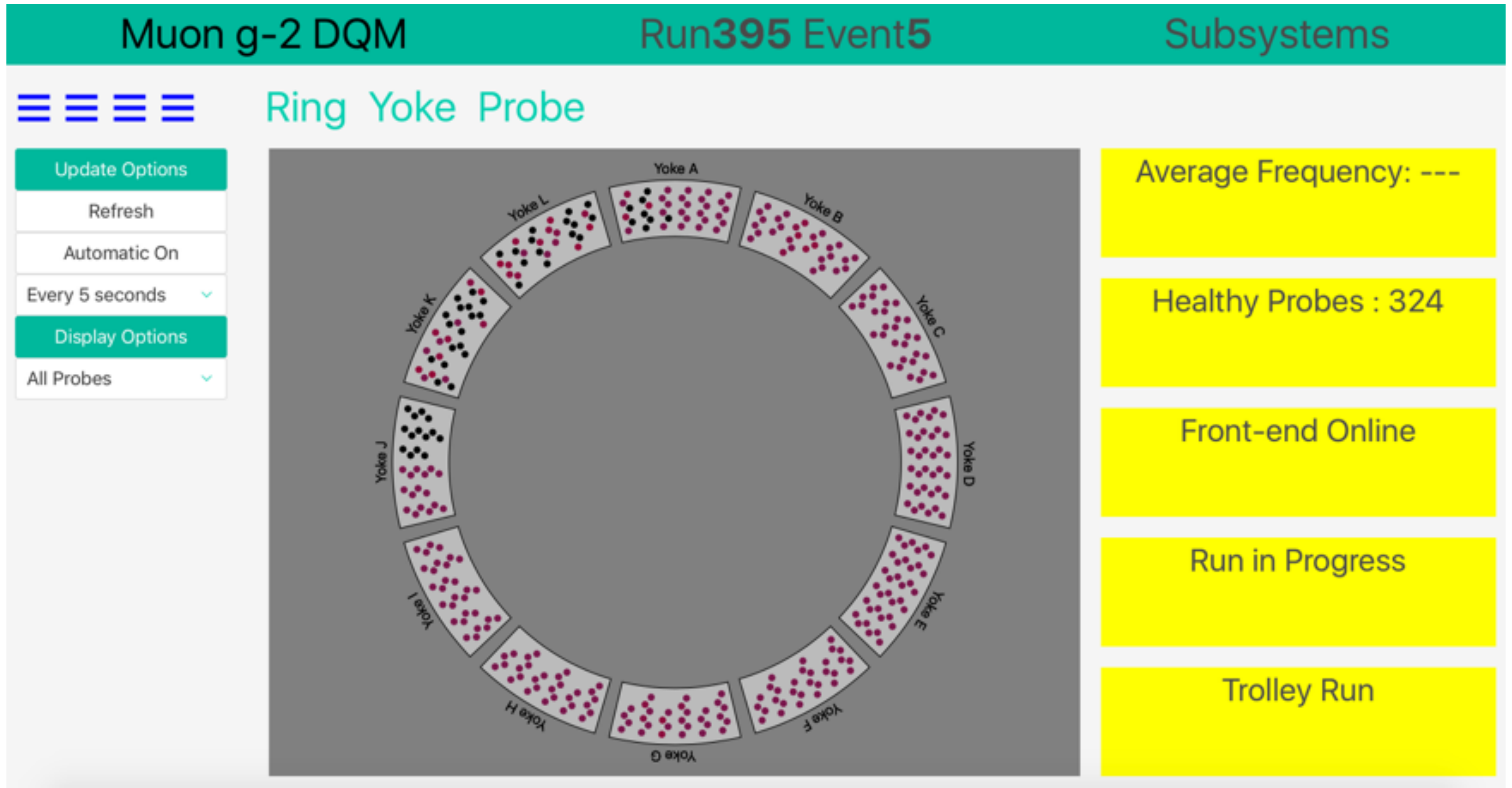
Slow Controls

- DAQ includes six SCS3000 mscb devices.
- 24 beaglebones reading slow control data from calorimeters.
- HV and LV frontends for tracker system.
- Slow frontend reading magnet properties from IFIX via an OPC client.
- Beamline frontend periodically reading output of beam components from database.
- Slow control data is stored in a Postgres database and displayed using a custom Django web display.

Field DAQ

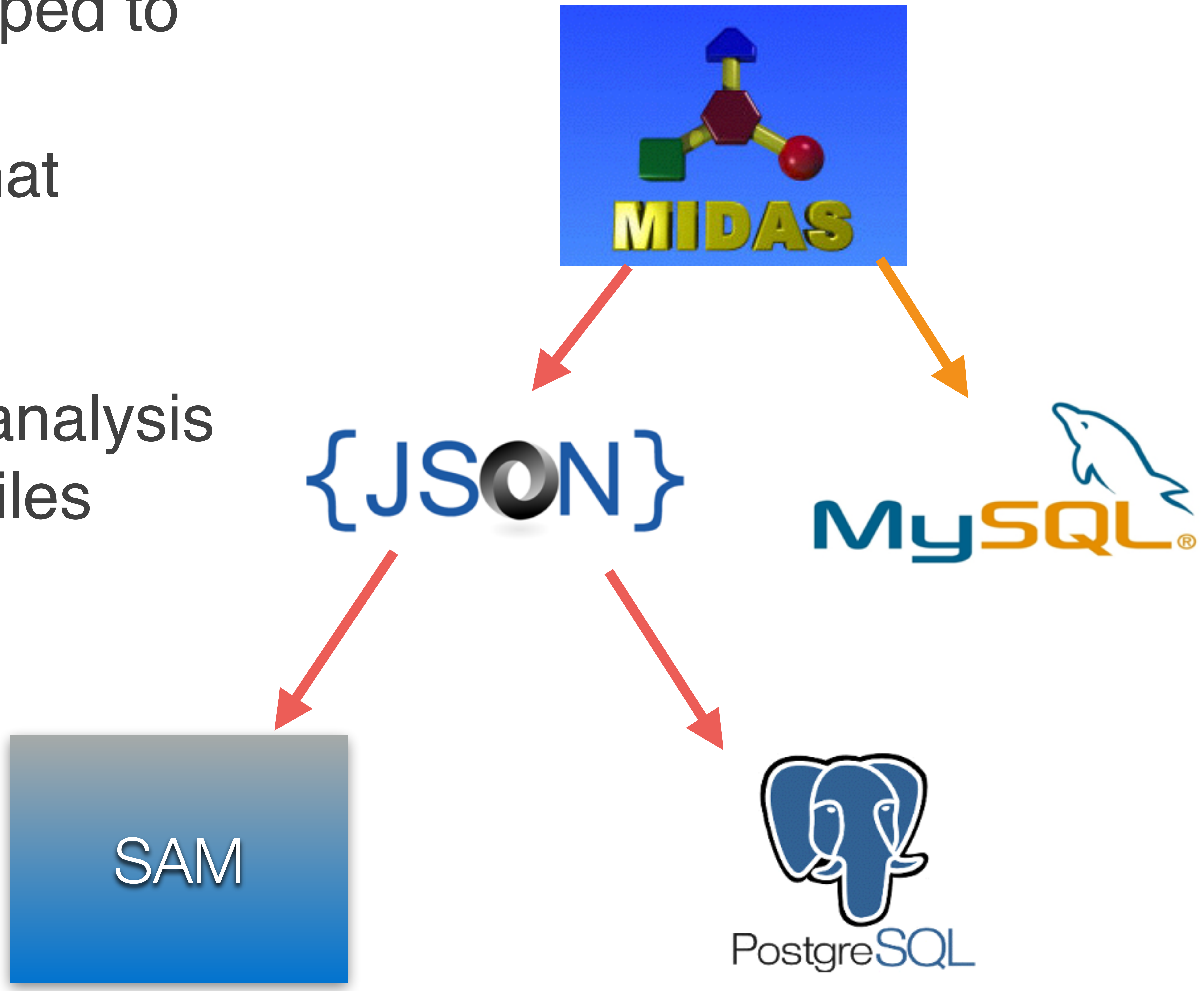
- Field DAQ runs in independent MIDAS experiment.
- Contains seven asynchronous frontends reading data from fixed magnetic field probes and from a trolley that periodically transverses the ring to perform precision measurements of magnetic field.
- Data is correlated with the fast DAQ offline using GPS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





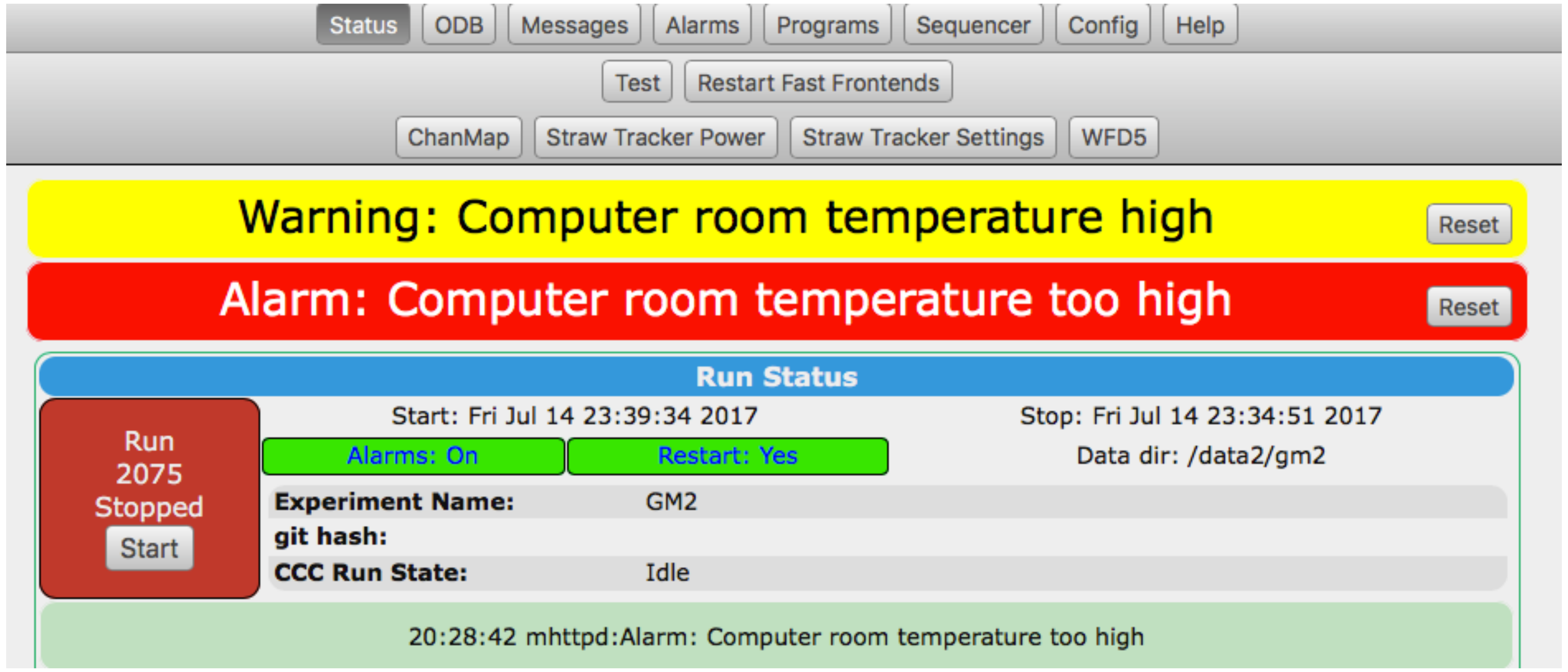
MIDAS ODB Archive

- 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 Alarms

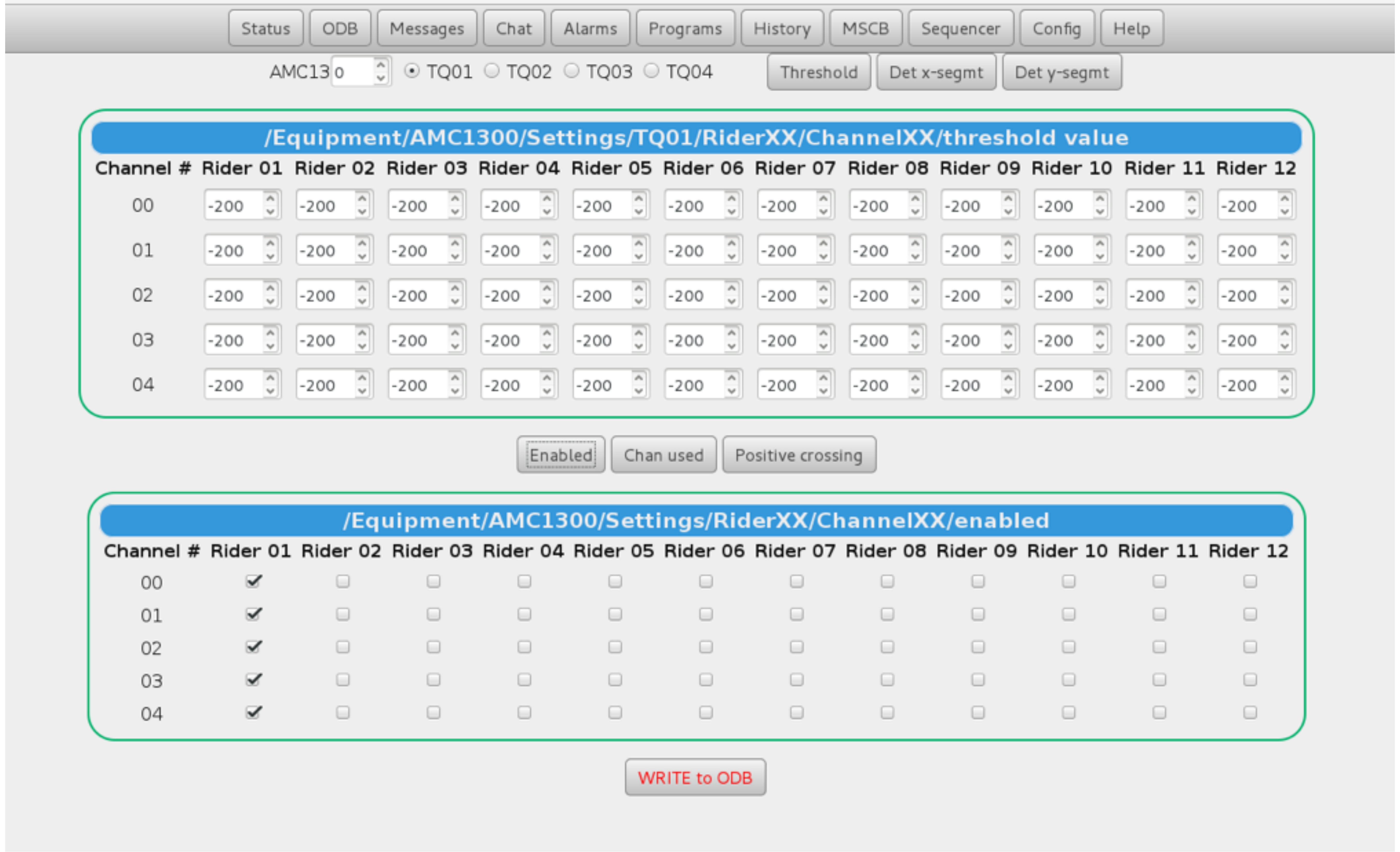
- The MIDAS alarm system was used as the primary alarm system.
- Alarms were set on temperatures and voltages from MSCB devices.
- Other slow frontends set alarms automatically when encountering an error.
- Periodic alarm reminded shifters to perform shift checks.
- Had problems at first with alarm audio, which was traced to a recent lack of mp3 support in scientific linux — this was later rectified with a recent update.



Fermilab

Custom Controls Page

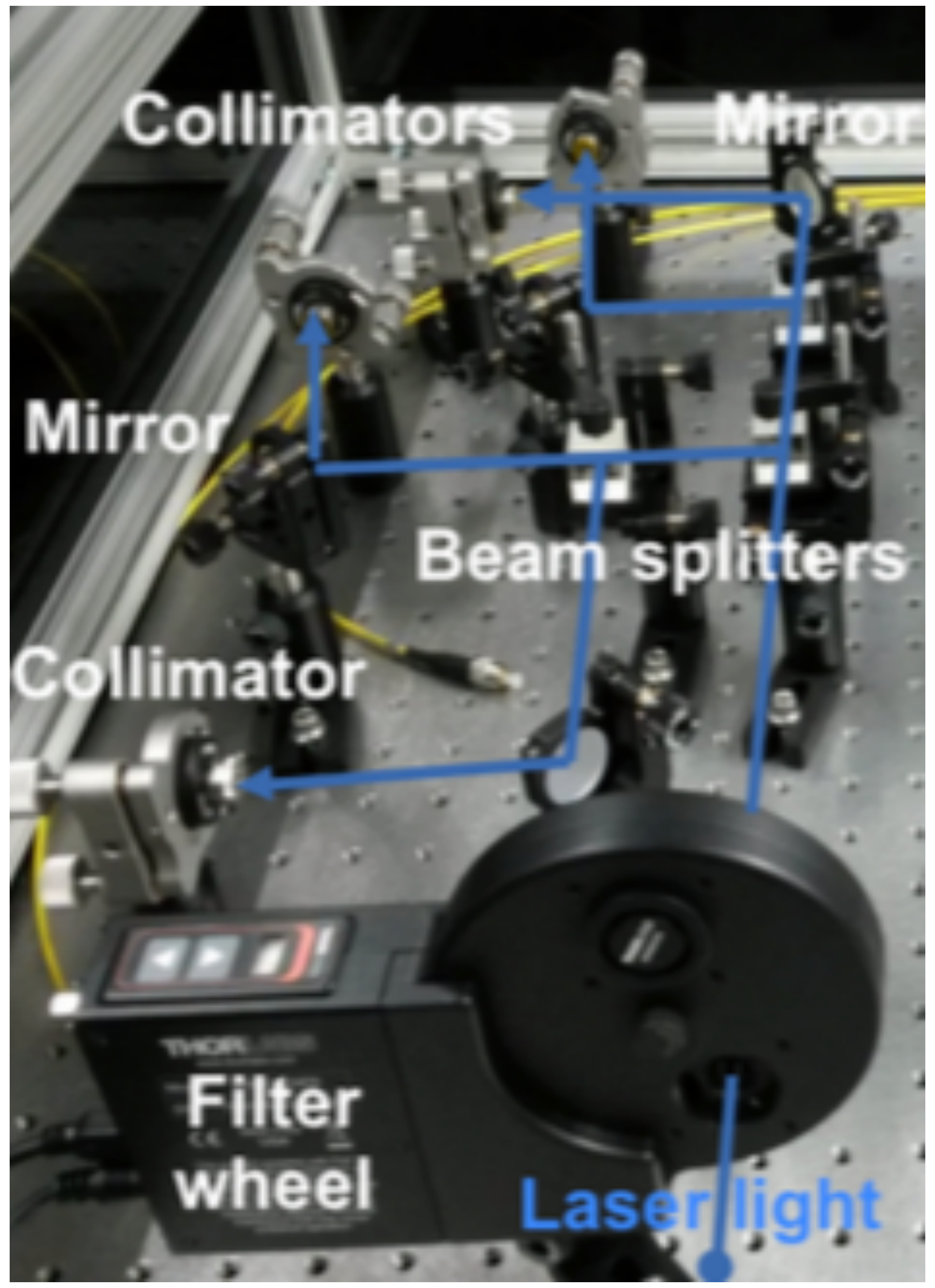
- With this number of frontends, configuring settings via the standard ODB tree is very cumbersome.
- A set of custom Javascript pages were written to manipulate ODB values en masse.

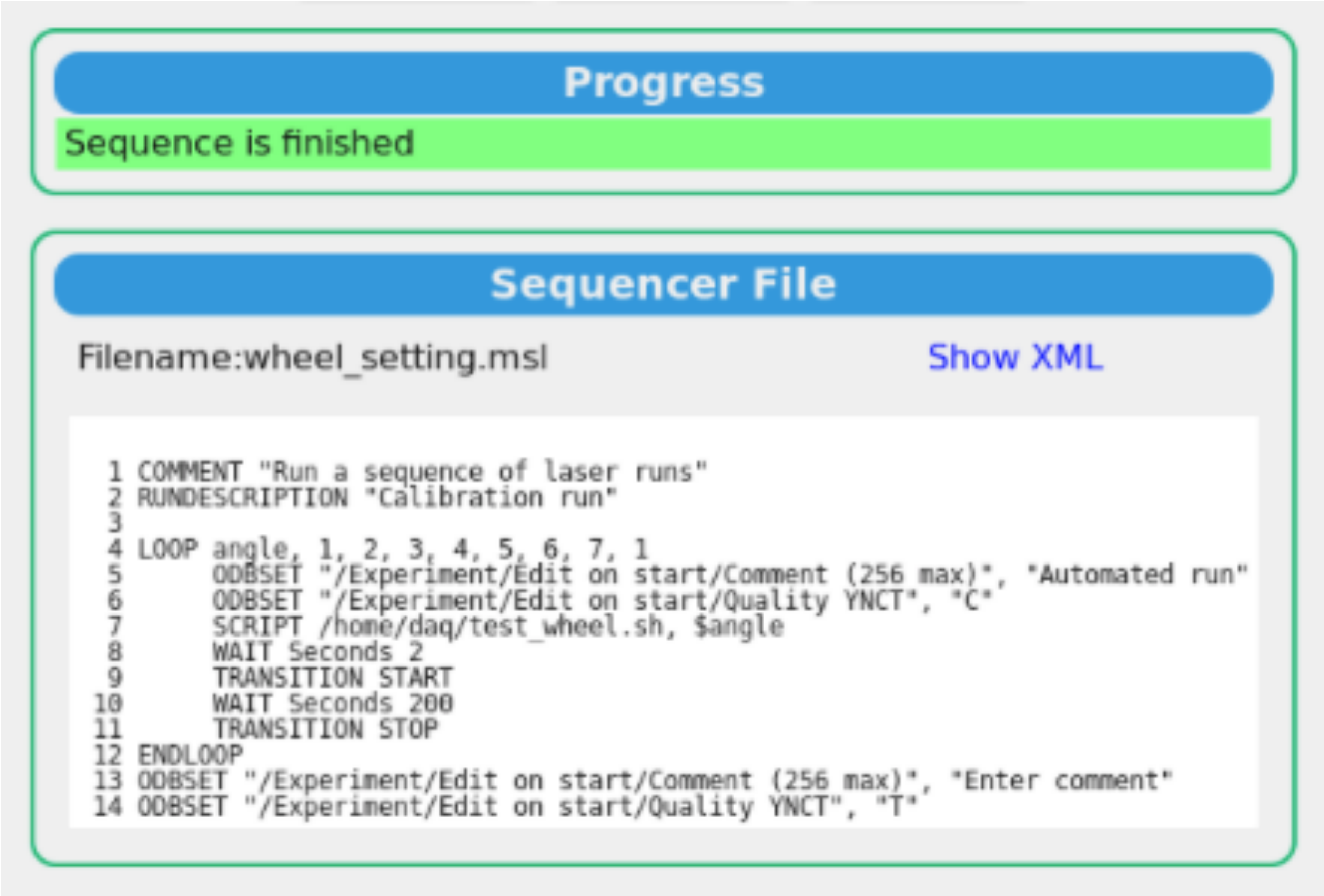


MIDAS Sequencer

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

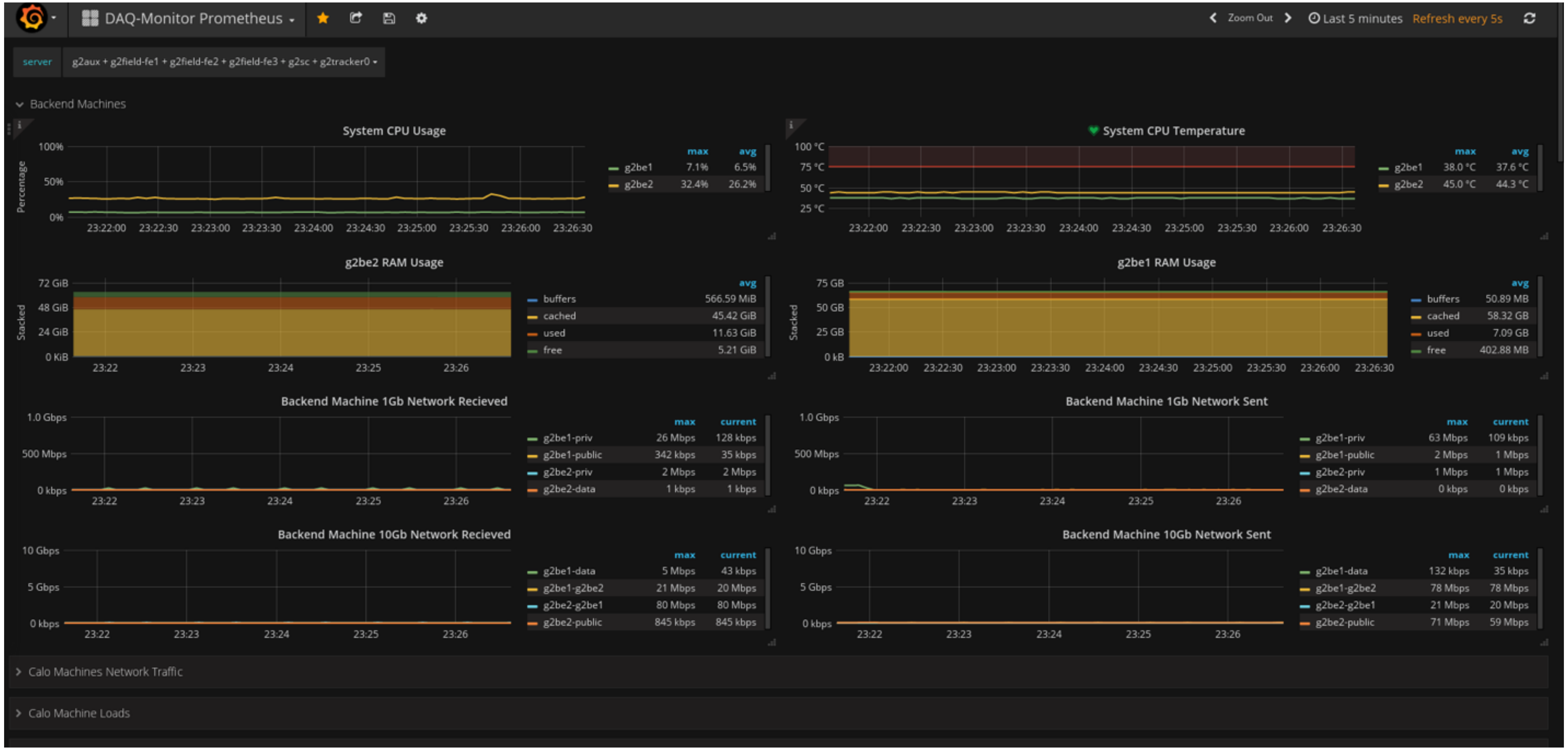
- A typical sequence would be:
  - Execute script to move wheel.
  - Update ODB values
  - Take data for 10 minutes
  - Repeat





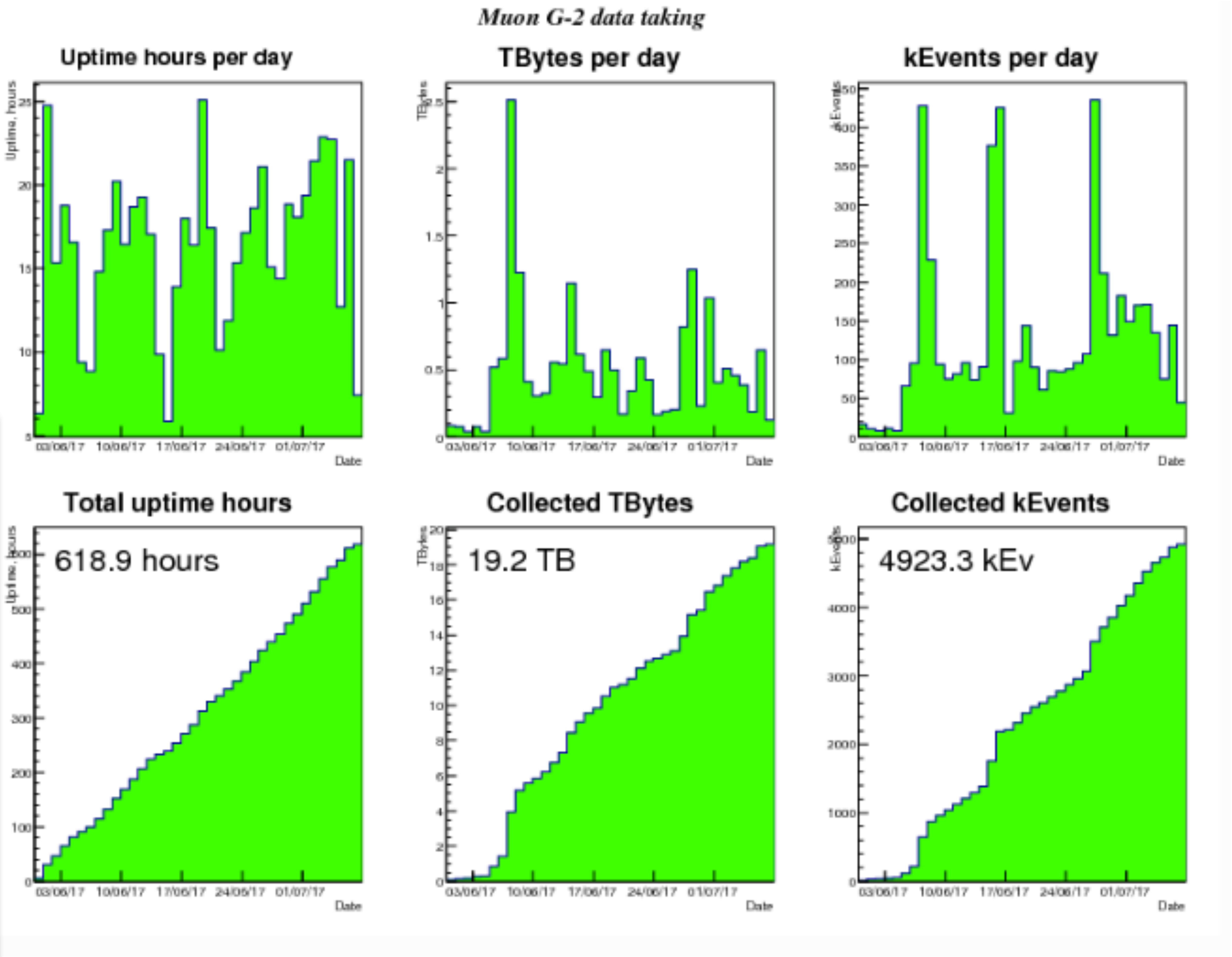
DAQ Health Monitor

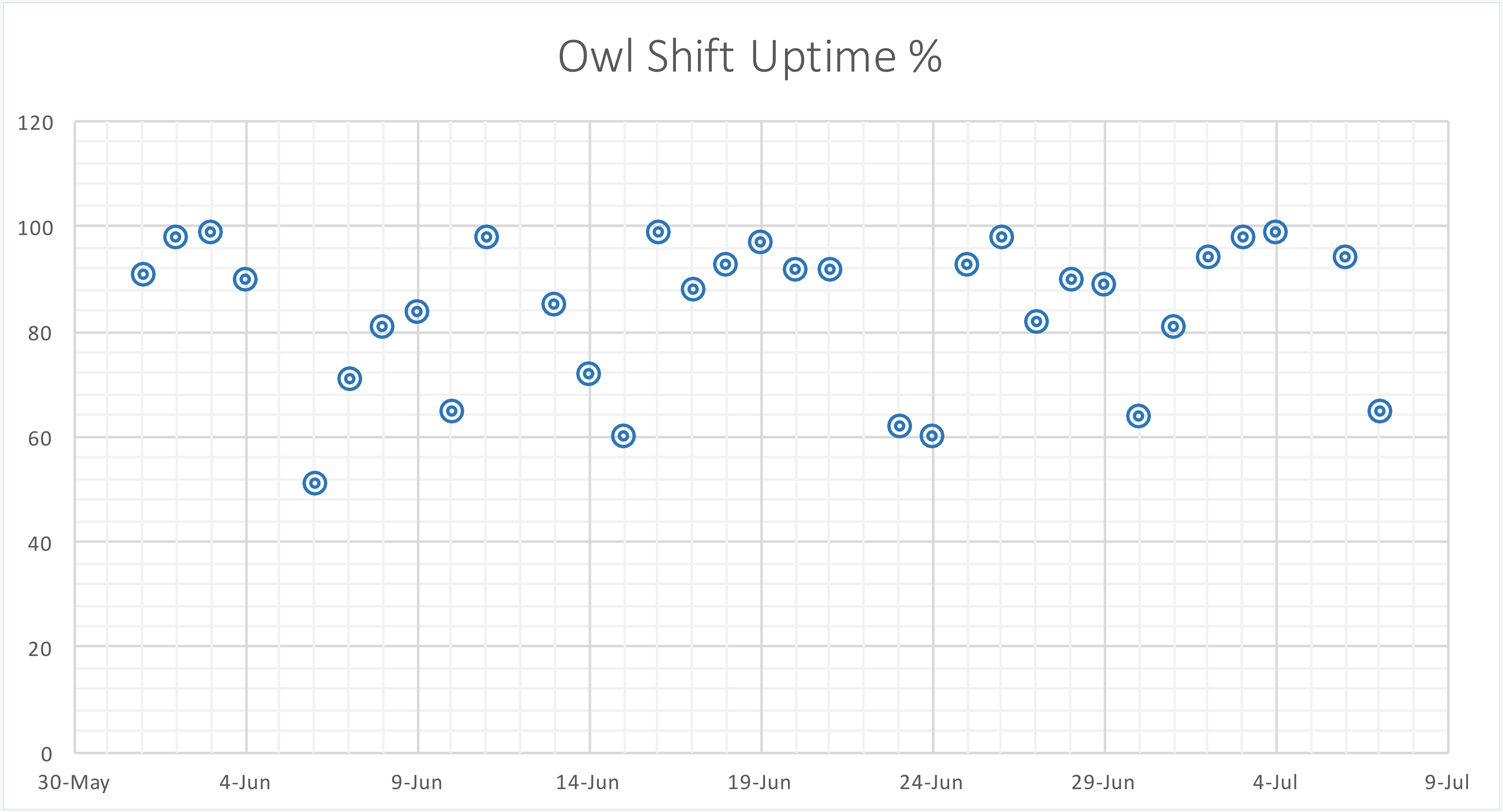
- Monitored health of DAQ systems using netdata for system monitoring, prometheus for short term data storage, and grafana to display data.



DAQ Performance During First 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.





MIDAS Issues experienced during run

- Had to increase maximum number of clients. The procedure was not obvious. We had to change:
  - MAX_CLIENTS in midas.h
  - MAX_RPC_CONNECTIONS in mserver.h
  - assert(size_of(BUFFER_HEADER)) in odb.c
  - DATABASE_HEADER calculation in odb.c
- ODB full errors and ODB Corruption.
- Serial Begin-of-run for frontends.
- MIDAS Web interface was sometimes very slow.
- Best way to protect ODB during running?

Summary

- MIDAS is working very well as the DAQ software for Muon g-2 at Fermilab.
- 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.
- It takes 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.
- Thank you to the MIDAS team for your support!

Backup

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.