



Midas DAQ for Muon $g-2$ at Fermilab

Wes Gohn

TRIUMF Midas Seminar

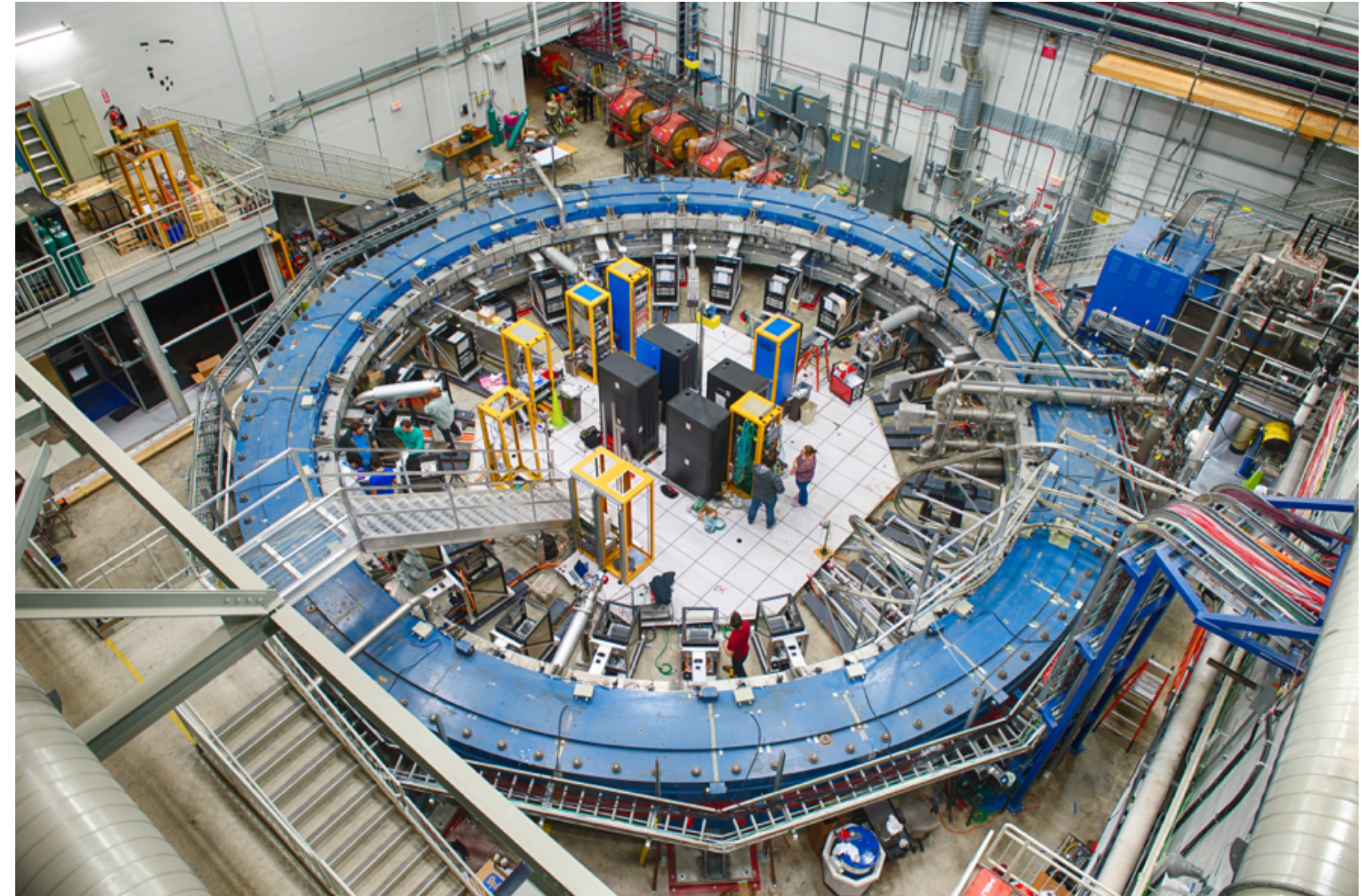
26 July 2017



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.



DAQ Input Sources



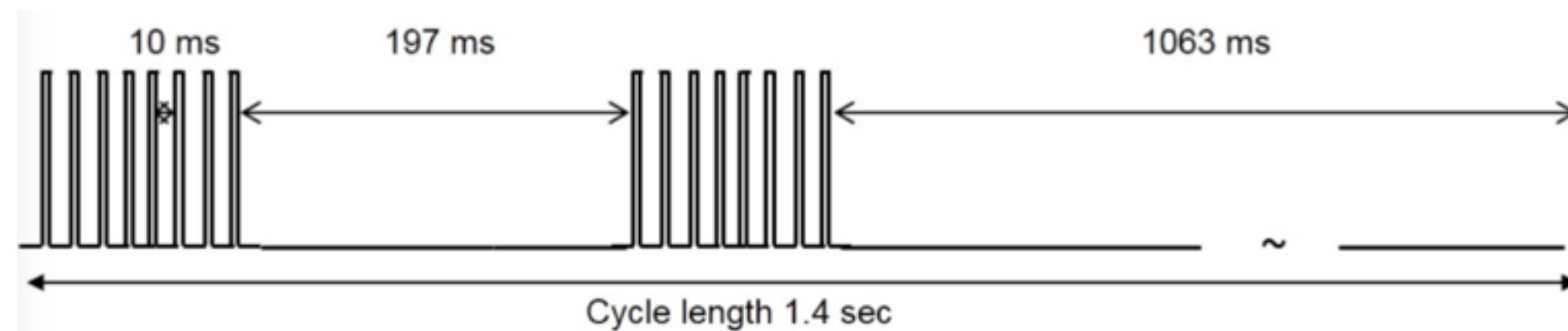
- 24 Calorimeters
 - 1 uTCA crate for each calorimeter
 - 54 channels * 24 calos = 1296 channels of digitized data.
 - Data processed by 12 Cornell WFD5s and sent from AMC13.
- 4 Fiber Harps
 - 7 channels * 4 harps = 28 channels
 - Data processed by Cornell WFD5s
- Quads and Kickers
 - Write 4 quad channels 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.



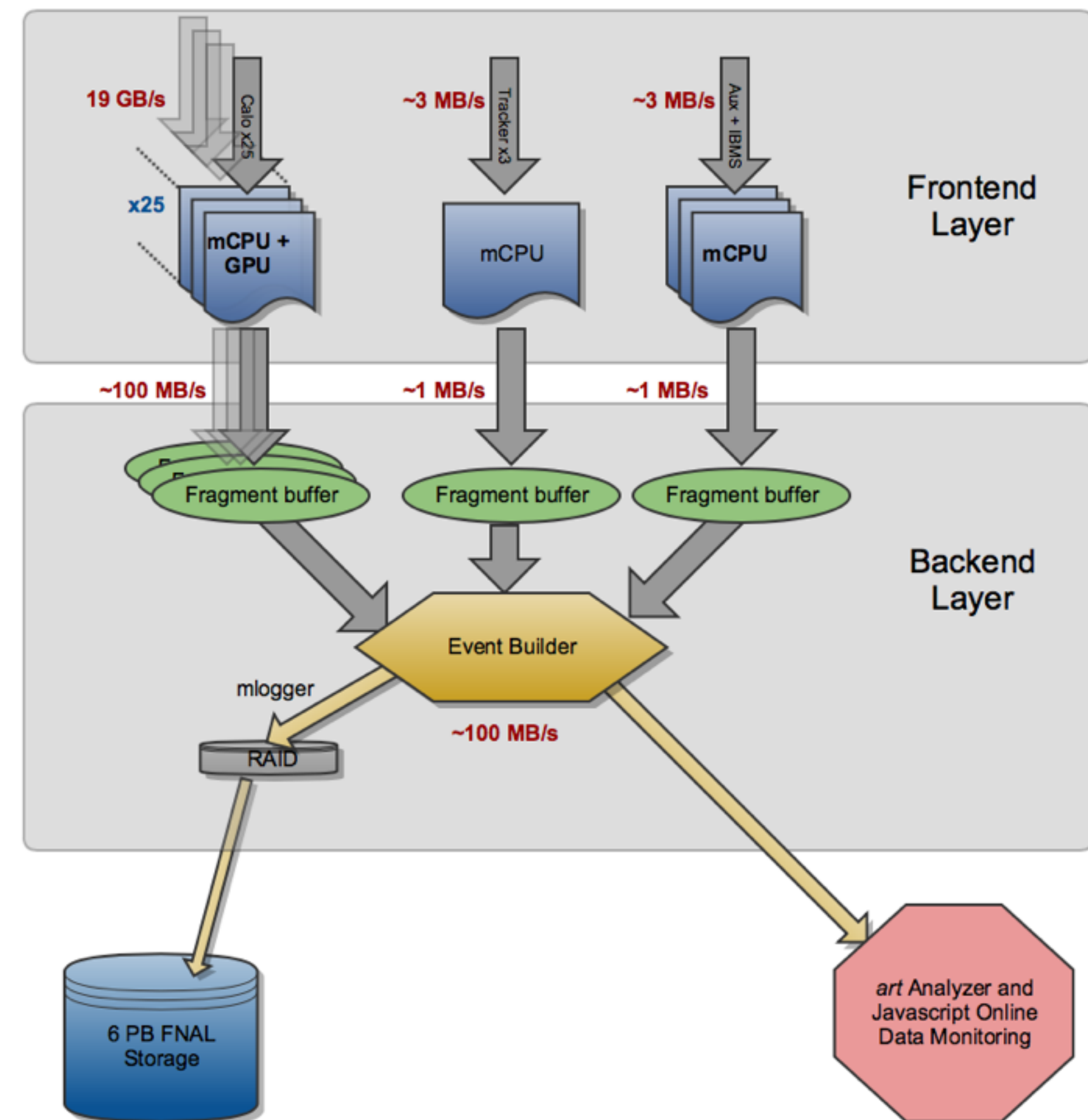
- Time-averaged rate of raw ADC samples is 20 GB/s, which must be reduced by a factor of 100.
- Data is processed in GPUs to accomplish this task.
- Total data on tape after 2 years of running will be 10 PB.

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

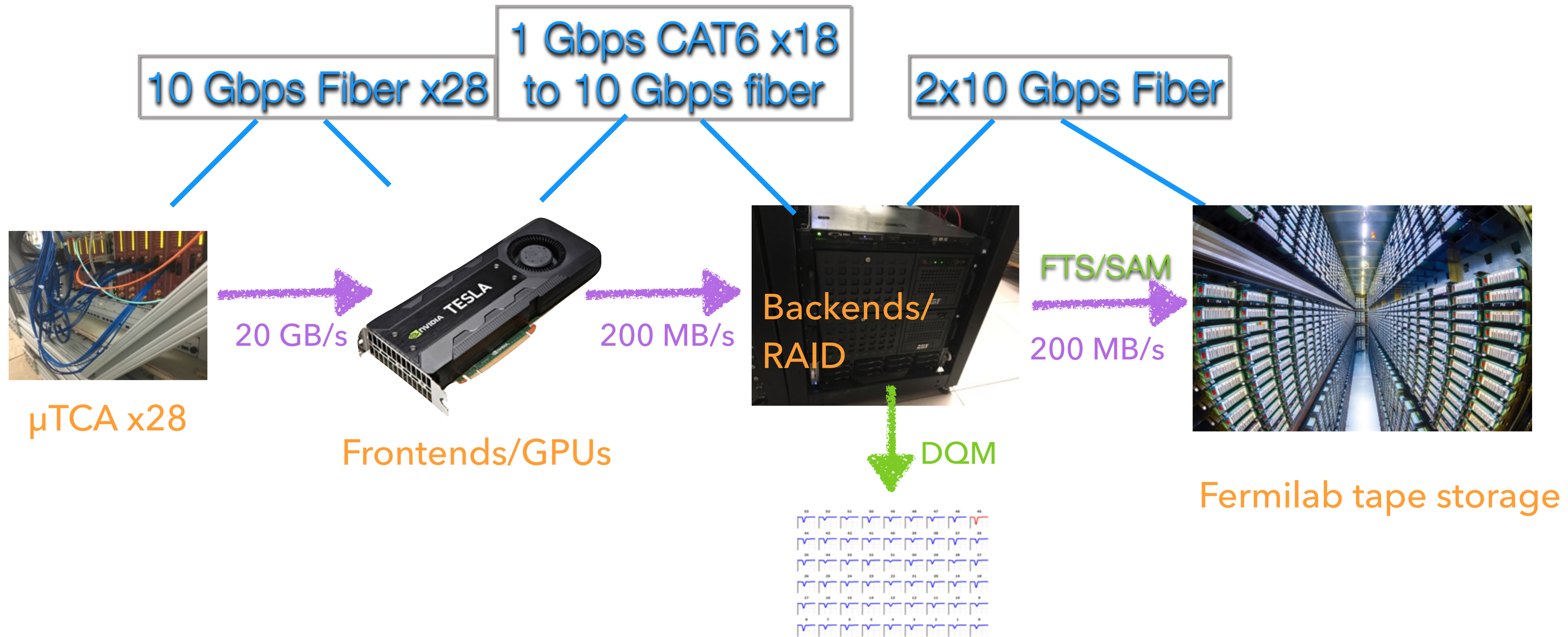
DAQ Design



- 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 (see talk by A. Fienberg).
- Separate MIDAS experiment running for magnetic field DAQ.

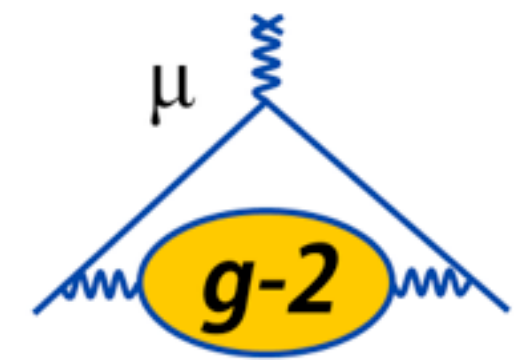
Run Status
 Start: Wed Jun 28 05:38:50 2017 Running time: 0h29m03s
 Alarms: On Restart: Yes Data dir: /data2/gm2
 Experiment Name: GM2
 git hash:
 CCC Run State: Run In Progress
 06:07:20 [Logger,INFO] channel /data2/gm2/gm2_run01618_47.mid writer chain: CRC32C | CRC32C | >

Equipment +	Status	Events	Events[/s]	Data[MB/s]
MasterGM2	MasterGM2@g2be1.fnal.gov	1222	0.7	0.000
EB	Ebuilder@g2be1.fnal.gov	1221	0.0	0.000
AMC1300	AMC1300@g2aux-priv	1223	1.0	0.001
AMC1301	AMC1301@g2calo0102-data	1223	1.0	2.016
AMC1302	AMC1302@g2calo0102-data	1222	0.7	1.949
AMC1303	AMC1303@g2calo0304-data	1221	0.7	0.096
AMC1304	AMC1304@g2calo0304-data	1221	0.7	0.095
AMC1305	AMC1305@g2calo0506-data	1222	0.9	1.780
AMC1306	AMC1306@g2calo0506-data	1223	1.0	1.986
AMC1307	AMC1307@g2calo-spare-priv	1222	0.7	1.939
AMC1308	AMC1308@g2calo-spare-priv	1221	0.7	0.093
AMC1309	AMC1309@g2calo0910-data	1221	0.7	0.094
AMC1310	AMC1310@g2calo0910-data	1222	0.7	1.968
AMC1311	AMC1311@g2calo1112-data	1222	0.7	1.940
AMC1312	AMC1312@g2calo1112-data	1223	1.0	1.944
AMC1313	AMC1313@g2calo1314-data	1223	1.0	1.972
AMC1314	AMC1314@g2calo1314-data	1222	0.7	1.963
AMC1315	AMC1315@g2calo1516-data	1221	0.7	0.095
AMC1316	AMC1316@g2calo1516-data	1223	0.7	1.911
AMC1317	AMC1317@g2calo1718-data	1222	0.7	1.954
AMC1318	AMC1318@g2calo1718-data	1223	1.0	1.928
AMC1319	AMC1319@g2calo1920-data	1222	0.7	1.892
AMC1320	AMC1320@g2calo1920-data	1221	0.7	0.092
AMC1321	AMC1321@g2calo2122-data	1221	0.7	0.093
AMC1322	AMC1322@g2calo2122-data	1221	0.7	0.125
AMC1323	AMC1323@g2calo2324-data	1222	0.7	1.858
AMC1324	AMC1324@g2calo2324-data	1223	1.0	1.977
AMC1325	AMC1325@g2laserdaq-data	1221	0.7	0.069
AMC1326	AMC1326@g2aux-priv	1221	0.7	5.735
StrawTrackerLVandSC03	StrawTrackerLVandSC03@g2tracker1.fnal.gov	0	0.0	0.000
StrawTrackerDAQ	StrawTrackerDAQ@g2tracker0.fnal.gov	1221	0.7	0.006
StrawTrackerHV03	StrawTrackerHV03@g2tracker1.fnal.gov	0	0.0	0.000
IBMS Detector	IBMS Detector@g2ibms-priv	1223	0.7	0.121
CaloSC01	CaloSC01@g2sc-priv	0	0.0	0.000
CaloSC02	CaloSC02@g2sc-priv	0	0.0	0.000
CaloSC03	CaloSC03@g2sc-priv	0	0.0	0.000
CaloSC04	CaloSC04@g2sc-priv	0	0.0	0.000
CaloSC05	CaloSC05@g2sc-priv	0	0.0	0.000
CaloSC06	CaloSC06@g2sc-priv	0	0.0	0.000
CaloSC07	CaloSC07@g2sc-priv	0	0.0	0.000
CaloSC08	CaloSC08@g2sc-priv	0	0.0	0.000
CaloSC09	CaloSC09@g2sc-priv	0	0.0	0.000
CaloSC10	CaloSC10@g2sc-priv	0	0.0	0.000
CaloSC11	CaloSC11@g2sc-priv	0	0.0	0.000
CaloSC12	CaloSC12@g2sc-priv	0	0.0	0.000
CaloSC13	CaloSC13@g2sc-priv	0	0.0	0.000
CaloSC14	CaloSC14@g2sc-priv	0	0.0	0.000
CaloSC15	CaloSC15@g2sc-priv	0	0.0	0.000
CaloSC16	CaloSC16@g2sc-priv	0	0.0	0.000
CaloSC17	CaloSC17@g2sc-priv	0	0.0	0.000
CaloSC18	CaloSC18@g2sc-priv	0	0.0	0.000
CaloSC19	CaloSC19@g2sc-priv	0	0.0	0.000
CaloSC20	CaloSC20@g2sc-priv	0	0.0	0.000
CaloSC21	CaloSC21@g2sc-priv	0	0.0	0.000
CaloSC22	CaloSC22@g2sc-priv	0	0.0	0.000
CaloSC23	CaloSC23@g2sc-priv	0	0.0	0.000
CaloSC24	CaloSC24@g2sc-priv	0	0.0	0.000
ESQ_slow	ESQ_slow@g2quad-01	1729	1.0	0.000
ESQ	ESQ@g2quad-02-priv	1223	0.7	0.000
IFIX	Ok	173	0.0	0.000
mscb110	Ok	29	0.0	0.000
mscb13e	Ok	2871	0.0	0.000
mscb319	Ok	29	0.0	0.000
mscb323	Ok	29	0.0	0.000
KickerSC_mscb282	Ok	29	0.0	0.000
mscb174	Ok	29	0.0	0.000
Beam	Beam@g2sc-priv	346	0.3	0.000

Logging Channels

Channel	Events	MIB written	Compr.	Disk level
#0: gm2_run01618_48.mid	4992	48906.884	N/A	51.1%

Clients



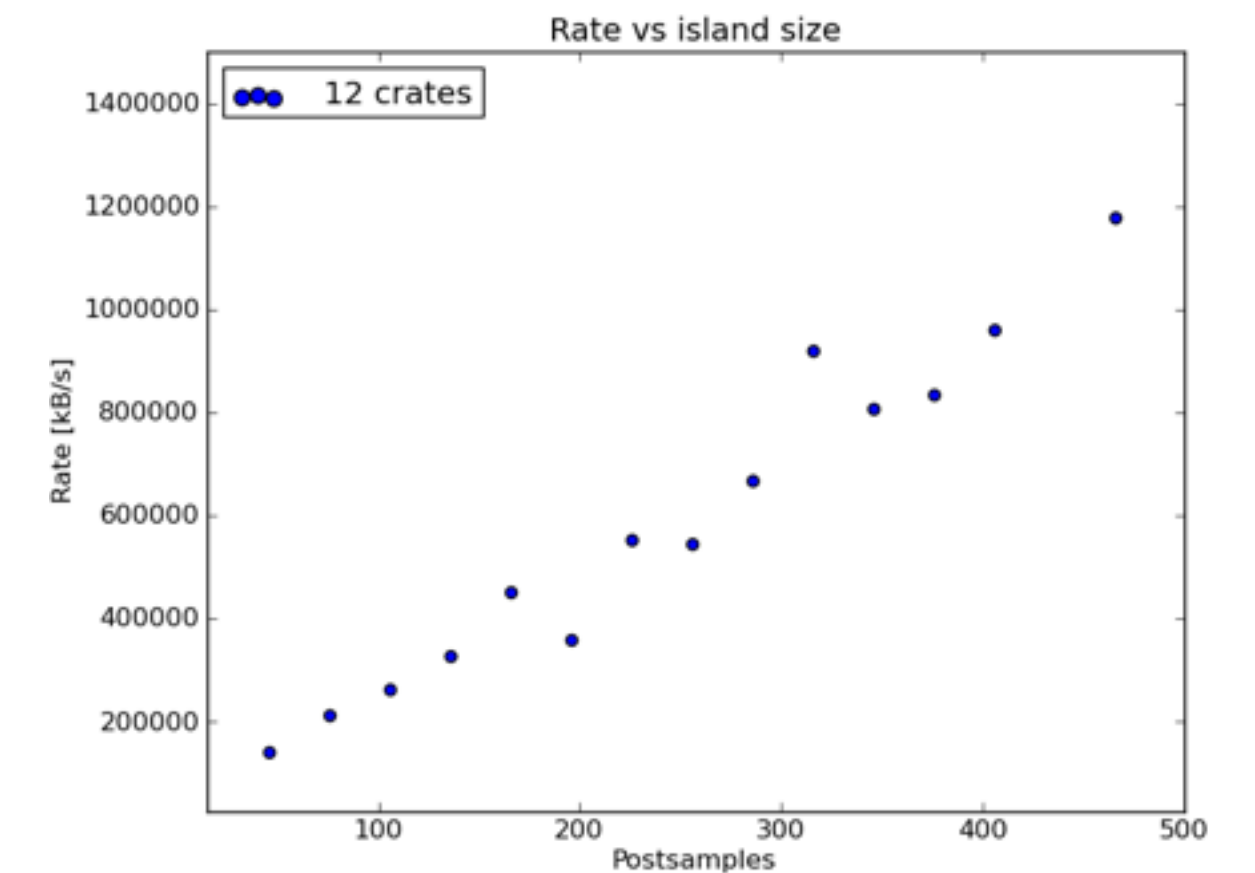
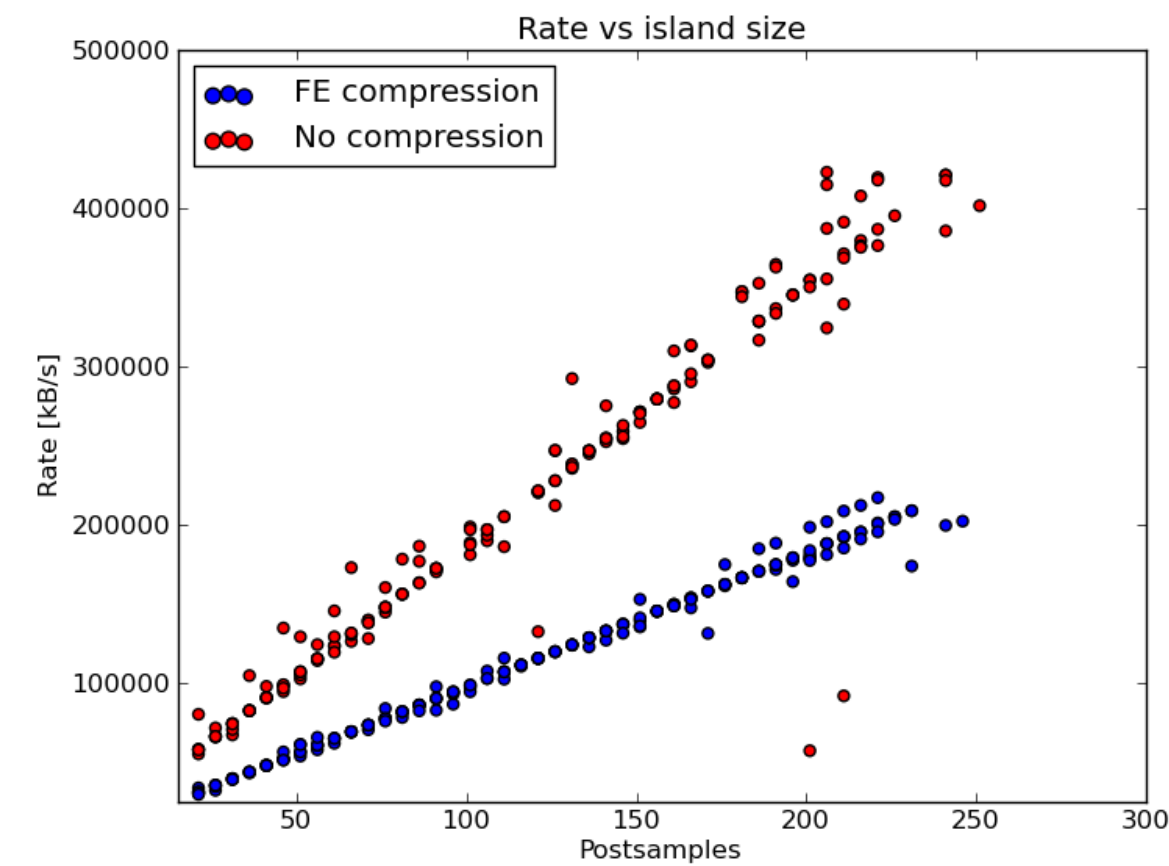
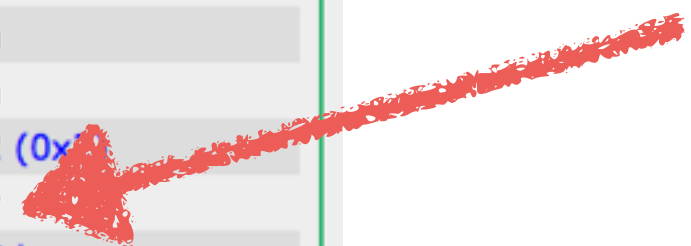
Event builder



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

Online Database Browser		
Find Create Delete Create Elog from this page		
/ Equipment / AMC1308 / Settings / Globals /		
Key	Value	
sync	n	
use AMC13 simulator	n	
GPU Device ID	2 (0x)	
Send to Event Builder	y	
Shelf configuration	rider	
FE lossless compression	n	
raw data store	y	
raw data prescale	1000 (0x3E8)	
raw data prescale offset	8 (0x8)	

Enable here!

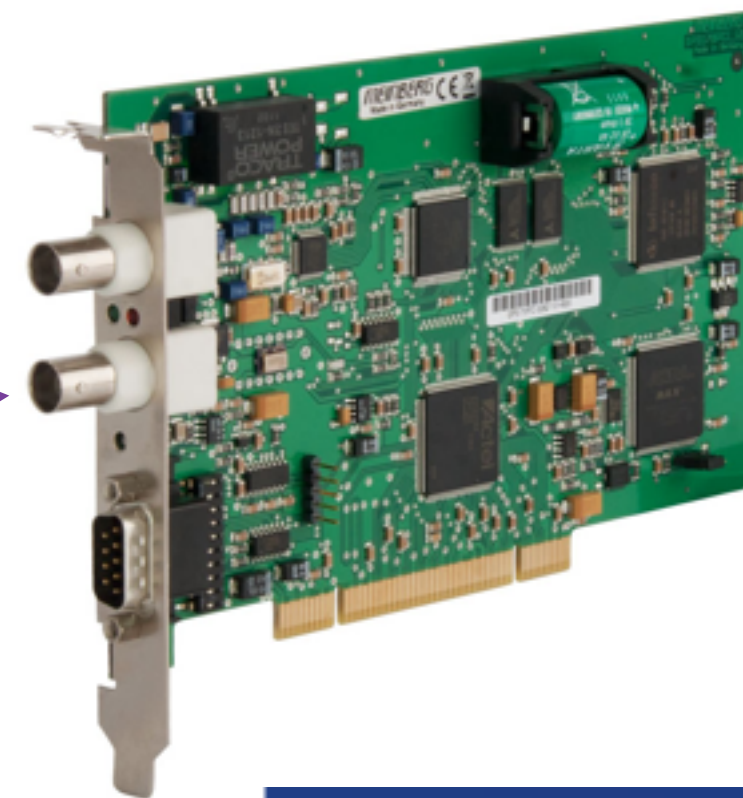


- Total EB rate maxed out at > 1.2 GB/s (limited by network bandwidth)
- The event builder combines up to 270 banks for each event.

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.

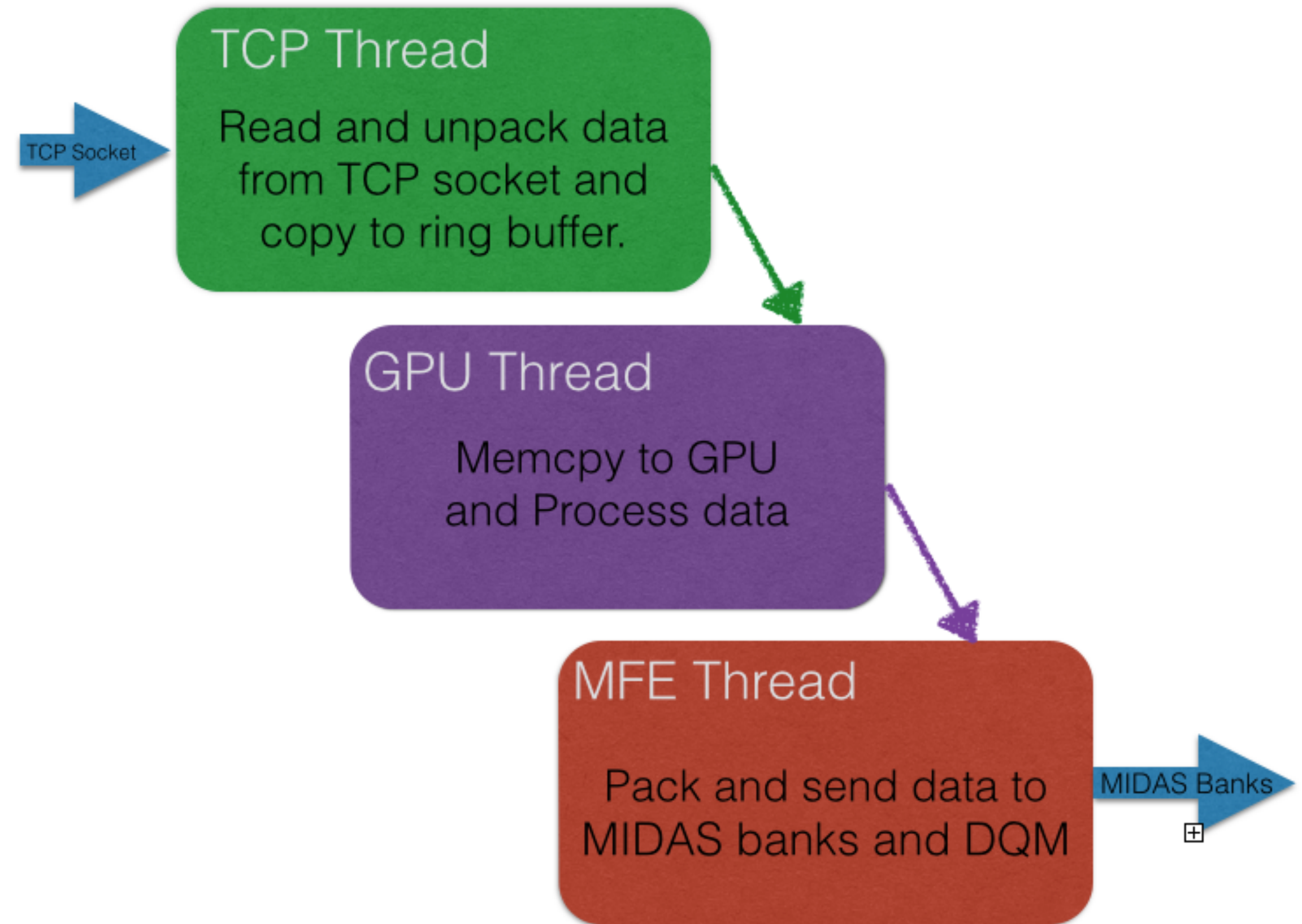


```
Bank:GPS0 Length: 20(I*1)/5(I*4)/5(Type) Type:Unsigned Integer*4  
1-> 0x000018c2 0x580838bc 0x5dcabfb0 0x580838bc 0x5dd48308
```

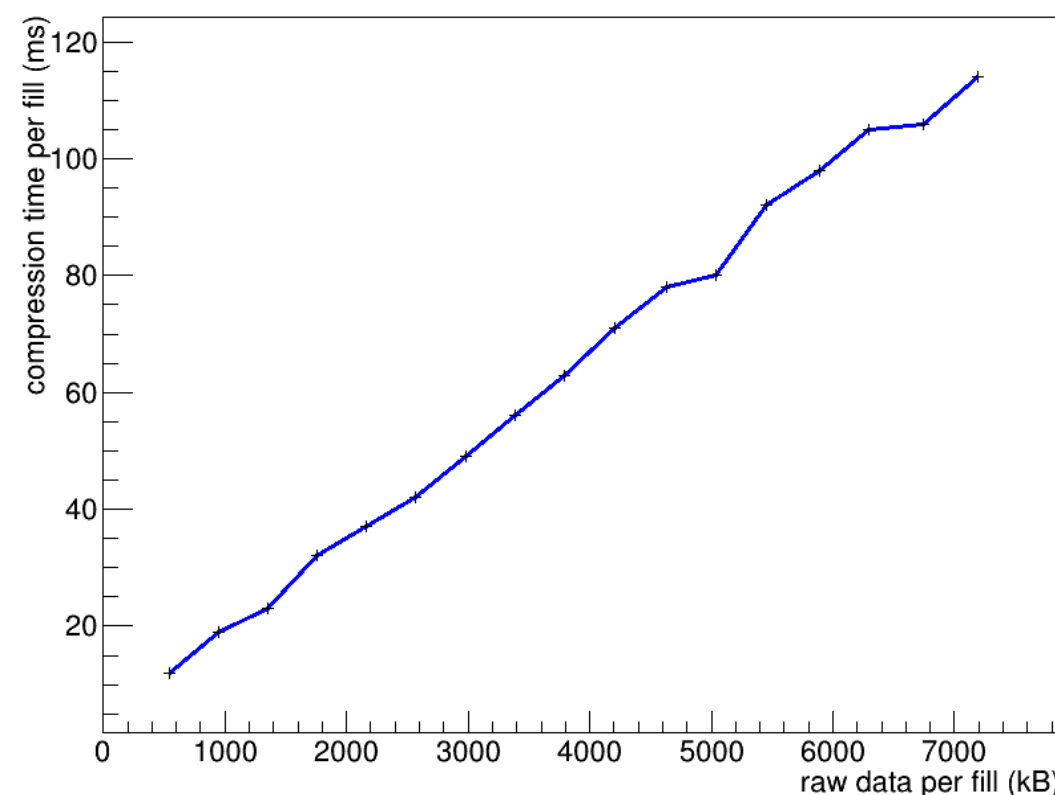


AMC13 Frontend

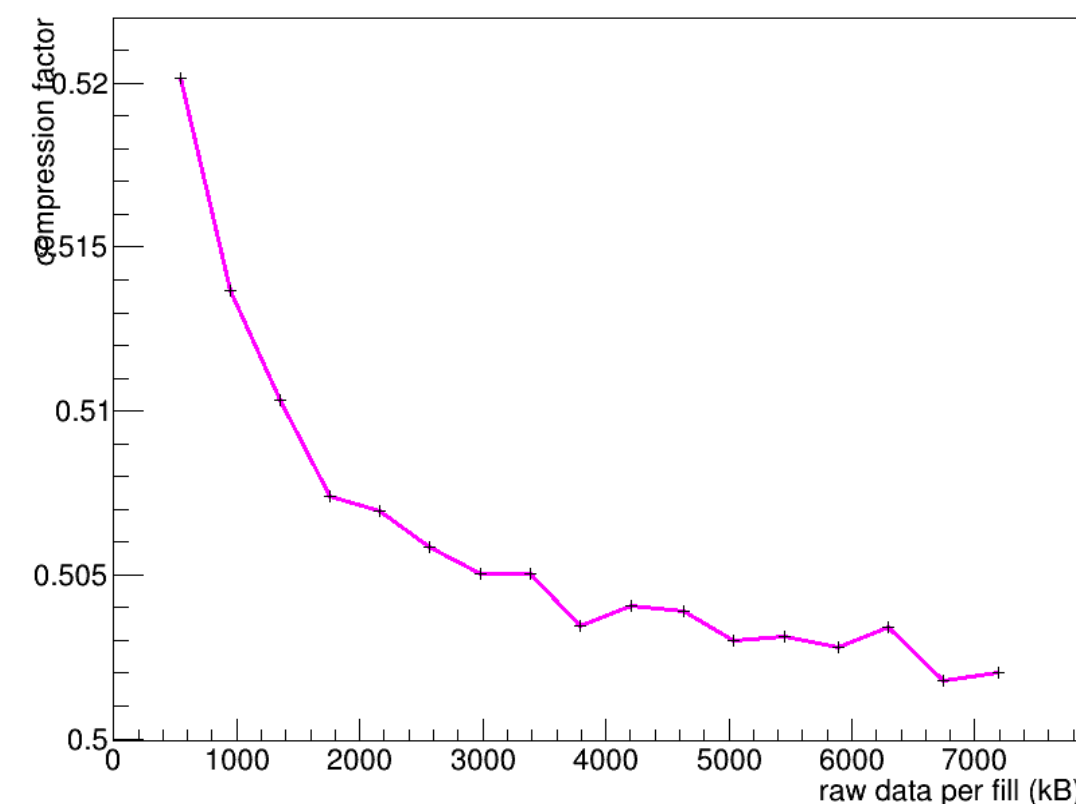
- Each frontend process reads data from one uTCA crate over 10 Gb ethernet with TCPIP.
- Data is processed in Nvidia Tesla K40 GPUs using CUDA code that is integrated into the frontend.
- Midas banks are losslessly compressed using zlib.
- Frontend is multithreaded with mutex locks.
- Full configuration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



compression time versus raw data



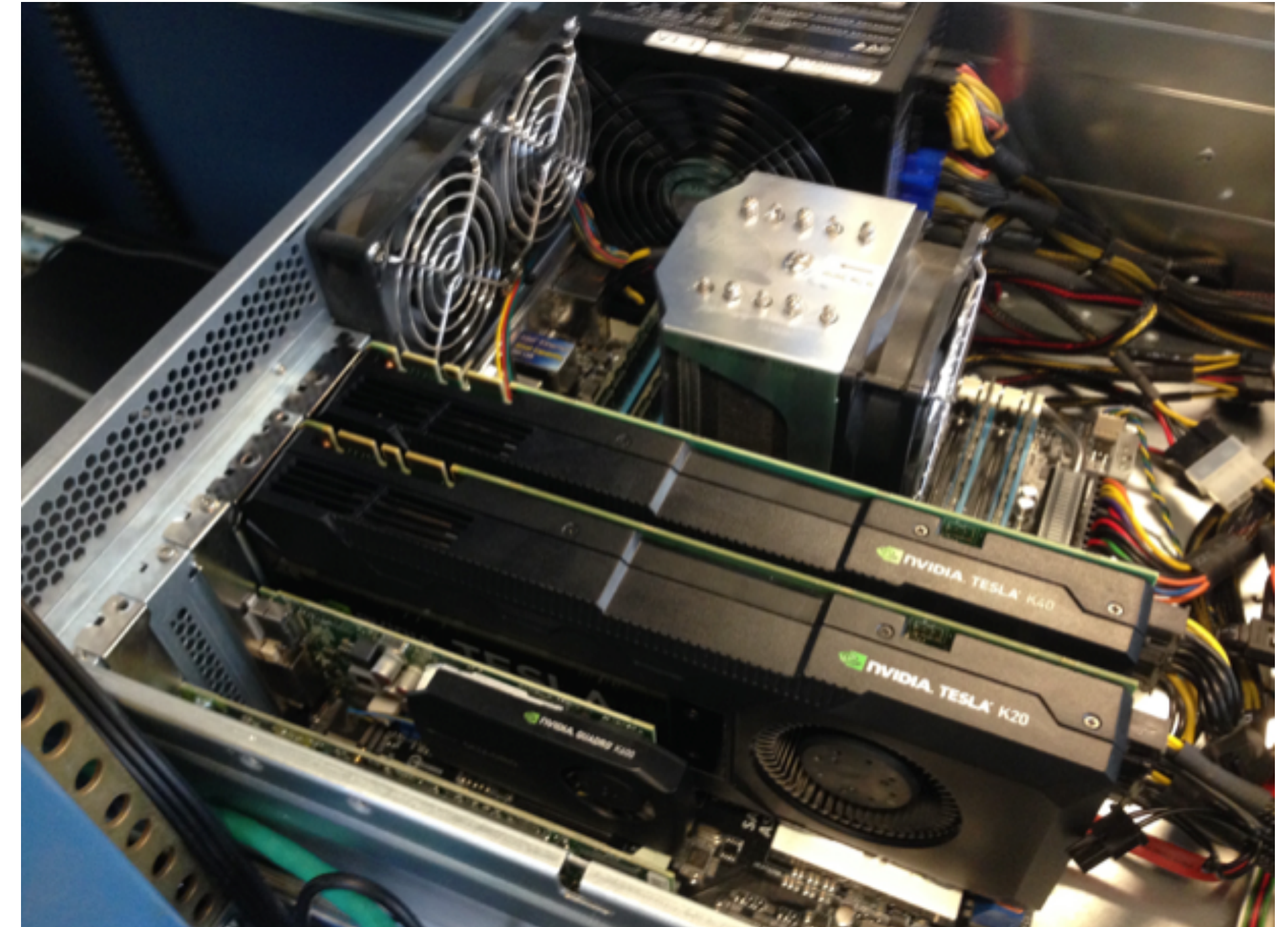
compression factor versus raw data



GPU Processing

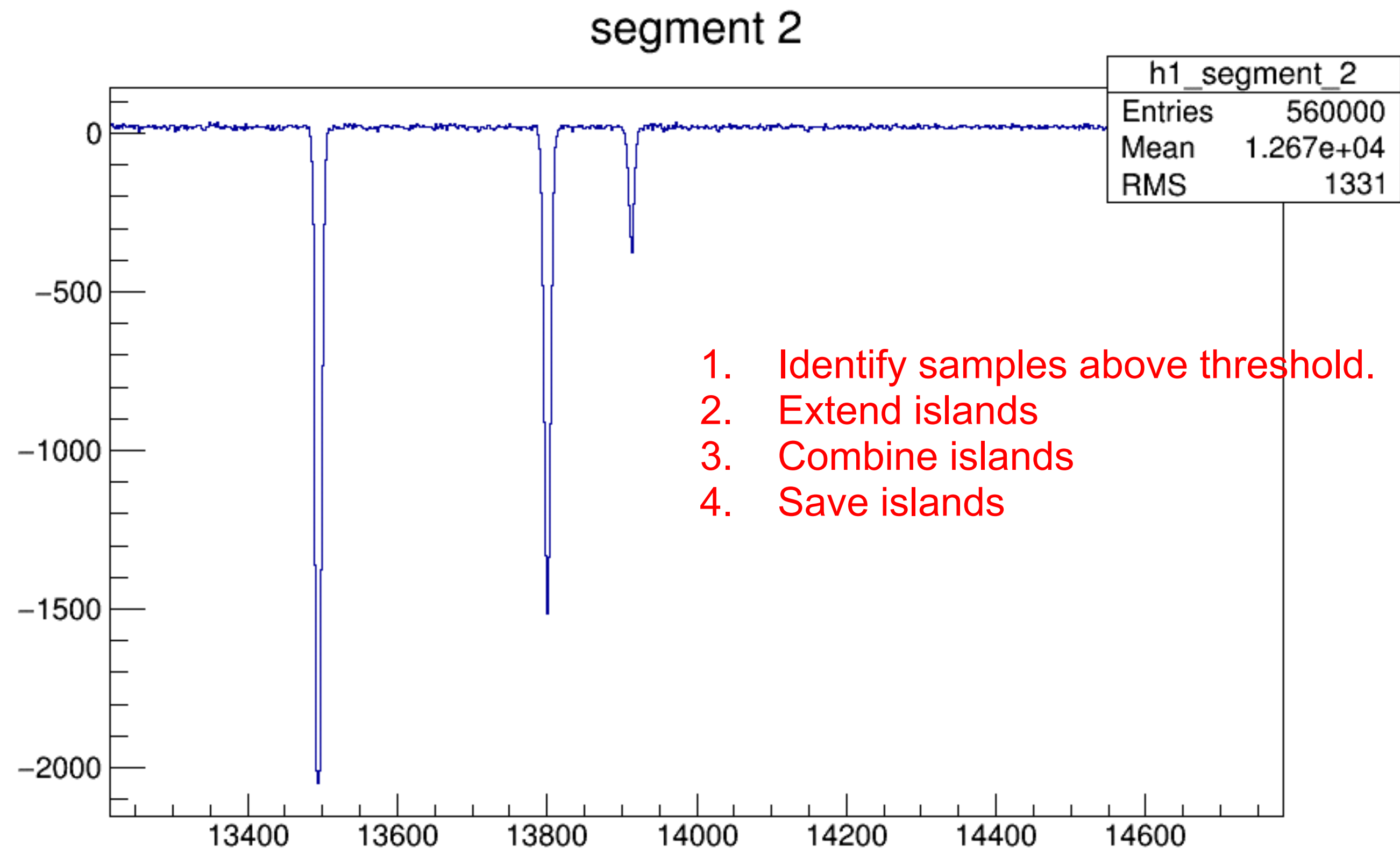


- The frontend includes CUDA routines for data processing.
- Each GPU processes data from one calorimeter.
- Raw fill is copied to GPU memory, where it is reduced using T-method (island chopping), Q-method (histogramming), pedestal calculation, and template fitting.
- The output of each process is written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

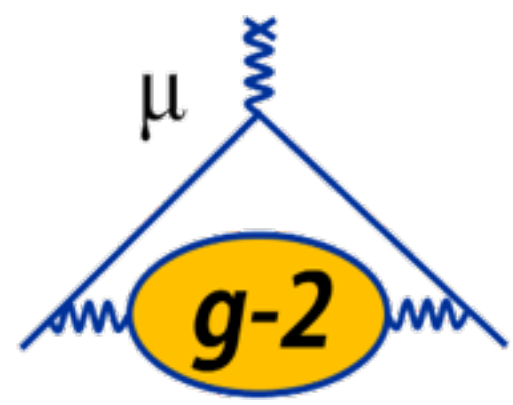


T-Method

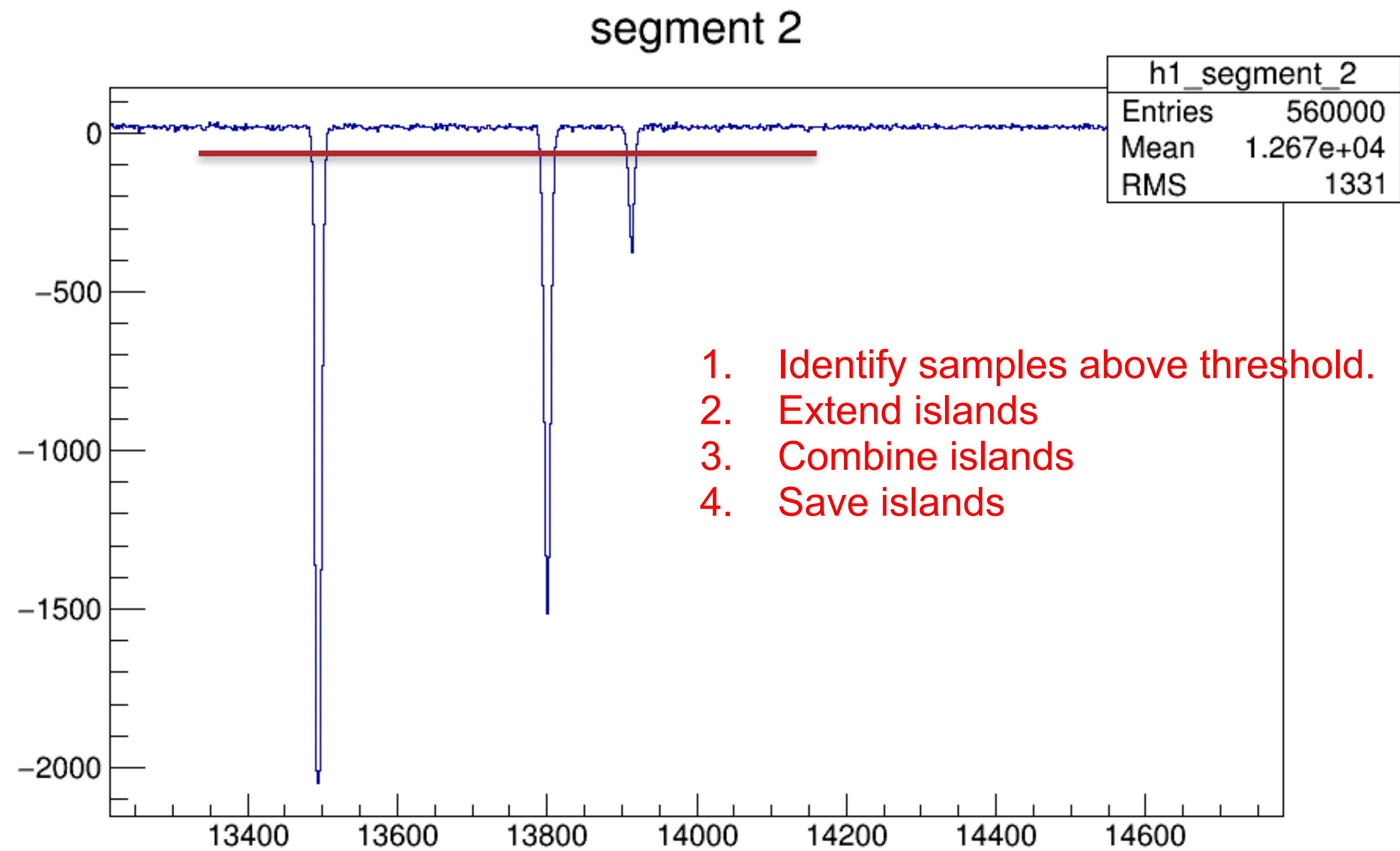
- Identify and save regions of the waveform containing positron hits.
- A typical waveform will have ~ 180 islands.



T-Method

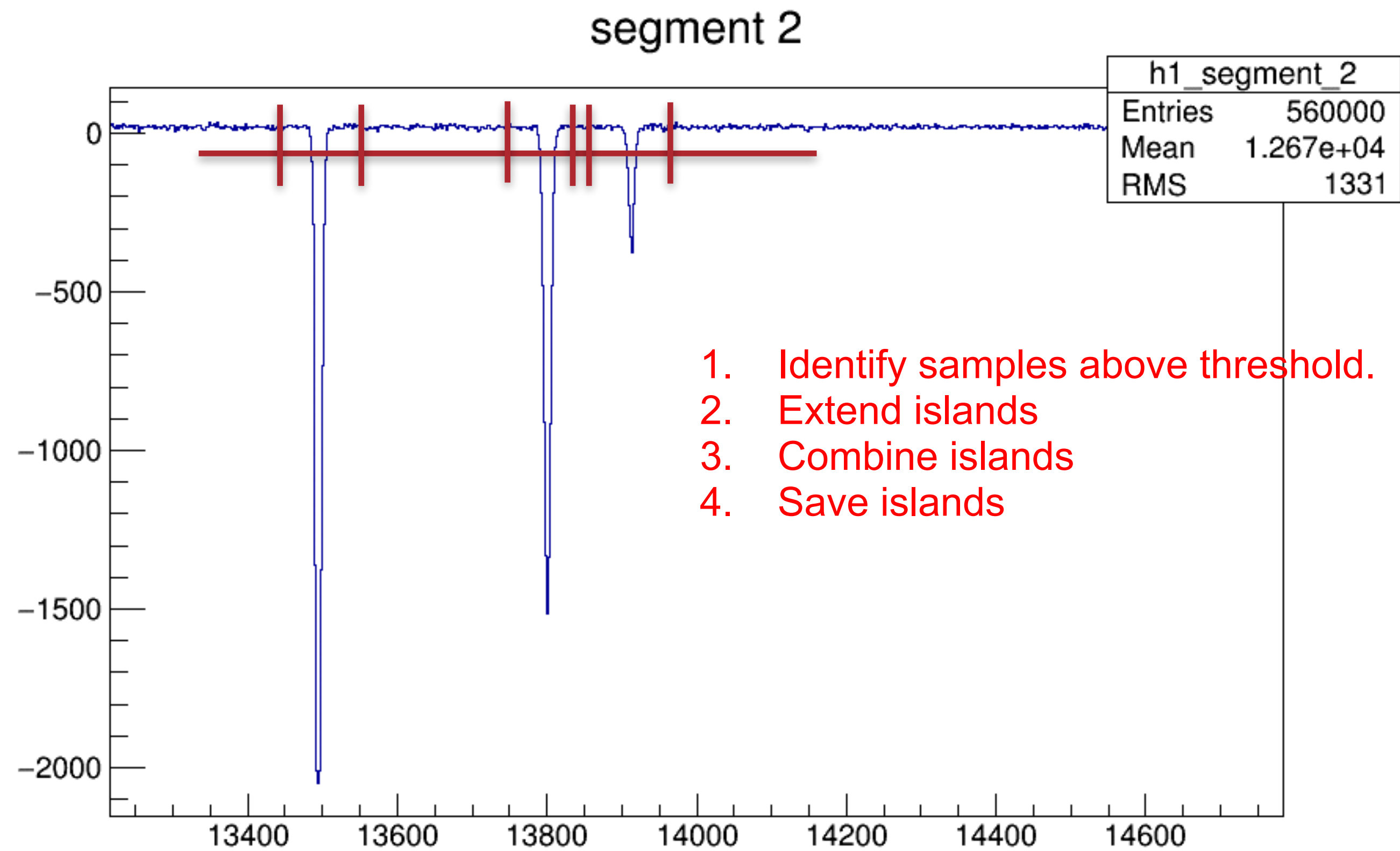


- Identify and save regions of the waveform containing positron hits.
- A typical waveform will have ~ 180 islands.



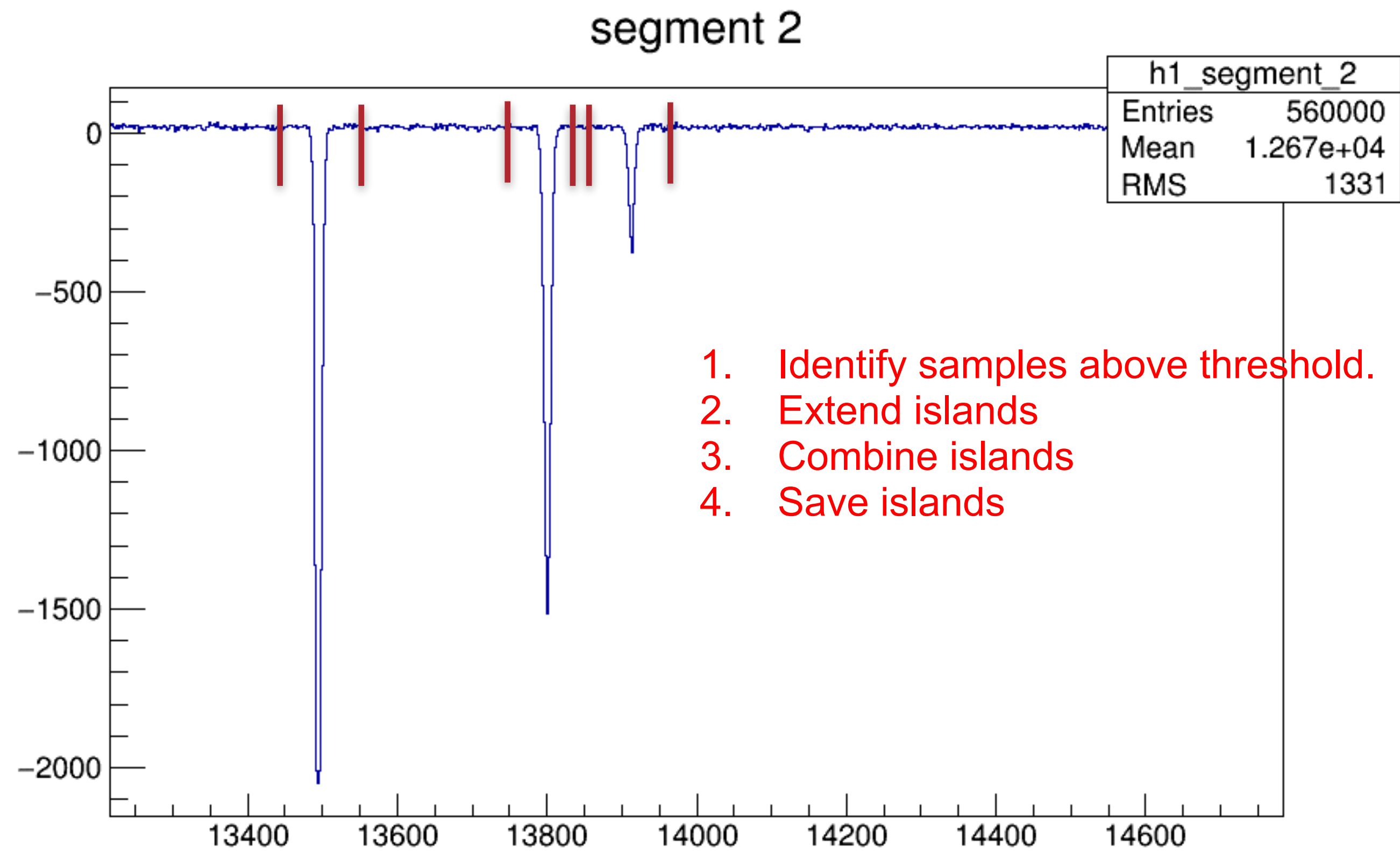
T-Method

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



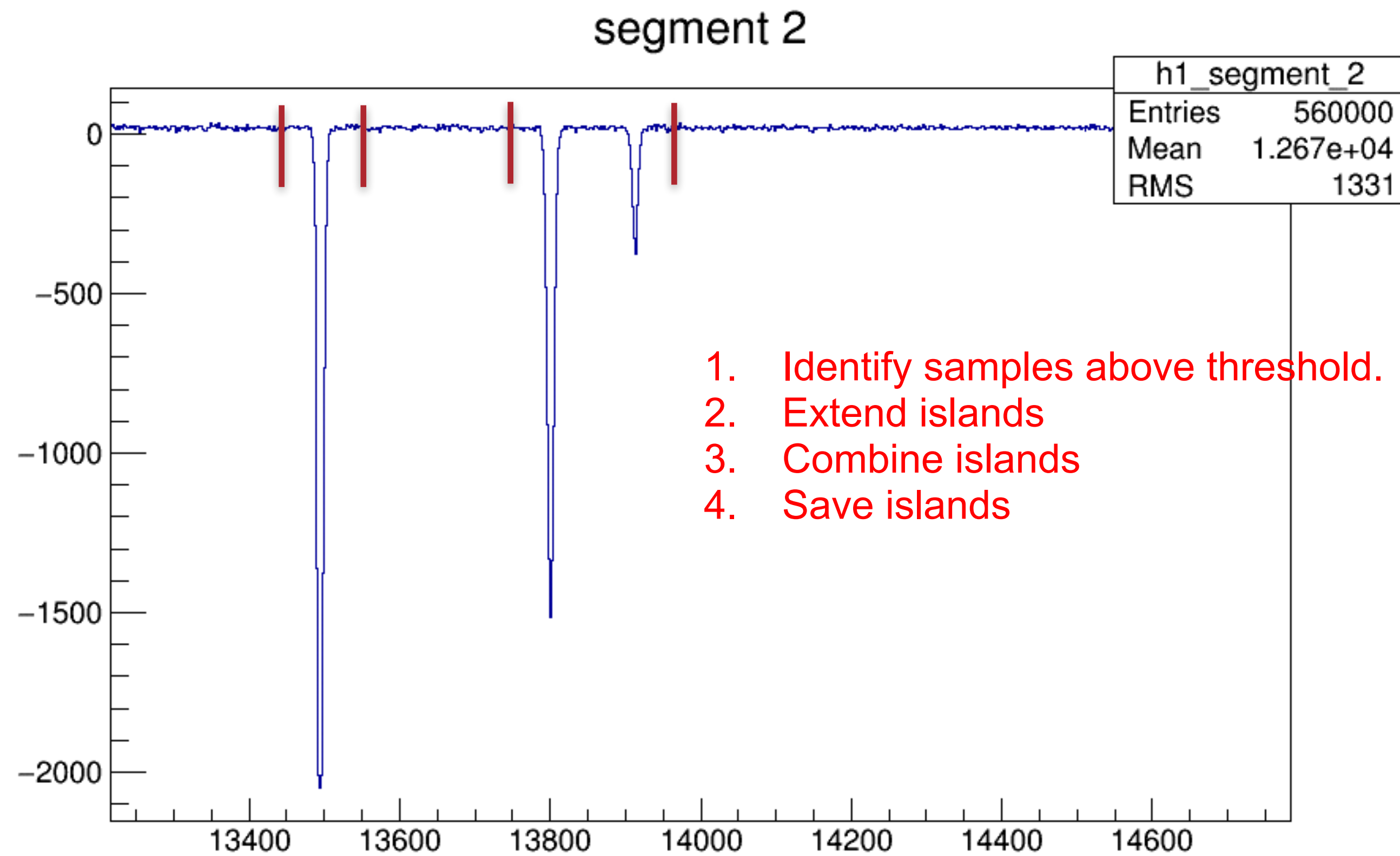
T-Method

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



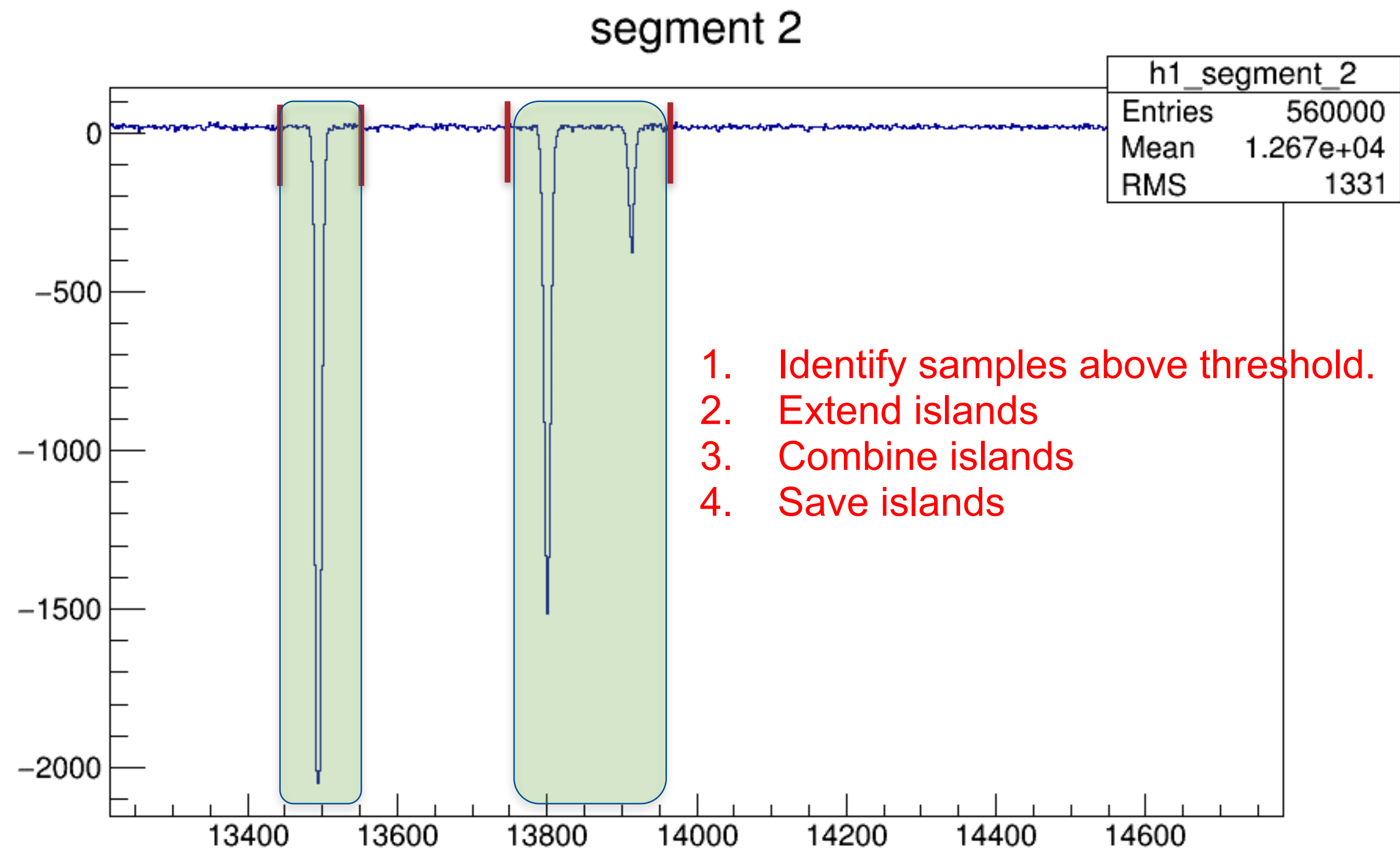
T-Method

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



T-Method

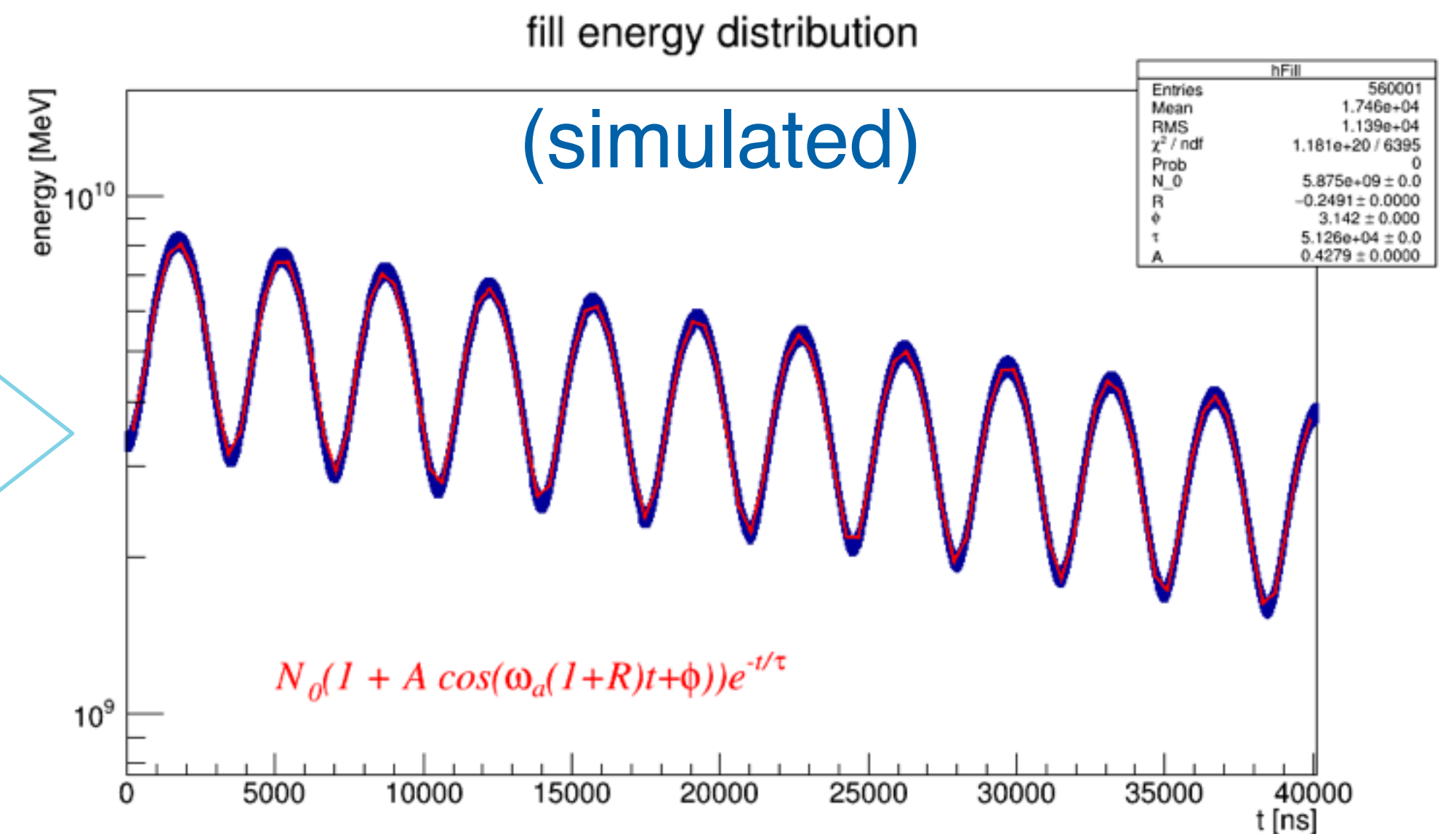
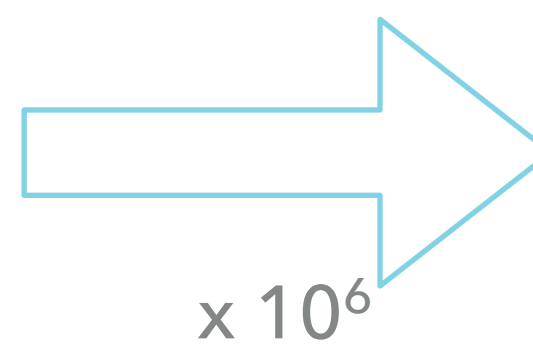
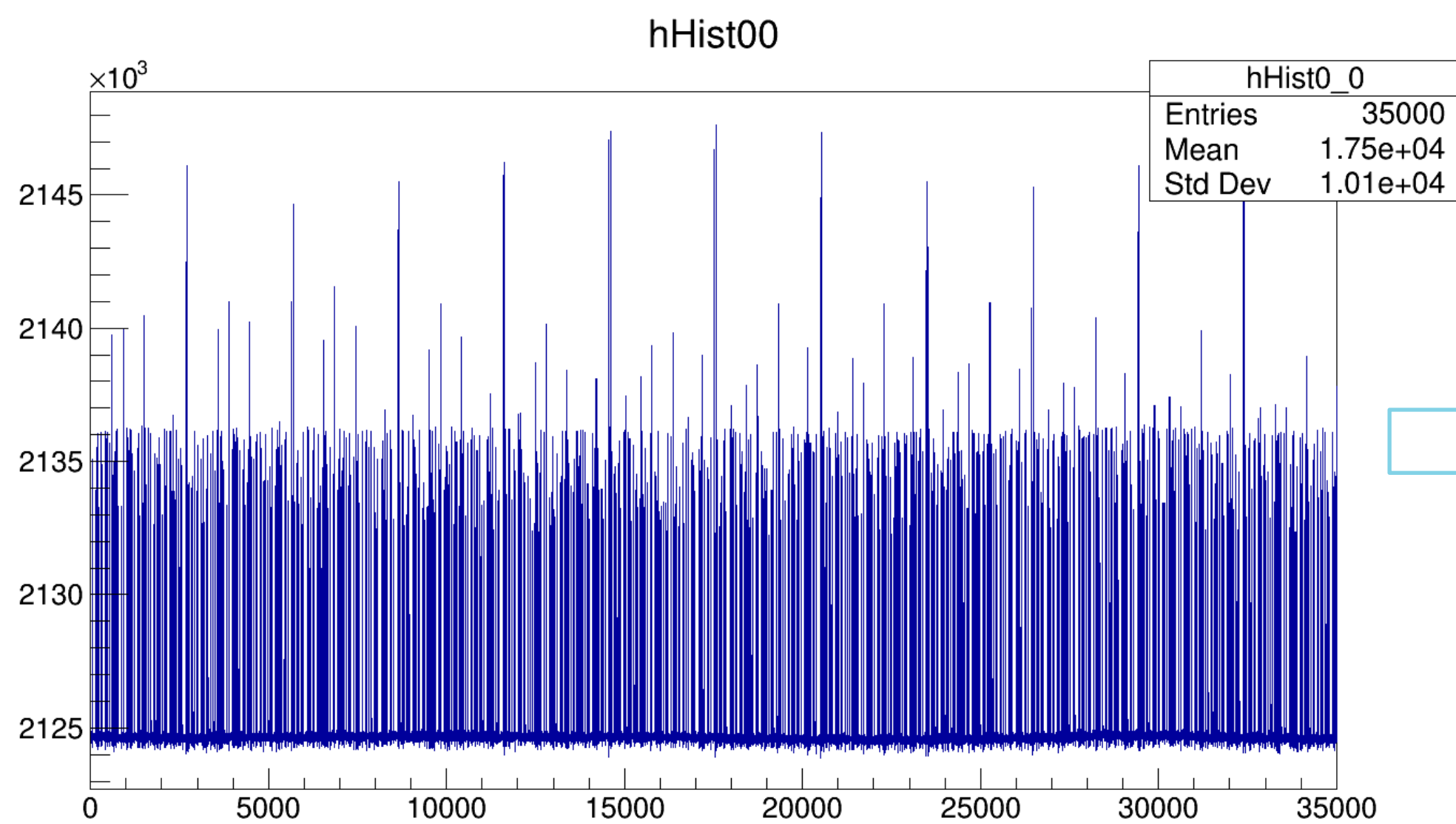
- Identify and save regions of the waveform containing positron hits.
- A typical waveform will have ~ 180 islands.



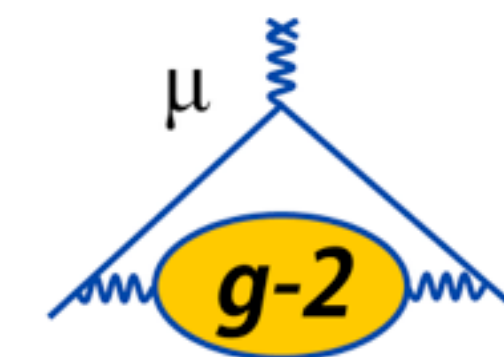
Q-method



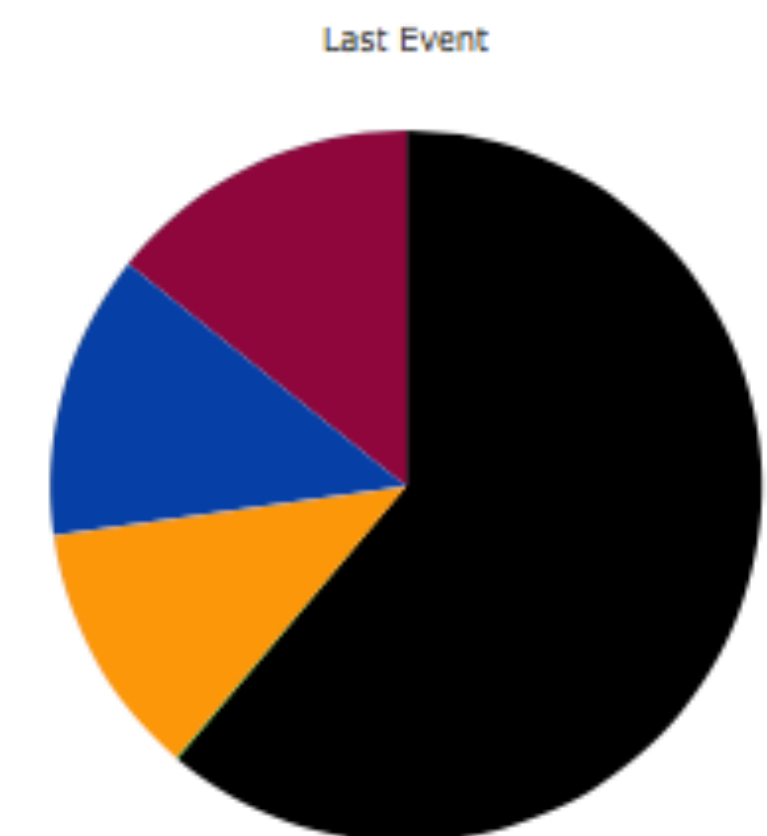
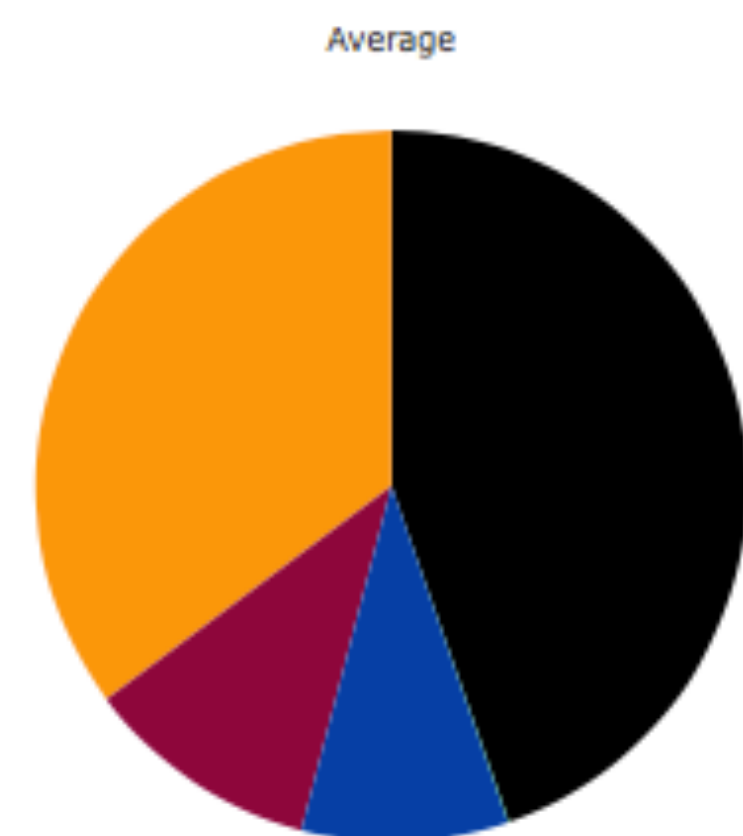
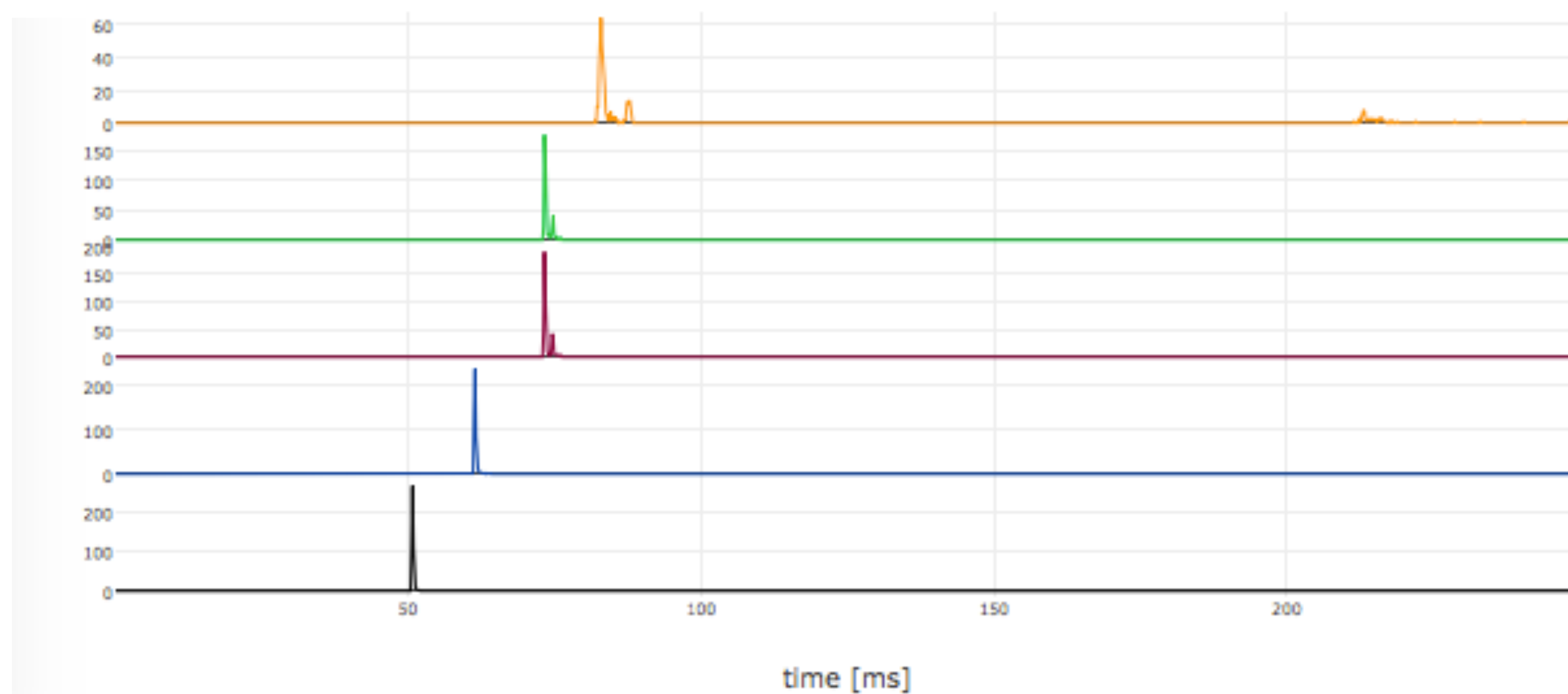
- Full waveforms are decimated in time and summed over many fills to create a histogram that is saved in the data file.
 - i.e. If we decimate in time by 10 and flush every 100 fills, we reduce the data rate by a factor of 1000, so from 20 GB/s to 20 MB/s.
- Use smaller bins at lower times and wider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



Processing time



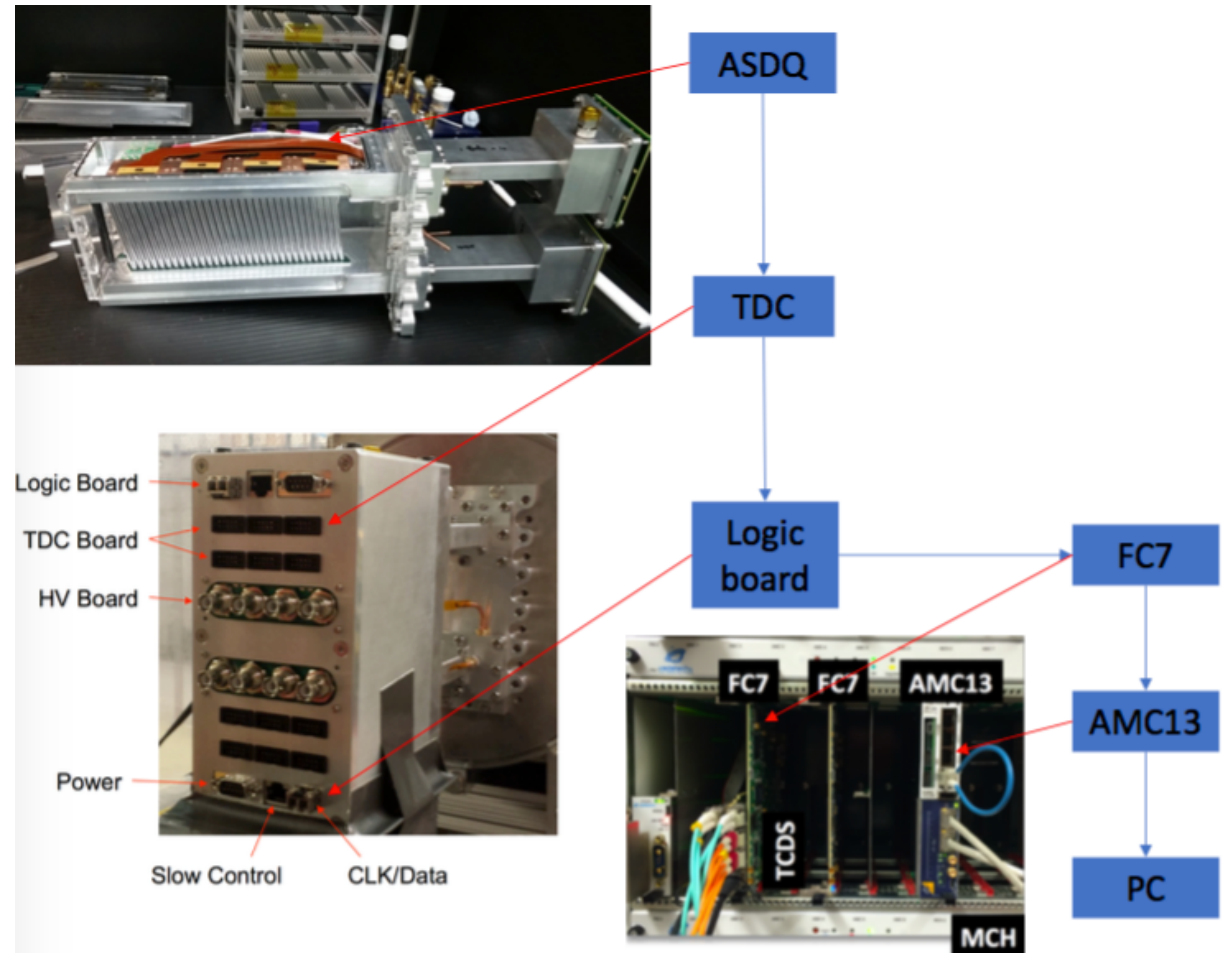
- Must process each event in 83 ms to keep up with average beam rate of 12 Hz.
- Most time is spent reading data from TCP socket and copying it to the GPU.
- Processing time in the GPU is very small.



- got tcp data
- compression done
- processed gpu data
- copied gpu data
- mfe start

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.

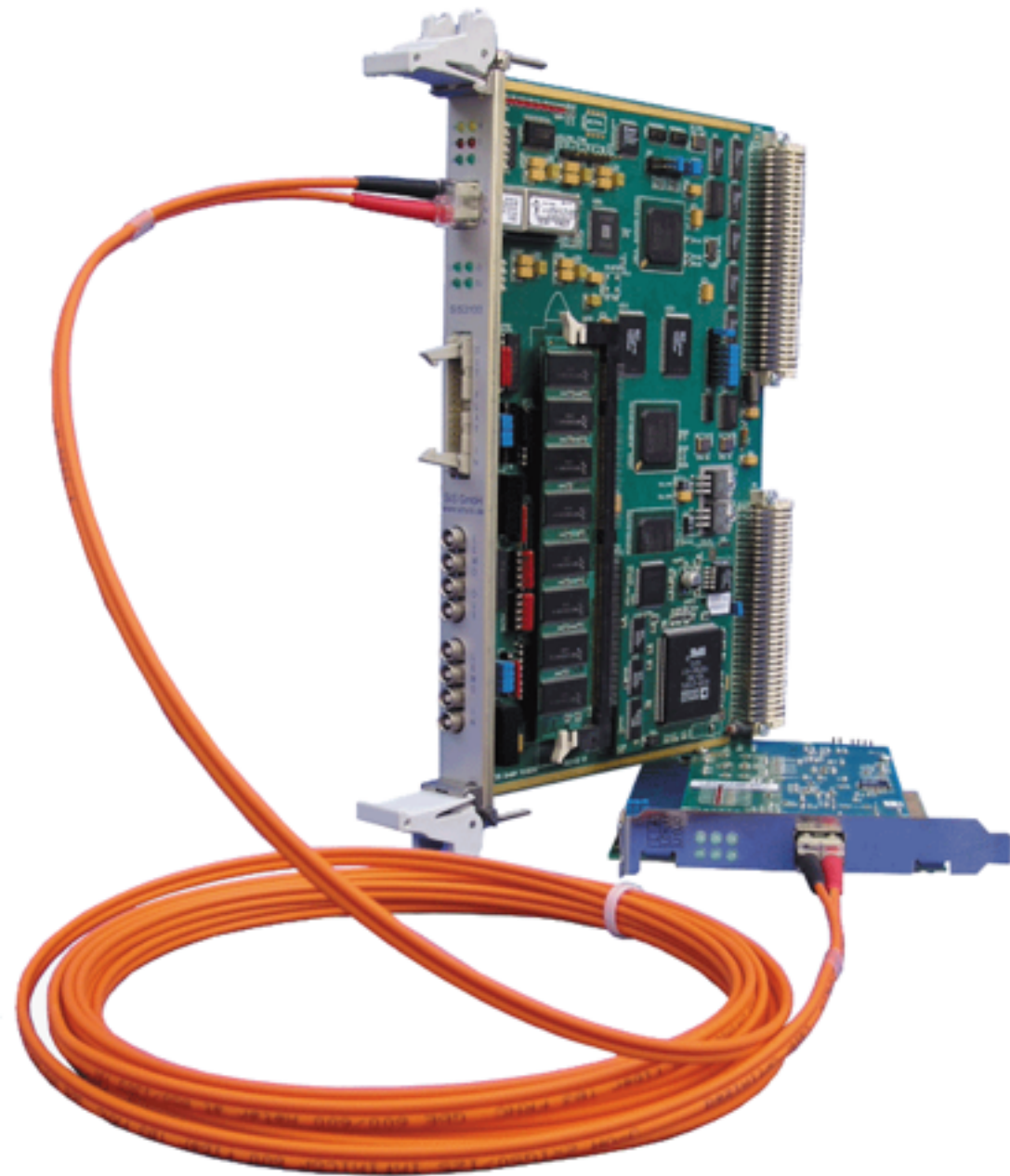


(Thanks R. Chislett for the diagram)

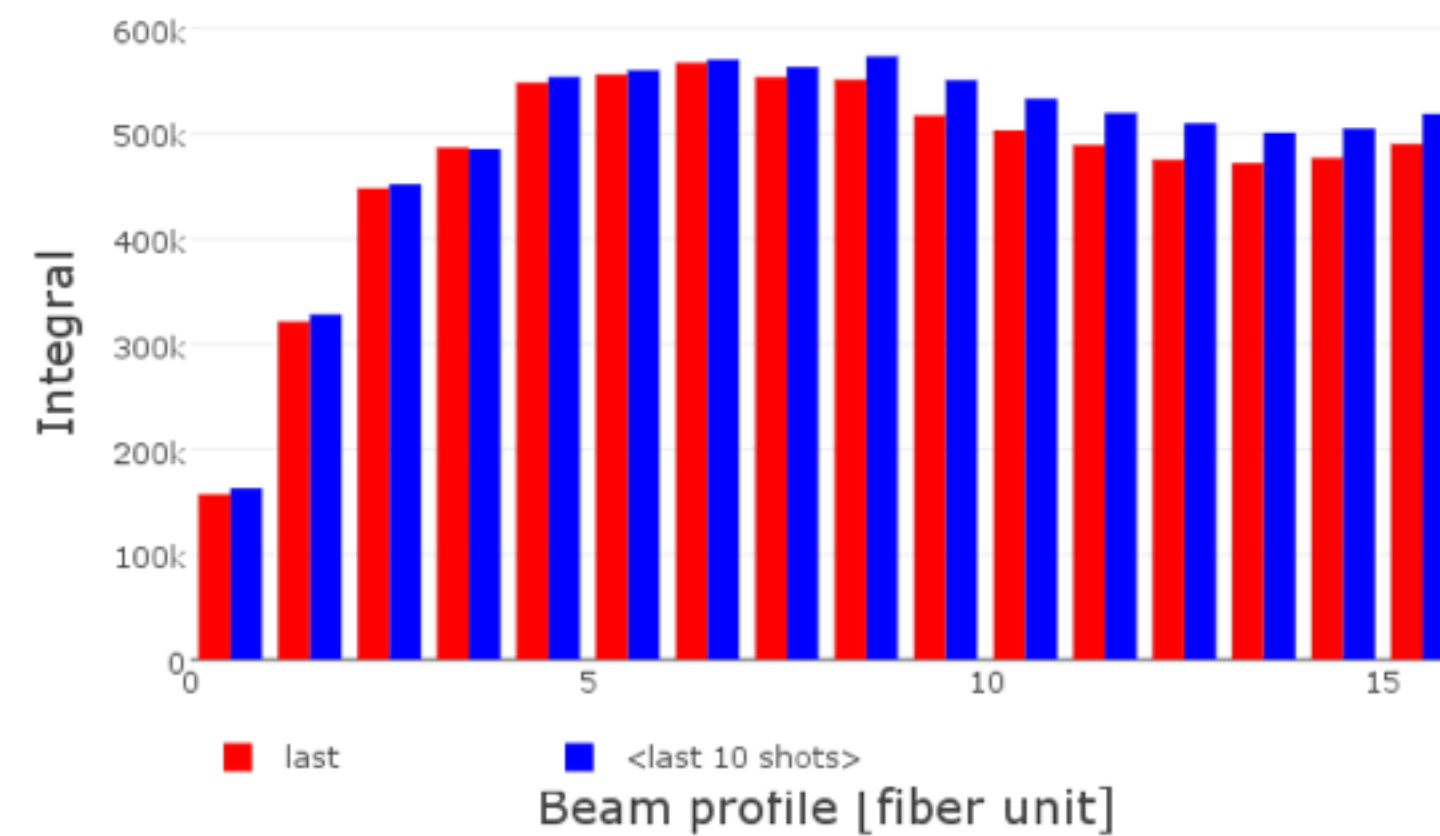
IBMS Frontend



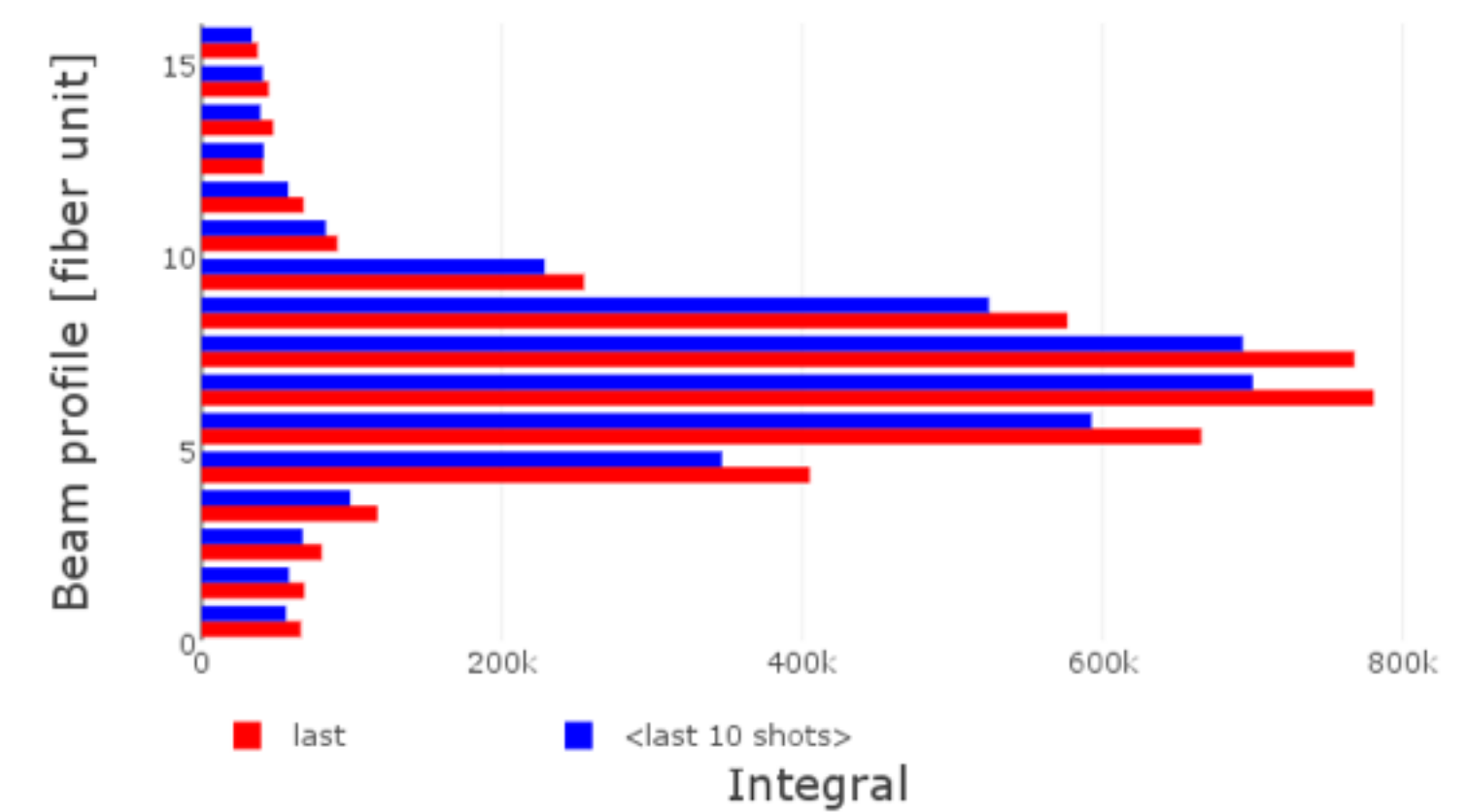
- Data from the inflector beam monitoring system (IBMS) is read out via a CAEN digitizer.
- A custom MIDAS frontend was written to integrate this detector into the DAQ.



IBMS 1 X Profile



IBMS 1 Y Profile



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.

Run Status

Run 395 Running
Start: Thu Jun 1 08:52:27 2017 Running time: 0h26m35s
Alarms: On Restart: No Data dir: /home/newg2/gm2Data/
Experiment Name: g2-field
Trolley Status: n
09:19:00 [Fixed Probes,DEBUG] issued trigger

Equipment

Equipment	Status	Events	Events[/s]	Data[MB/s]
Fixed Probes	Fixed Probes@g2field-fe2-priv	398	0.3	0.995
TrolleyInterface	Frontend stopped	0	0.0	0.000
GalilFermi	Frontend stopped	0	0.0	0.000
Surface Coils	Frontend stopped	0	0.0	0.000
Monitor	Frontend stopped	0	0.0	0.000
Fluxgate	Frontend stopped	0	0.0	0.000
PS Feedback	Frontend stopped	0	0.0	0.000

Logging Channels

Channel	Events	MiB written	Compr.	Disk level
#0: run00395_00.mid.gz	399	949.078	N/A	17.1 %

Muon g-2 DQM Run395 Event5 Subsystems

Ring Yoke Probe

Update Options: Refresh, Automatic On, Every 5 seconds, Display Options: All Probes

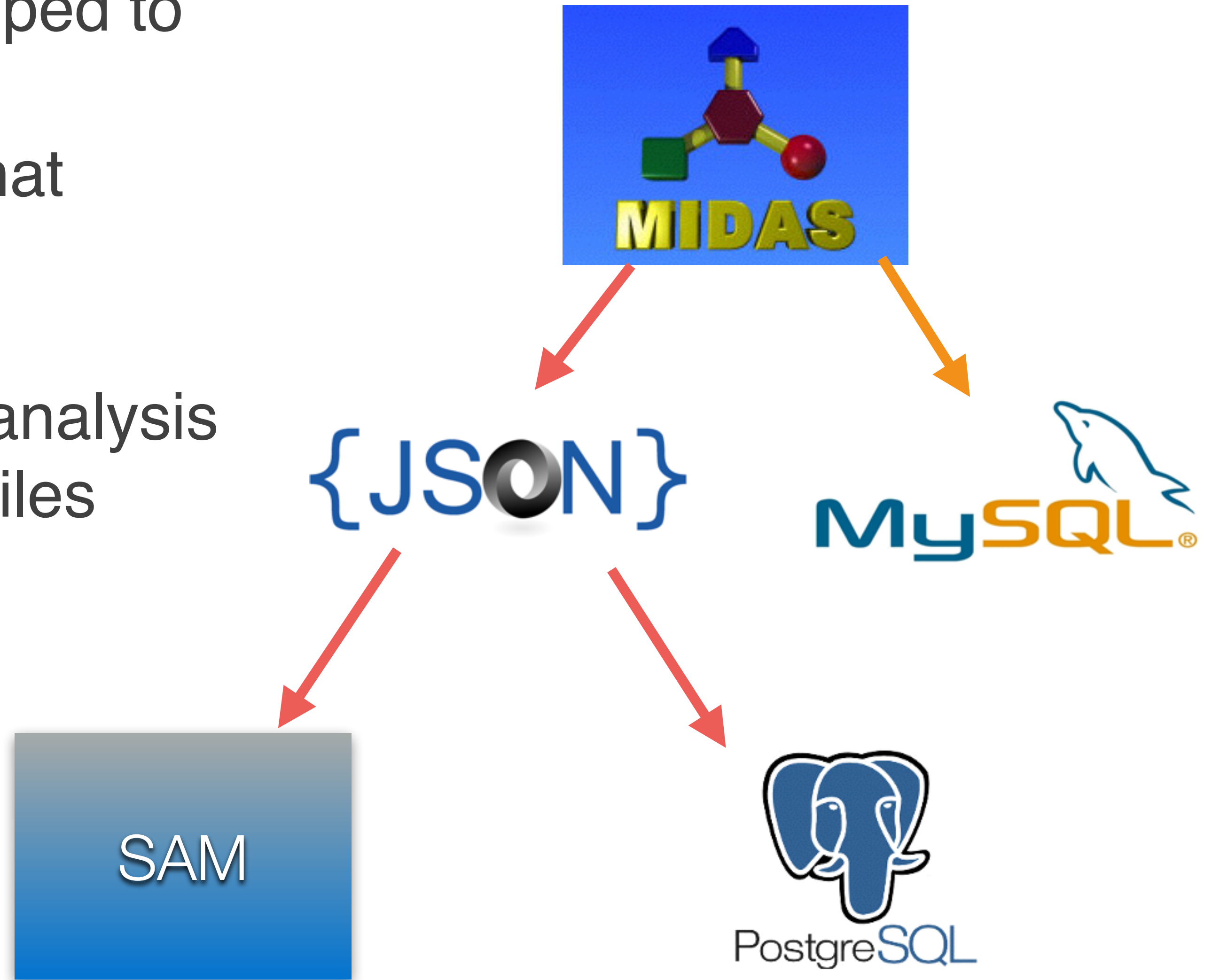
Subsystems:

- Average Frequency: ---
- Healthy Probes : 324
- Front-end Online
- Run in Progress
- Trolley Run

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

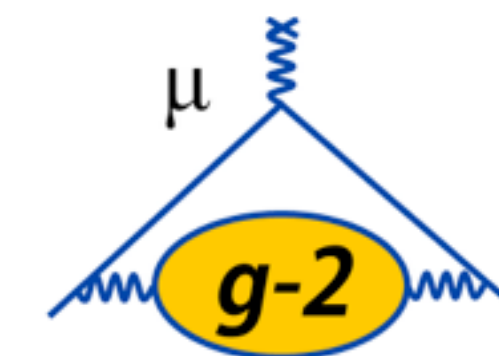


- The MIDAS alarm system was used as the primary alarm system.
- Alarms were set on temperatures and voltages from MSCB devices.
- Other slow frontends set alarms automatically when encountering an error.
- Periodic alarm reminded shifters to perform shift checks.
- Had problems at first with alarm audio,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

The screenshot shows the MIDAS web interface with the following elements:

- Navigation tabs: Status (selected), ODB, Messages, Alarms, Programs, Sequencer, Config, Help.
- Buttons: Test, Restart Fast Frontends, ChanMap, Straw Tracker Power, Straw Tracker Settings, WFD5.
- Warning: Computer room temperature high (yellow bar) with a Reset button.
- Alarm: Computer room temperature too high (red bar) with a Reset button.
- Run Status section (blue header):
 - Run 2075 Stopped (red box) with a Start button.
 - Start: Fri Jul 14 23:39:34 2017
 - Stop: Fri Jul 14 23:34:51 2017
 - Alarms: On (green bar)
 - Restart: Yes (green bar)
 - Data dir: /data2/gm2
 - Experiment Name: GM2
 - git hash:
 - CCC Run State: Idle
- Log entry: 20:28:42 mhttpd:Alarm: Computer room temperature too high (green bar).

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.

AMC1300 TQ01 TQ02 TQ03 TQ04 Threshold Det x-segmt Det y-segmt

/Equipment/AMC1300/Settings/TQ01/RiderXX/ChannelXX/threshold value

Channel #	Rider 01	Rider 02	Rider 03	Rider 04	Rider 05	Rider 06	Rider 07	Rider 08	Rider 09	Rider 10	Rider 11	Rider 12
00	-200	-200	-200	-200	-200	-200	-200	-200	-200	-200	-200	-200
01	-200	-200	-200	-200	-200	-200	-200	-200	-200	-200	-200	-200
02	-200	-200	-200	-200	-200	-200	-200	-200	-200	-200	-200	-200
03	-200	-200	-200	-200	-200	-200	-200	-200	-200	-200	-200	-200
04	-200	-200	-200	-200	-200	-200	-200	-200	-200	-200	-200	-200

Enabled Chan used Positive crossing

/Equipment/AMC1300/Settings/RiderXX/ChannelXX/enabled

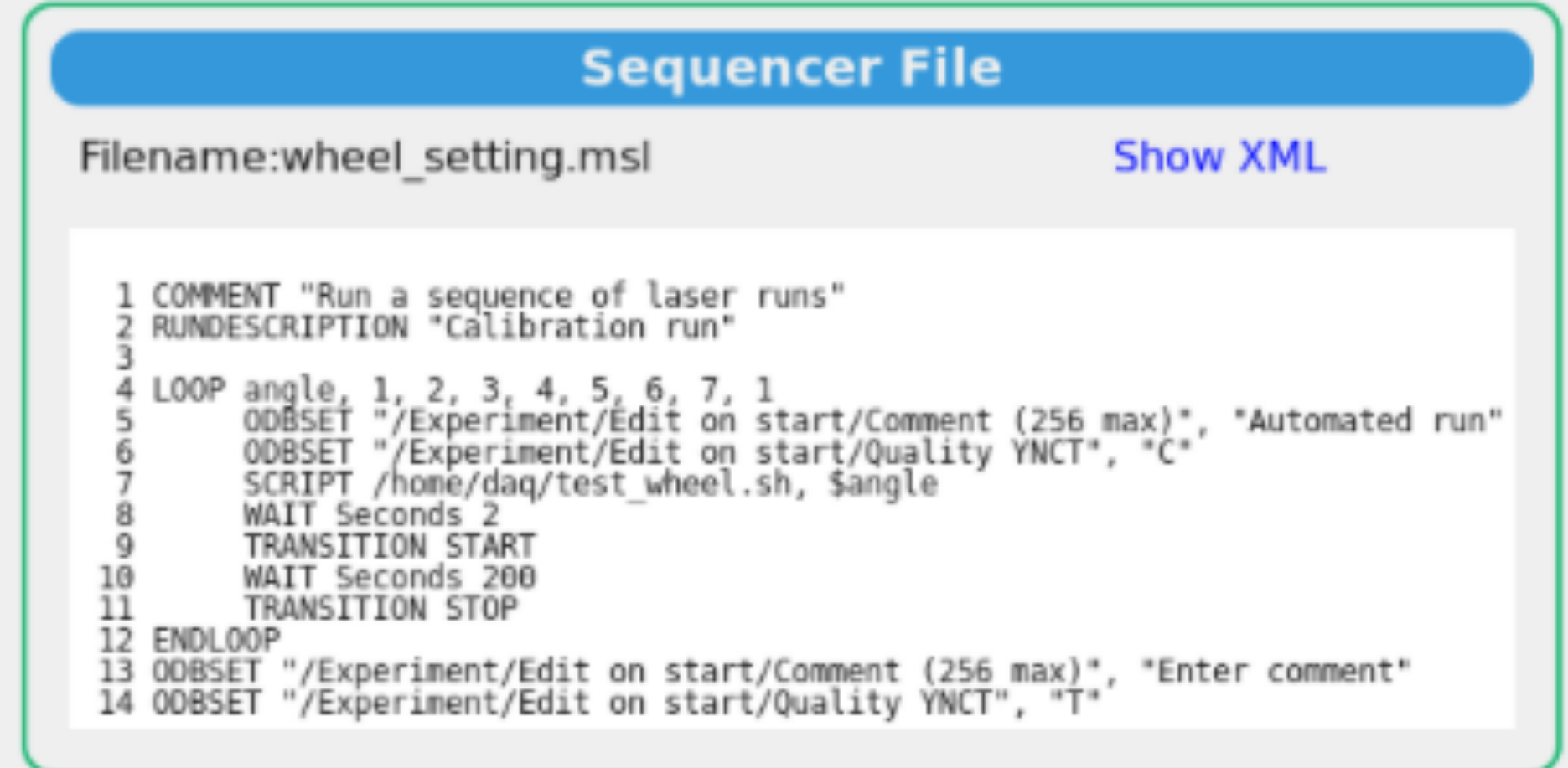
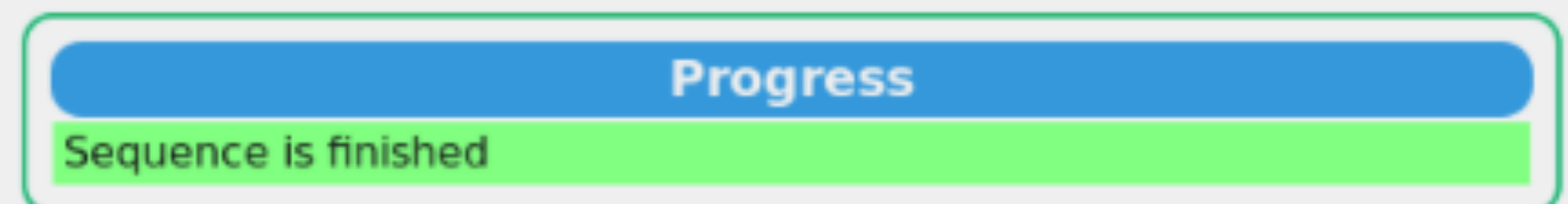
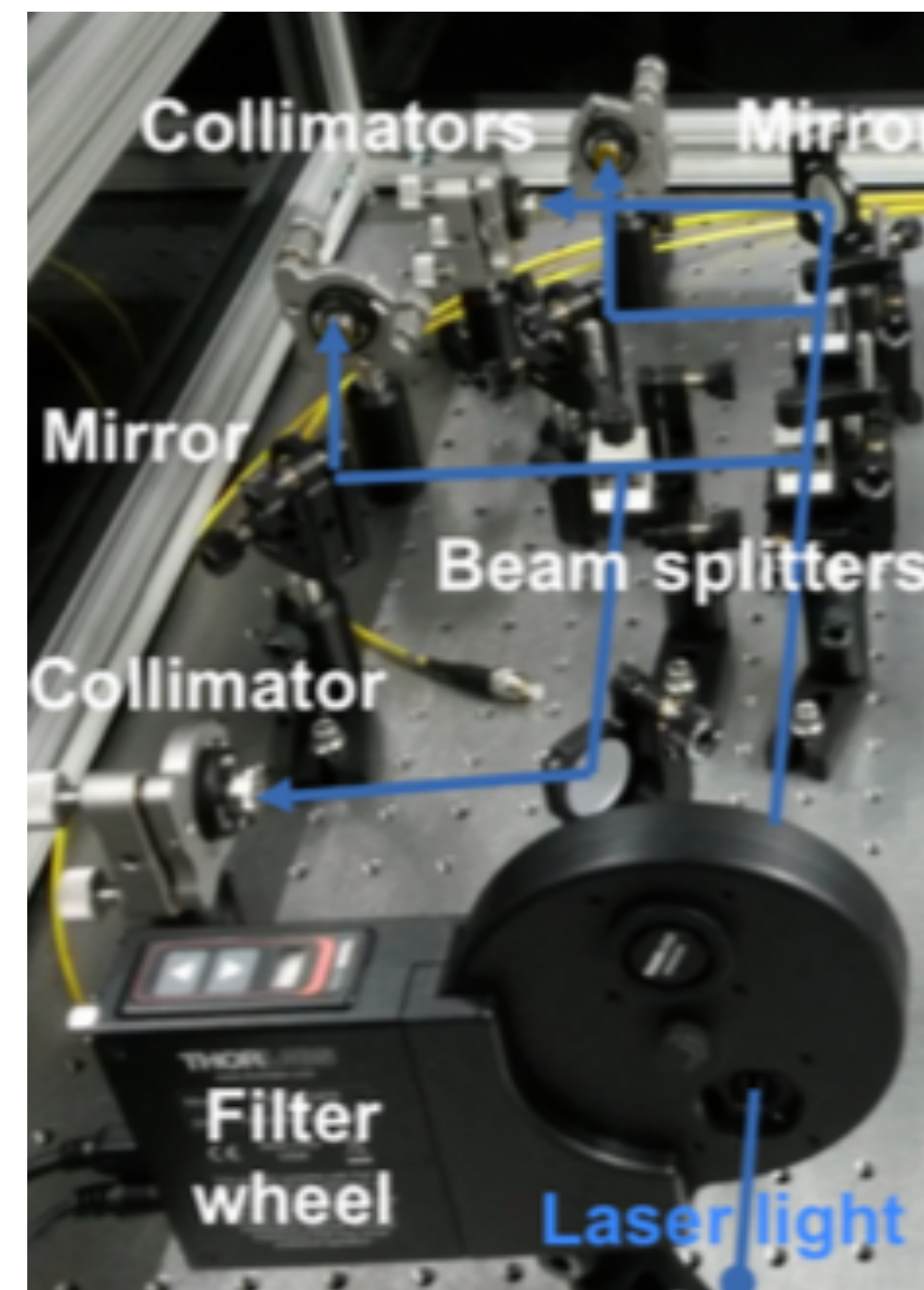
Channel #	Rider 01	Rider 02	Rider 03	Rider 04	Rider 05	Rider 06	Rider 07	Rider 08	Rider 09	Rider 10	Rider 11	Rider 12
00	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
01	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
02	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
03	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
04	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

WRITE to ODB

MIDAS Sequencer



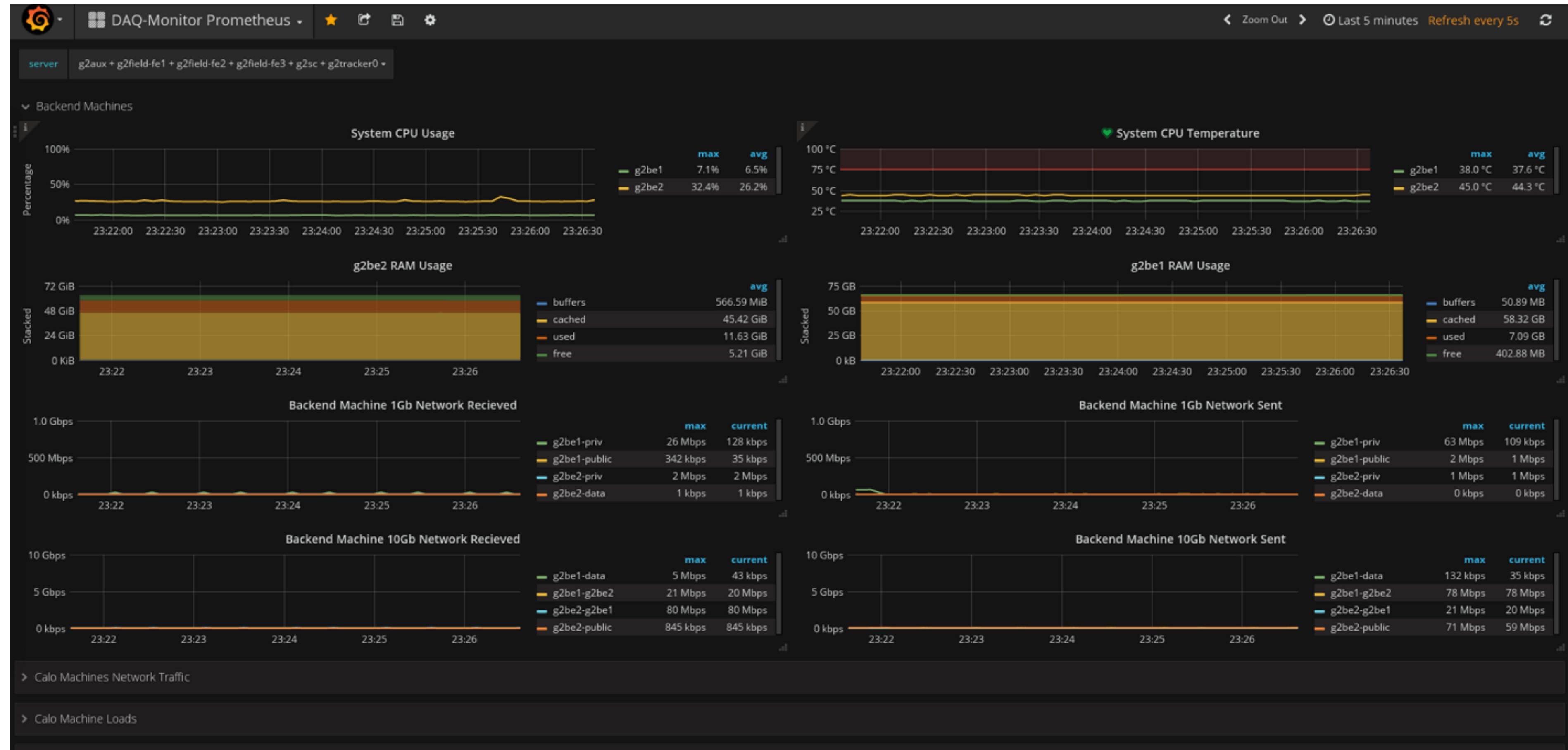
- The sequencer was used extensively for calibration runs such as filter wheel scans and bias voltage scans.
- A typical sequence would be:
 - Execute script to move wheel.
 - Update ODB values
 - Take data for 10 minutes
 - Repeat



DAQ Health Monitor



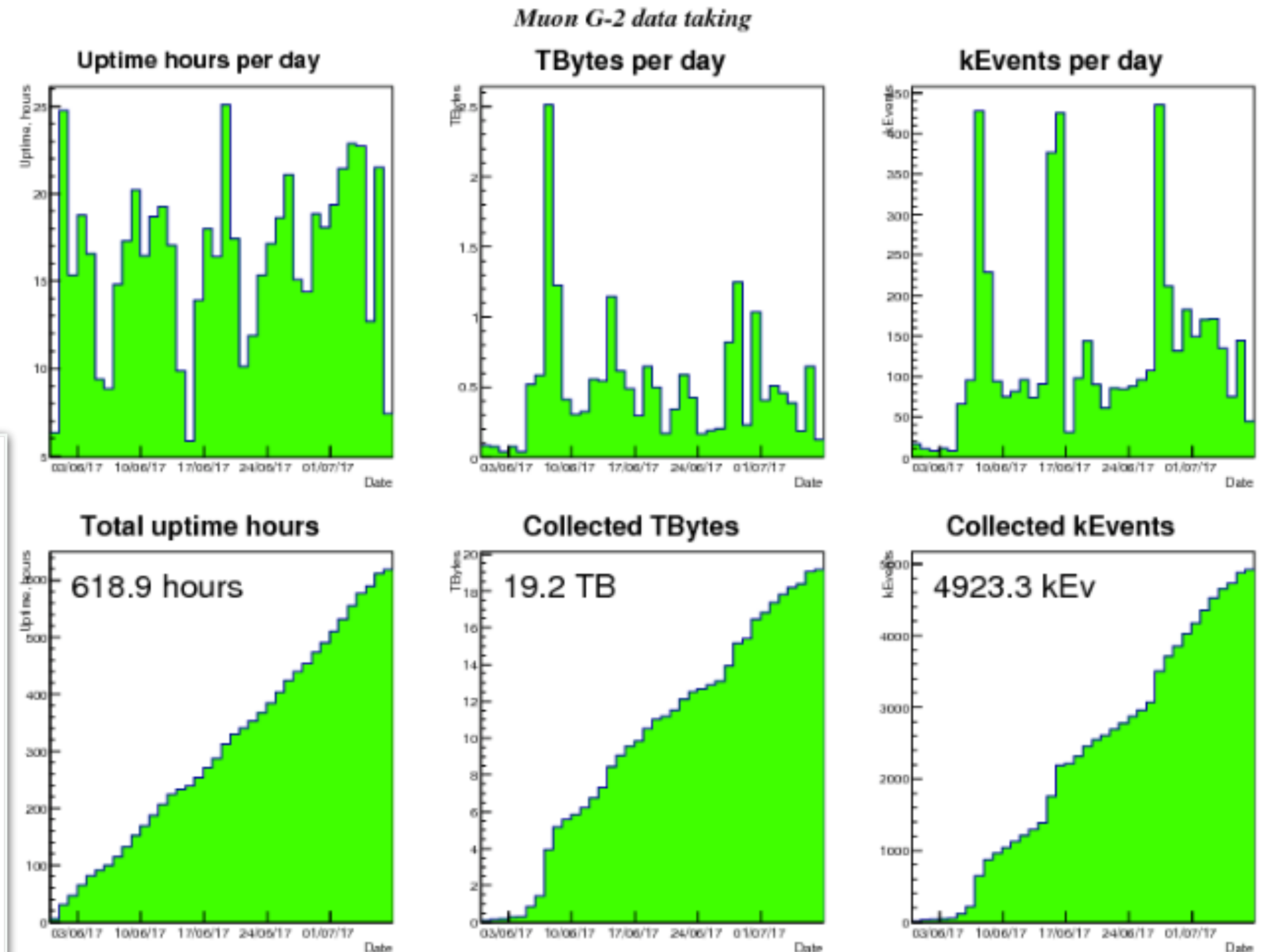
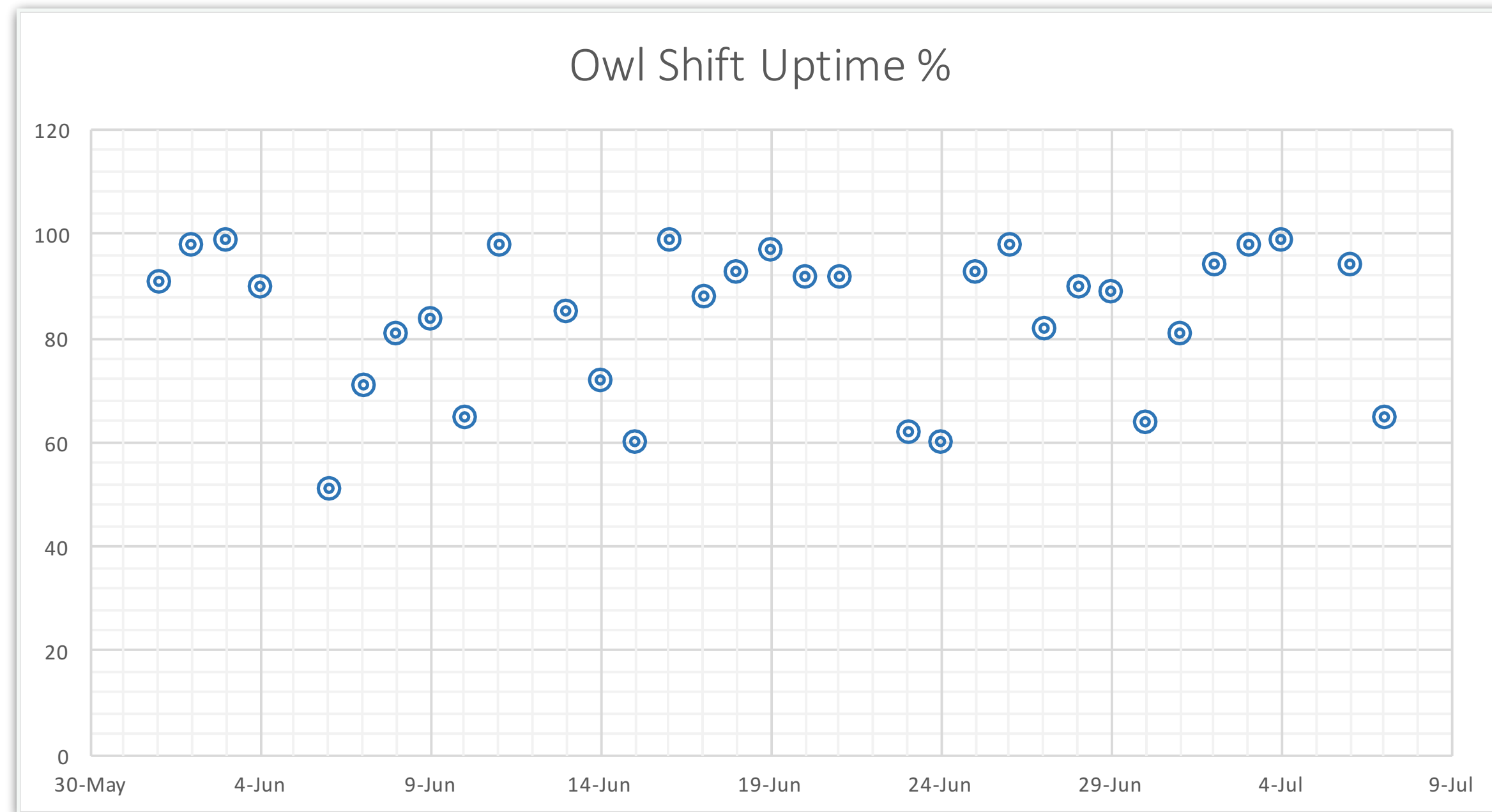
- Monitored health of DAQ systems using netdata for system monitoring, prometheus for short term data storage, and grafana to display data.



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

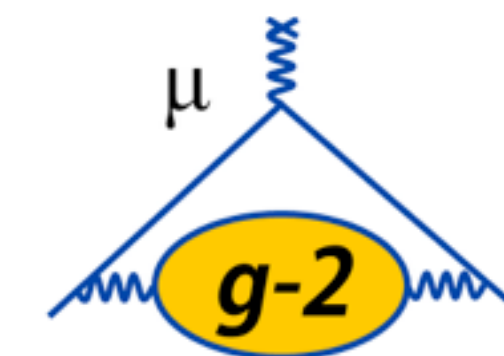


- MIDAS is working very well as the DAQ software for Muon g-2 at Fermilab.
- The DAQ hardware includes 17 frontend machines, 5 backend machines, 2 dedicated near line analysis machines, 3 computers for slow control, 3 servers, and 24 beagle bones running 67 MIDAS frontends.
- It takes an input data rate 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.