muon (g-2) Data Quality Monitor (DQM)

Aaron Fienberg

MIDAS Workshop

26 July 2017

(g-2) DQM one year ago: ROME

June 2016 SLAC test beam: prototype DQM using ROME

- test beam run with 1.5/24 calorimeters
- DQM often fell far behind the DAQ
- frequent crashes
- turning on DQM slowed down event builder
- no local ROME experts to help us
- separate analysis software very difficult to maintain
- need DQM that will scale to 24 calos + subsystems

After test beam run, made decision to build new DQM system addressing the above issues

Fermilab *art* framework and *midas-to-art*

art is an event-processing framework developed/maintained by Fermilab SCD

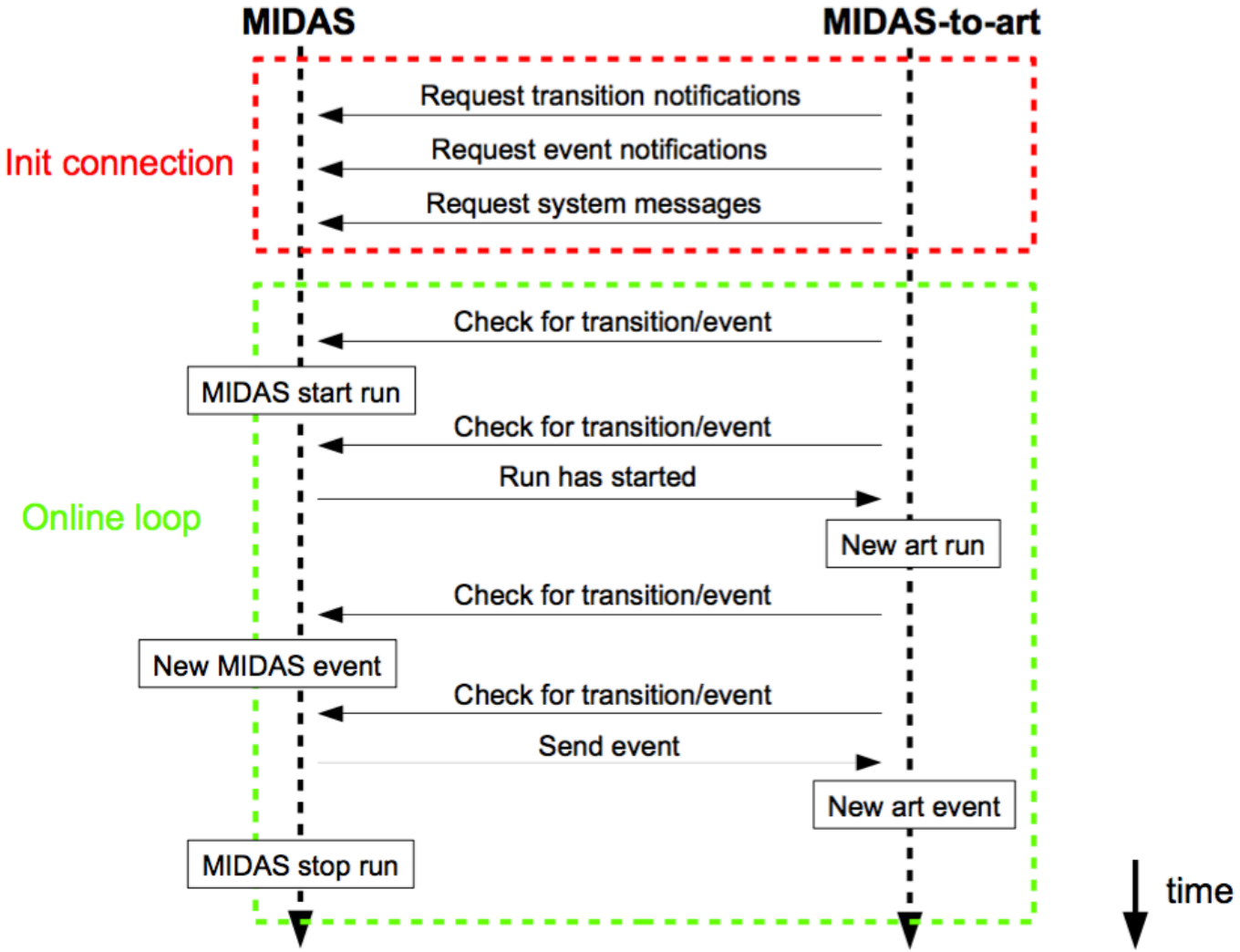
- used by (g-2), mu2e, NOvA, and others
- (g-2) unpacking/recon/analysis done in *art* modules
- *art* has its own file/data format
- *midas-to-art* plugin: reads MIDAS files into *art* jobs, translates MIDAS banks into *art* event data
- *midas-to-art* written originally for another (g-2) test beam, has been used successfully since
- strong preference in collaboration to develop *art*-based DQM so we can reuse offline modules

midas-to-art online extension added last summer

couple configuration parameters to change *midas-to-art* from offline-mode to online-mode

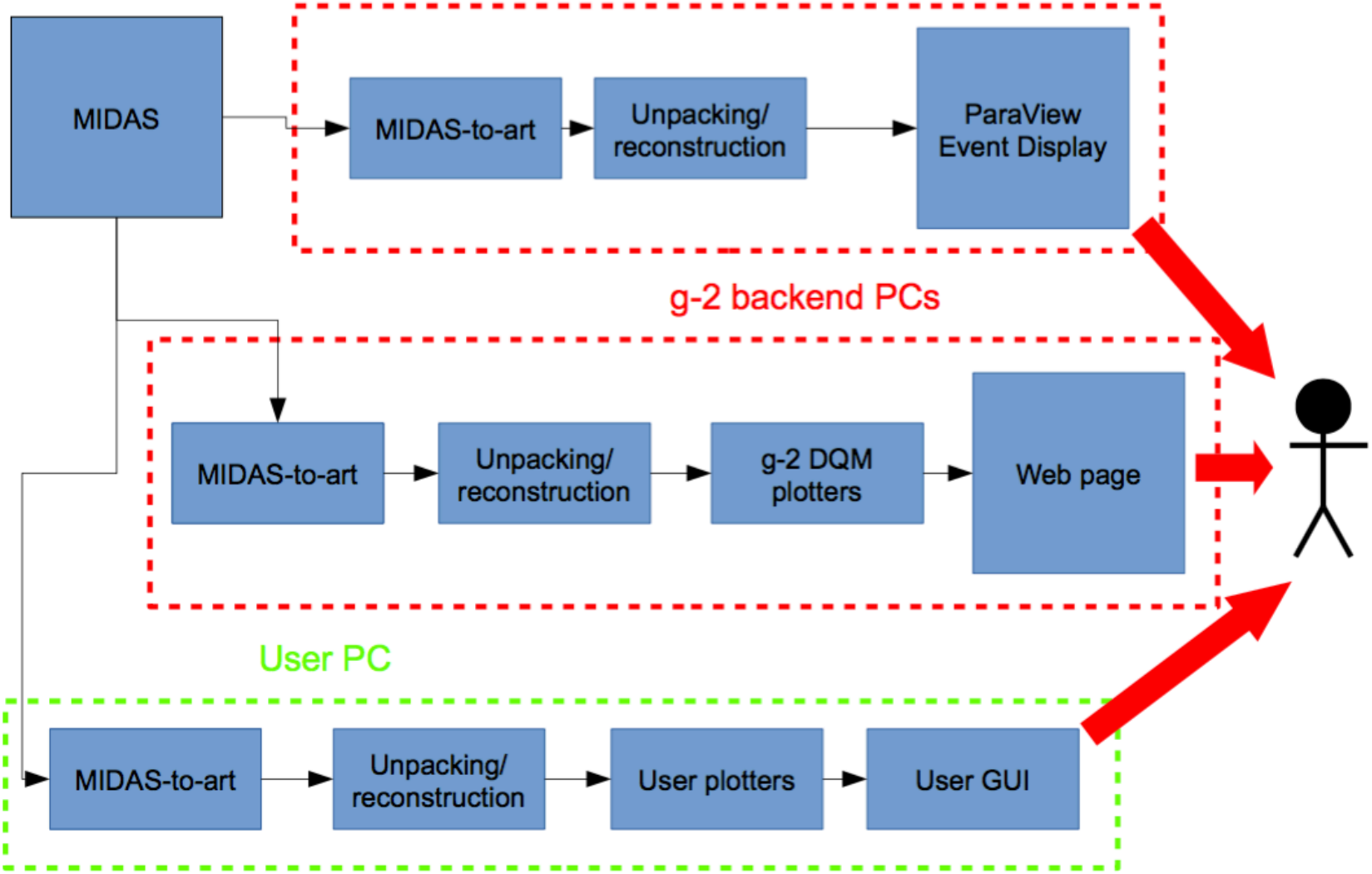
- connects to mserver like a remote analyzer
- uses GET_NONBLOCKING
- *art* events created in online-mode are identical to those created in offline-mode
- allows us to run our highly tested, optimized, and multithreaded reconstruction and analysis modules online without any modification at all
- *art* can be used as backend for number of different frontend DQM pages/event-displays, etc.

midas-to-art online



Fermilab

potential system



Fermilab

issues with *midas-to-art* online

- *midas-to-art* saves ODB XML string for each run,
  using *db_copy_xml()* as remote client proved so slow that we would miss first 10-20 events of a run (many seconds to execute command)
  - workaround: run remote ssh command to get ODB xml string

- *art* is very strict about run and event transitions, events coming in after end of run transition would cause art to throw an exception and exit
  - workaround: set very high end of run transition priority and ignore events that come in after end of run

- otherwise, *midas-to-art* online has performed very well and very reliably

we use ZeroMQ to get data out of the art job

- *art* is not particularly interactive, need to stream data out
- we are using the ZeroMQ messaging library for this
- bindings in many major languages

- simple *art* analyzer modules take unpacked/reconstructed data and publish them

zeromq.org

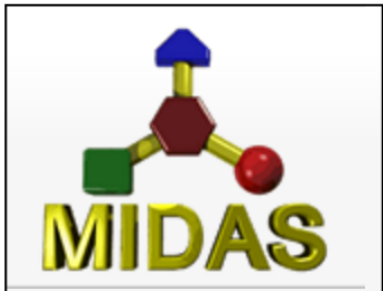
- data picked up by web GUIs, event displays, etc.

main DQM webserver built with node.js

- server-side javascript runtime environment
- event driven architecture
- asynchronous, non-blocking IO
- built for scalability and throughput
- numerous mature, open-source plugins with lots of community support

General Architecture

bidirectional communication



midas experiment/mserver

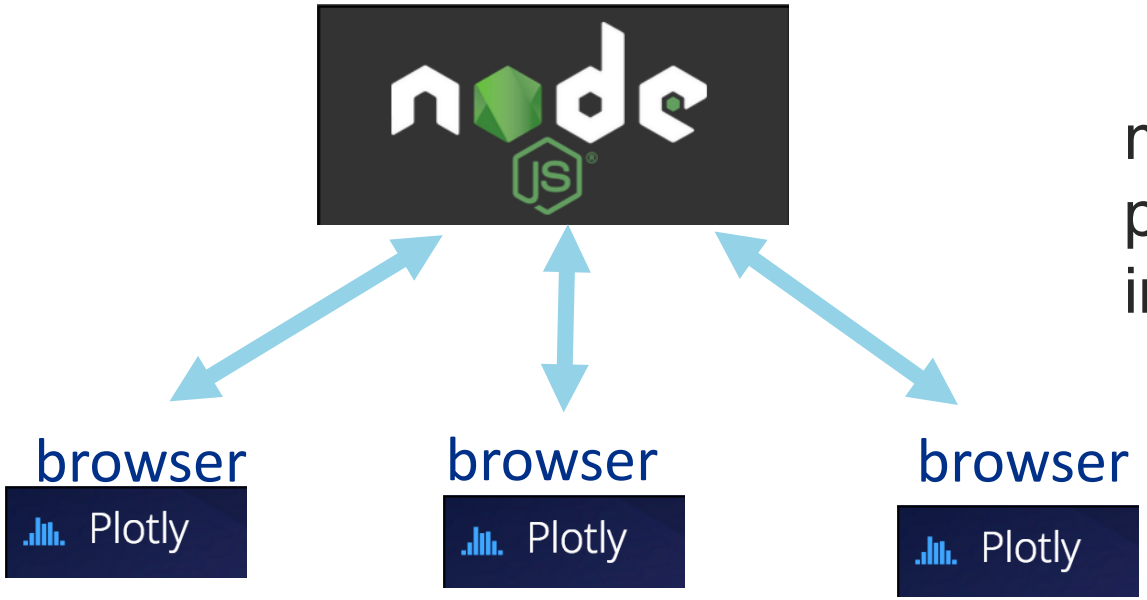
midas-to-art
unpackers
producers
analyzers

any module we've written can run online
(limited only by speed)

node.js webservers
data aggregation
client communication

connections to multiple clients

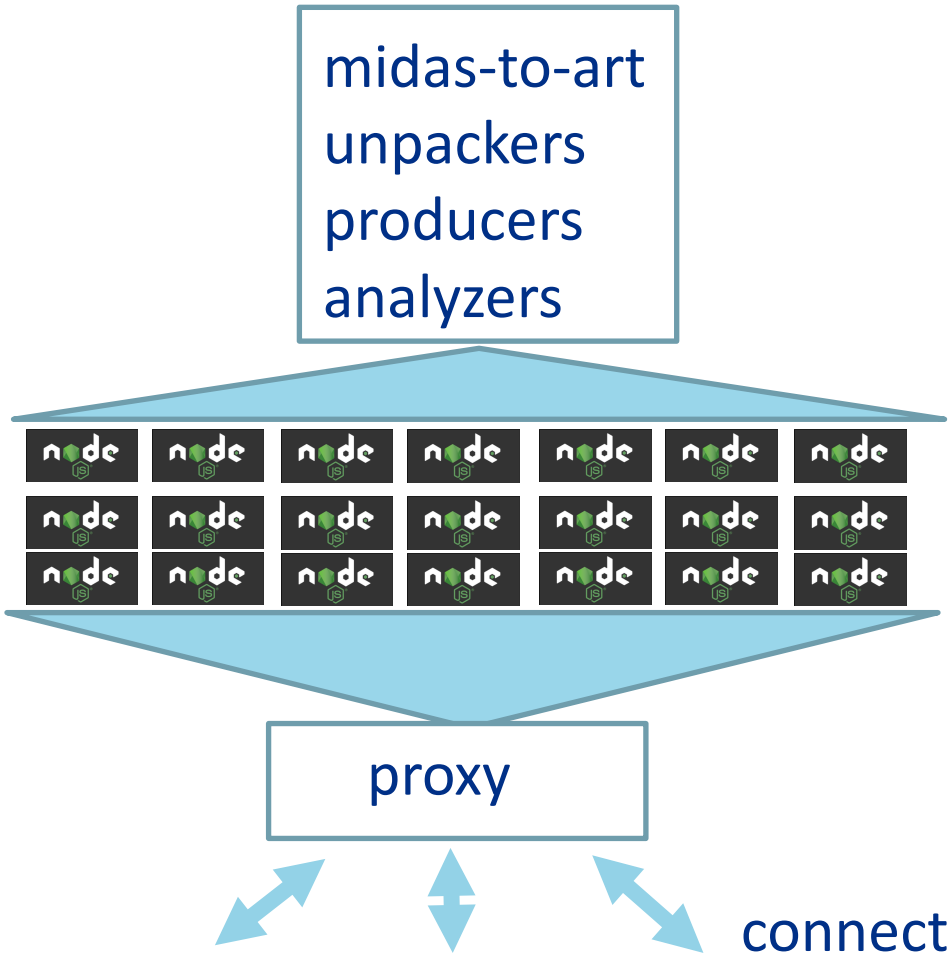
plot rendering is outsourced to clients



mainly using plotly.js for interactive plots

- server sends data through WebSockets on client request
- clients are insulated from both *art* job and MIDAS
- we use some additional js visualization tools (D3.js)

load distribution and modularity



- we use one server per calo + one per auxiliary detector system
- independently developed DQM apps can be run either independently or in concert
- proxy provides single point of entry, appears as one web page
- also run multiple *midas-to-art* instances

uses and tests so far

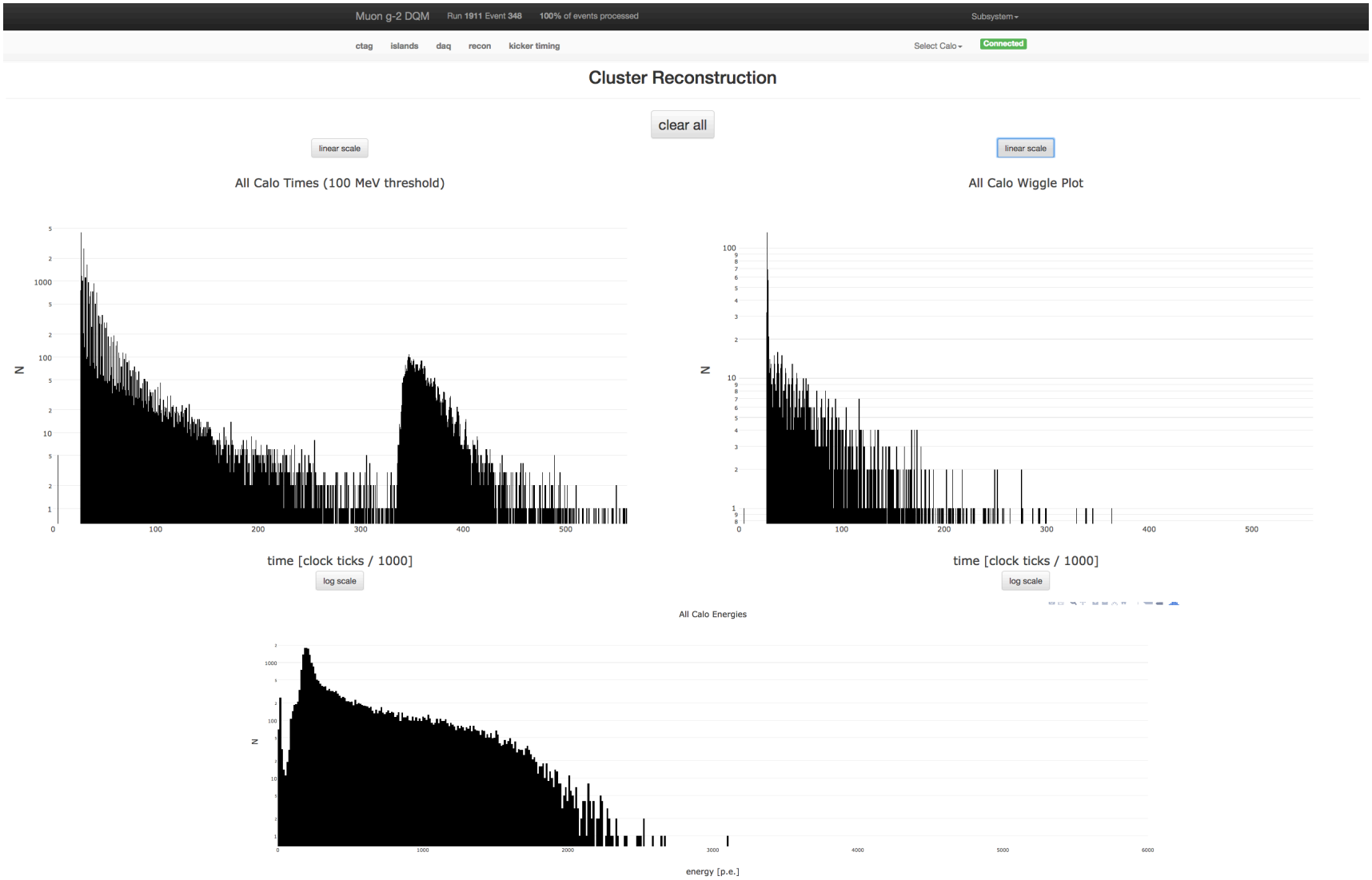
- initially tested on full-scale, full-rate 24 calo AMC13 simulator experiment, no apparent issues after a few days of running
- used in calorimeter, tracker, magnetic field test stands
- used for DAQ tests over the past year
- used during recent (g-2) commissioning run
- was able to serve numerous remote and local clients during the run (about 20 peak concurrent users)

example DQM pages

injected beam intensity and storage efficiency

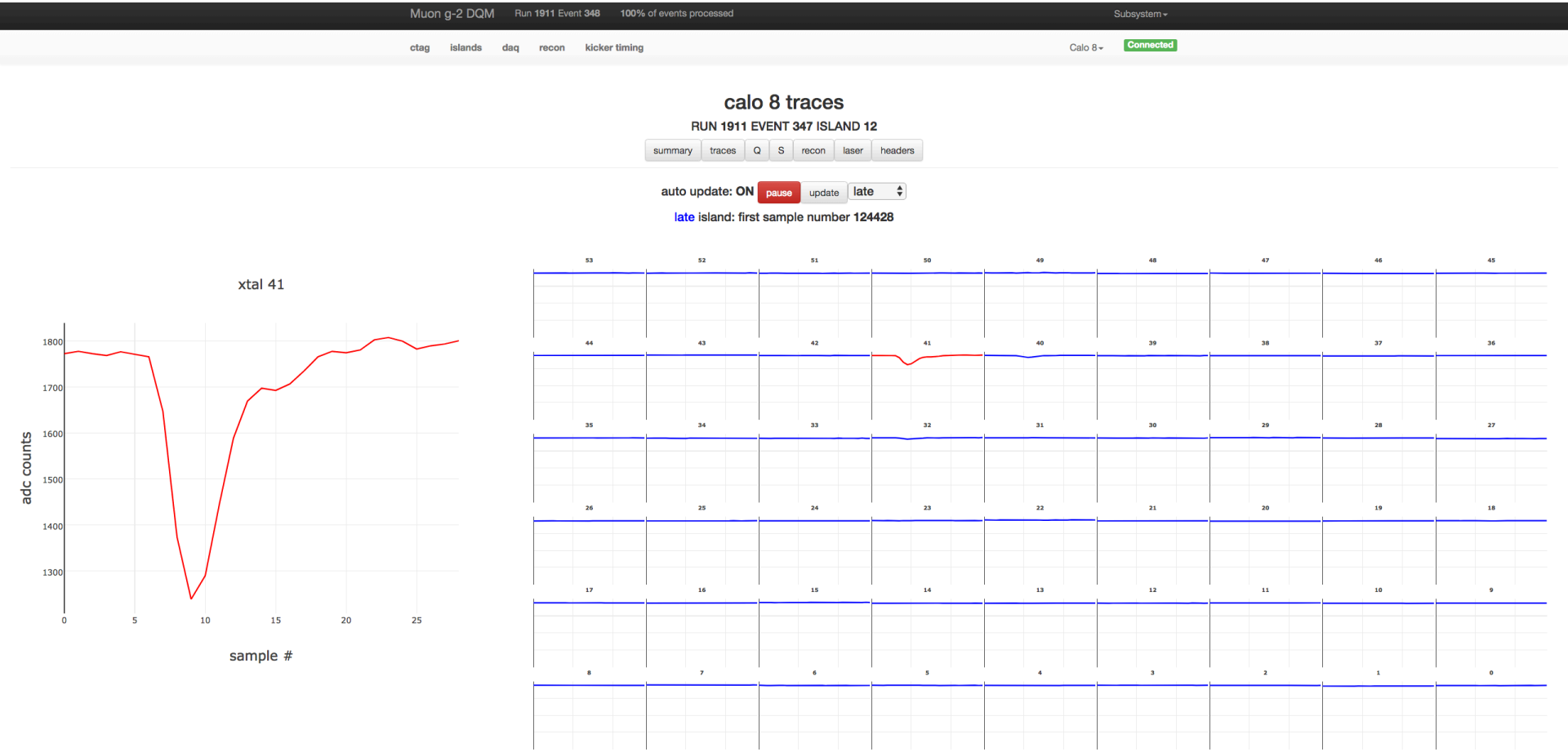
Fermilab

reconstructed calo hit times and energies



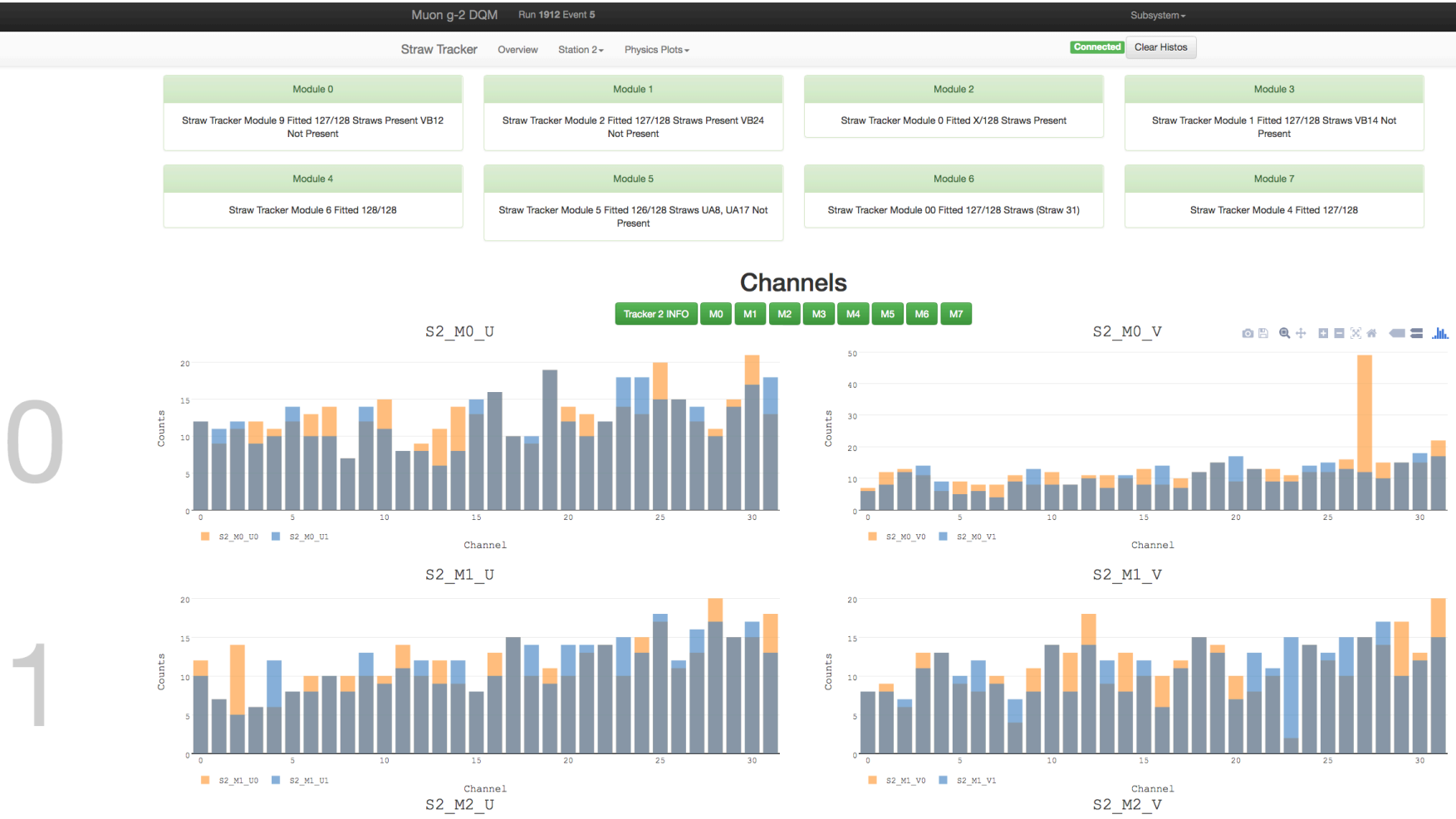
Fermilab

positron pulse in a calorimeter



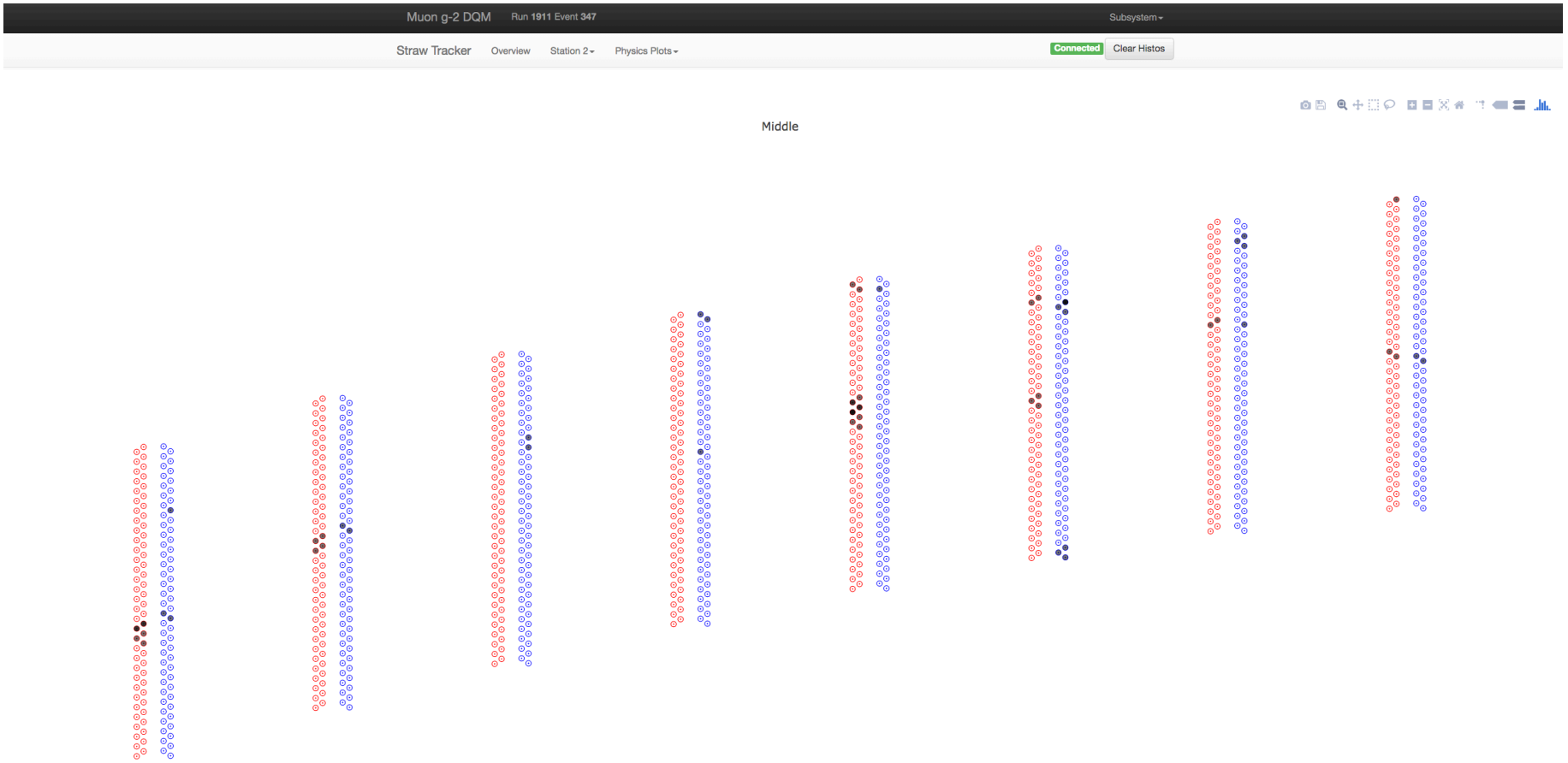
Fermilab

straw tracker TDC hits



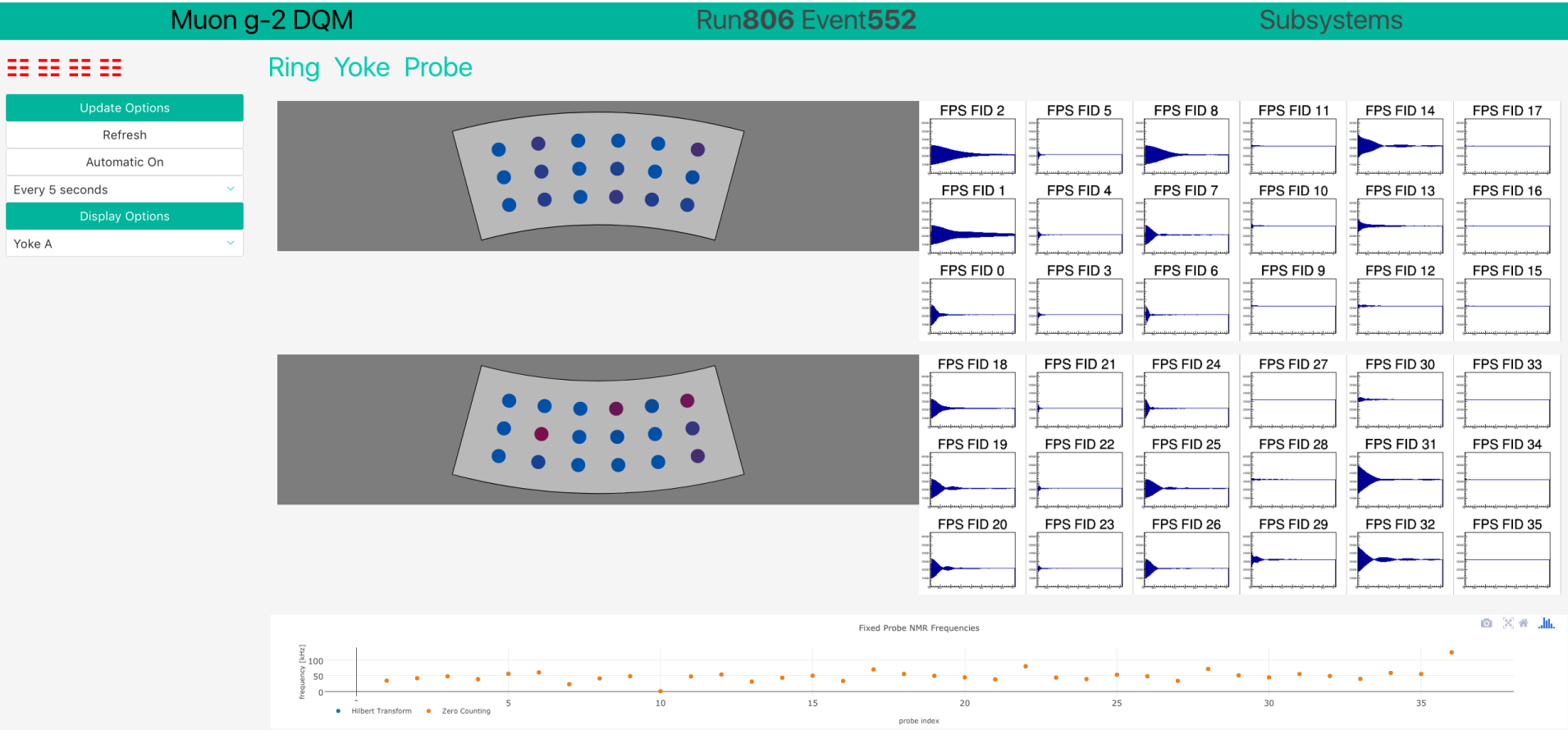
Fermilab

straw tracker occupancy

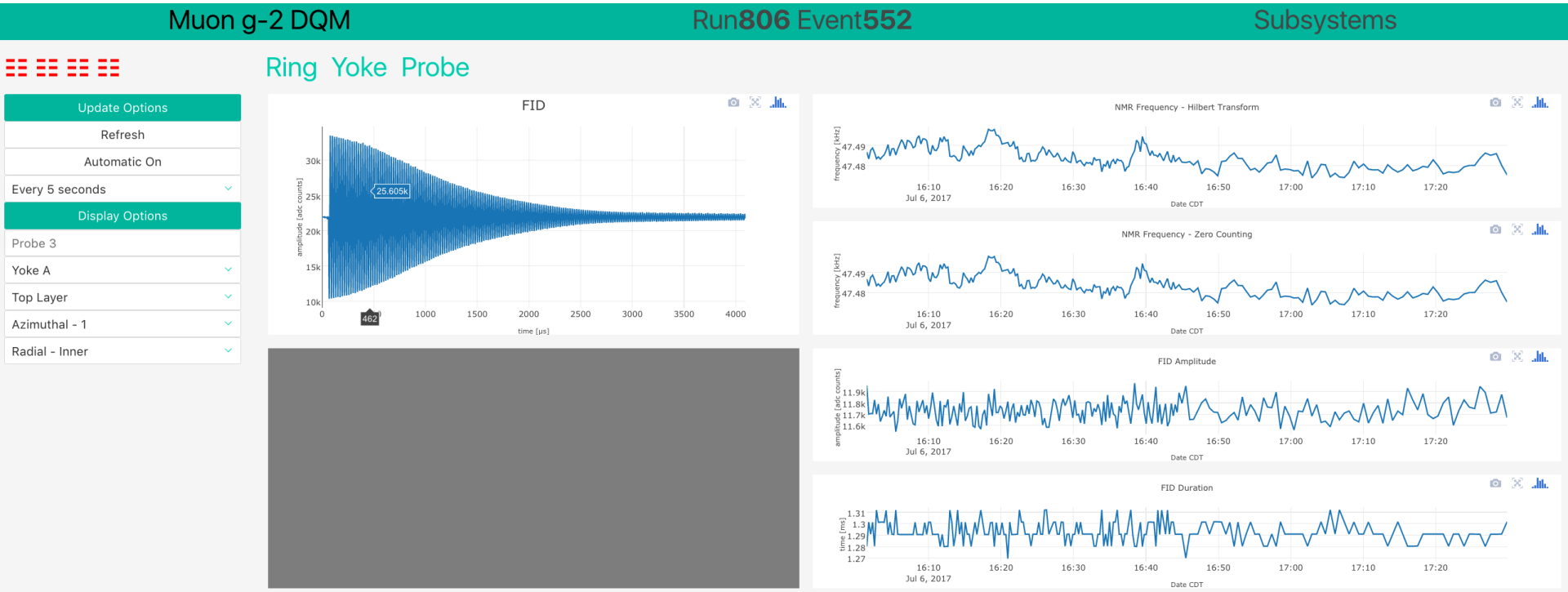


Fermilab

magnetic field NMR probe health and FID traces



single NMR probe FID and frequency histories



Fermilab

DAQ event builder monitor

Fermilab

reflections after the run

- DQM system received positive feedback from users, remote collaborators especially appreciated ease of access
- beam and storage diagnostics were invaluable in making optimal use of our short beam time
- system was stable throughout the run, and was easily adaptable as new plots/diagnostics were requested
- *art*-users seemed to have an easy time adding DQM pages for their systems, whereas subsystems that had yet to develop *art* offline code had a more difficult time
- weakest link is network IO between mserver and *midas-to-art*, our events can just get very large (max about 2 GB)
- average data rate is fine, instantaneous rate can cause lost events (perhaps should implement buffering in *midas-to-art*)

possible future directions

- 3-dimensional real-time event display with ParaView, reading data from the online *midas-to-art* job
- database integration allowing for easy generation of trend-plots based on analyzed parameters and for recording data necessary to quickly regenerate key DQM plots from old runs
- production reverse-proxy server with authentication so users can access without opening ssh tunnels
- Thank you!