DUNE DAQ DQM Update

Wes Ketchum

DUNE DAQ DQM Meeting

27 June 2024

Overview of Status

- Transfer script copying data from data volumes to common location
  - And monitoring to make sure that those locations used some maximum size
- Analysis scripts creating plot files from that data
  - TPC event display
  - WIB tests table
  - PDS plots
- Simple webpage for displaying most recent version of those plots
- *justintime* interactive display now looking at that shared area

Transfer script

- New script in np04daq user area: copyFileDQM.sh

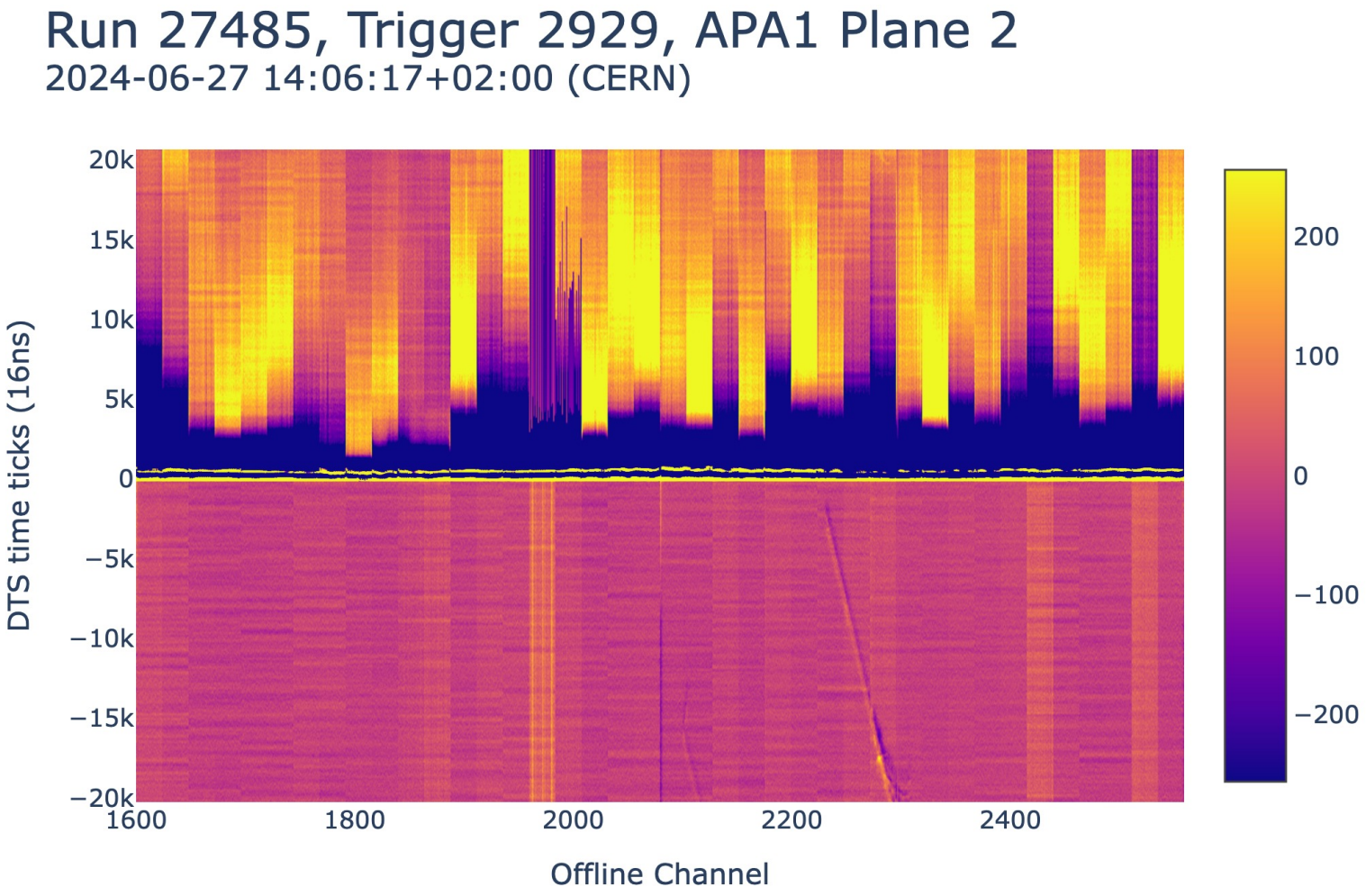
```
Usage: /nfs/home/np04daq/.cron/copyFileDQM.sh
<remote_host> <remote_directory>
<destination_directory> <max_data_volume_in_mb>
```

  - Find most recent file on remote host in chosen directory
    - No protection if multiple runs going at same time
  - Copy it a destination directory
    - /nfs/rscratch/np04daq/DQM_DATA/NP04
  - Check destination directory's size, and remove oldest files until within some max data volume
    - Uses a "cleanDQMFiles.sh" script, that is also used to clean event display plots
- Running in crontab as np04daq on np04-srv-015
  - Looking at np04-srv-005:/ssddata2, and np04-srv-004:/data0 every two minutes

Event Display script

- See code here
- Required arguments:
  - Specify an input file or input directory
    - If input directory, grabs latest file in that directory
  - Specify an output directory for location of plots
- Optional arguments:
  - `--nskip` to pick which trigger record of file (defaults to 0, first)
  - `--nrecords` to pick how many records to make plots for (defaults to 1, multiple are useful for laser runs)
  - `--component` and `--plane` to pick which APAs/planes to plot (defaults to all)
  - `--imgtype` to pick what kind of image file (defaults to svg, good for web display)
- Plots written with a standardized parse-able name
  - E.g. “EventDisplay_run27485_trigger2929_seq0_APA1_plane2.svg”
- Simple script running this in a loop with at 2 minute sleep in a tmux session on np04-srv-004

Event display example



Fermilab DUNE

WIB Tests Table

- See code here
- Same required arguments as before: input (files or dir) and output (dir)
- Runs through specified number of records and then runs all ”WIB tests” (checks of basic header information, timestamp alignment, number of frames, etc.)
- Prints a table showing whether each test passed or failed to an image file
  - Again, saved with a standardized and parse-able name …
  - Also has some terminal output…
- Also currently run via a simple script executing this in a loop with a 2 minute sleep in a tmux session on np04-srv-004

WIBTests Table Example

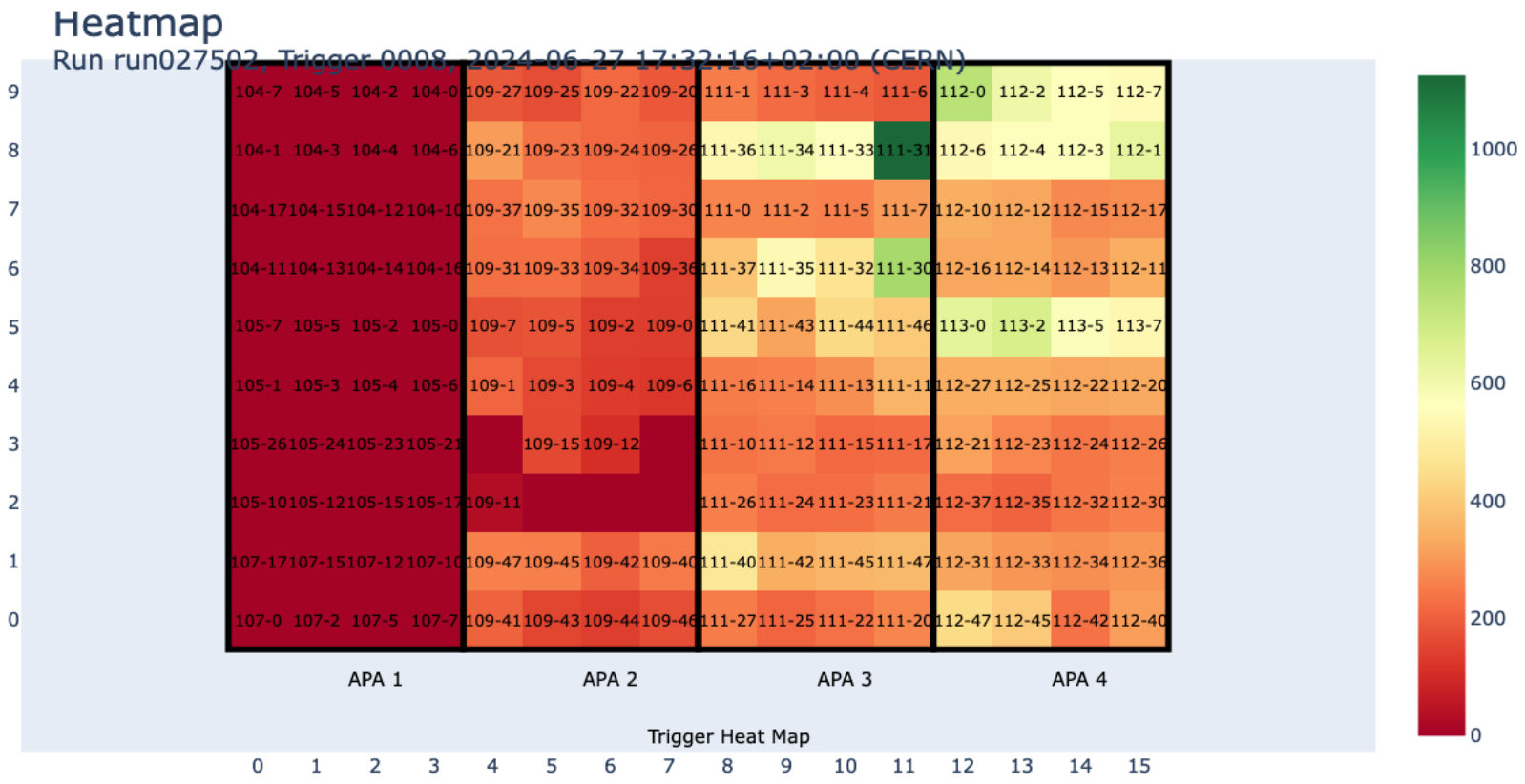
Latest Run,Trigger = (27502,315)
35 total records processed --- 2024-06-27 17:32:16+02:00 (CERN)

Test Name	Result
CheckRequestTimes_WIBEth	OK
CheckTimestampsAligned_HD_TPC	OK
CheckWIBEth_FEMB_Sync_HD_TPC	OK
CheckWIBEth_Link_Valid_HD_TPC	OK
CheckWIBEth_LOL_HD_TPC	OK
CheckWIBEth_CD_HD_TPC	OK
CheckWIBEth_Context_HD_TPC	OK
CheckWIBEth_Calibration_HD_TPC	OK
CheckWIBEth_Pulser_HD_TPC	OK
CheckWIBEth_CRC_Err_HD_TPC	OK
CheckWIBEth_COLDDATA_Timestamps_Aligned_HD_TPC	OK
CheckWIBEth_COLDDATA_Timestamp_1_Diff_HD_TPC	OK
CheckWIBEth_COLDDATA_Timestamp_0_Diff_HD_TPC	OK
CheckTimestampDiffs_WIBEth_HD_TPC	OK
CheckNFrames_WIBEth	OK
CheckAllExpectedFragmentsTest	OK

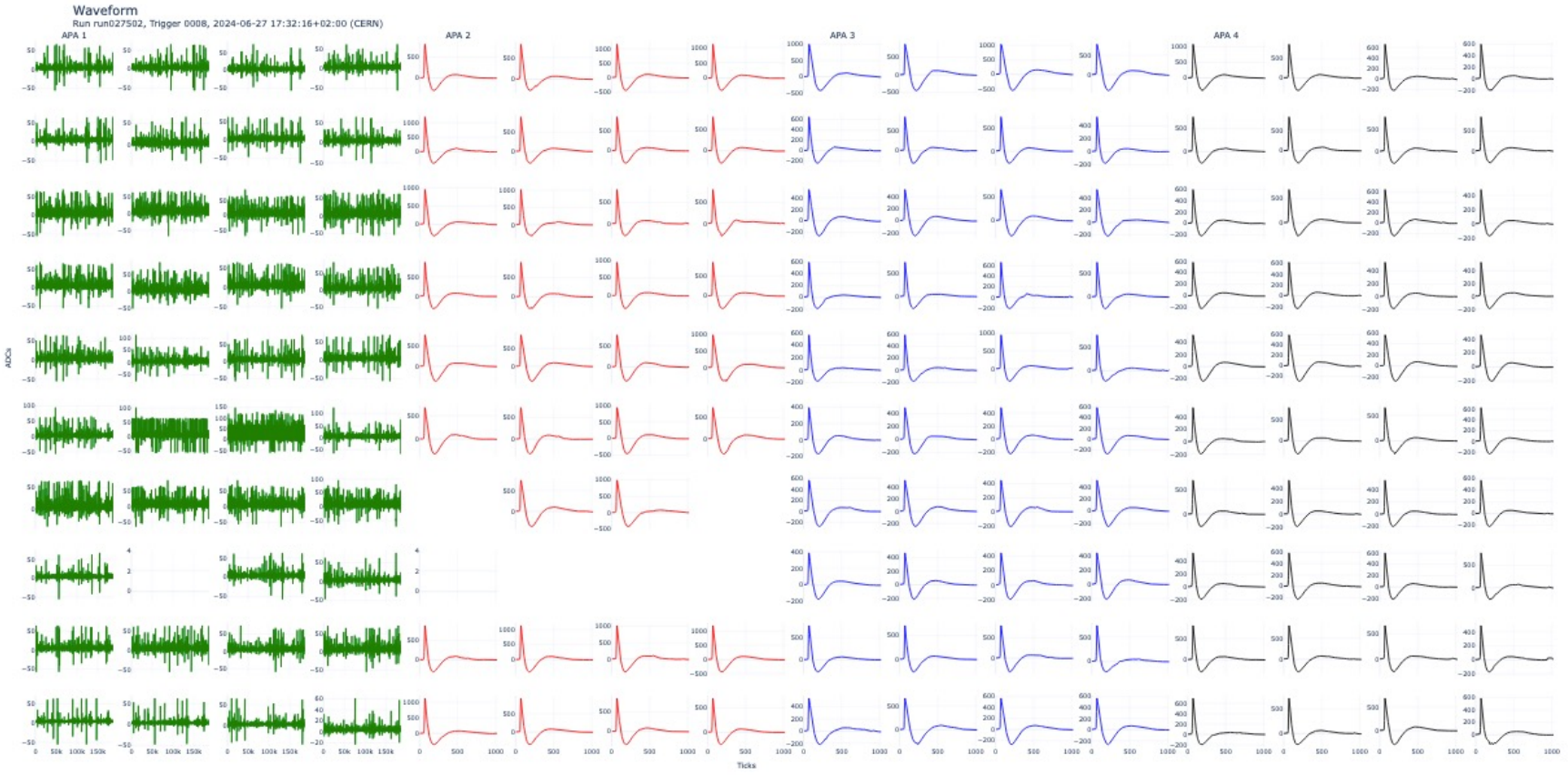
PDS DQM analysis

- Work from PDS team (largely Renan de Aguiar) to do analysis of PDS data over multiple trigger records
  - See here for code
- Defaults to looking at four most recent files in a directory, and processing through all trigger records in them
- Creates a few different plots and saves them to files (with parse-able names…)
  - Baseline and RMS checks for each PDS channel
  - A heatmap of number of counts for each PDS channel
  - Sample / averaged waveforms for each PDS channel
- Python script manages its own loop / sleep time
- Runs in a dedicated tmux session on np04-srv-004

PDS Plot examples



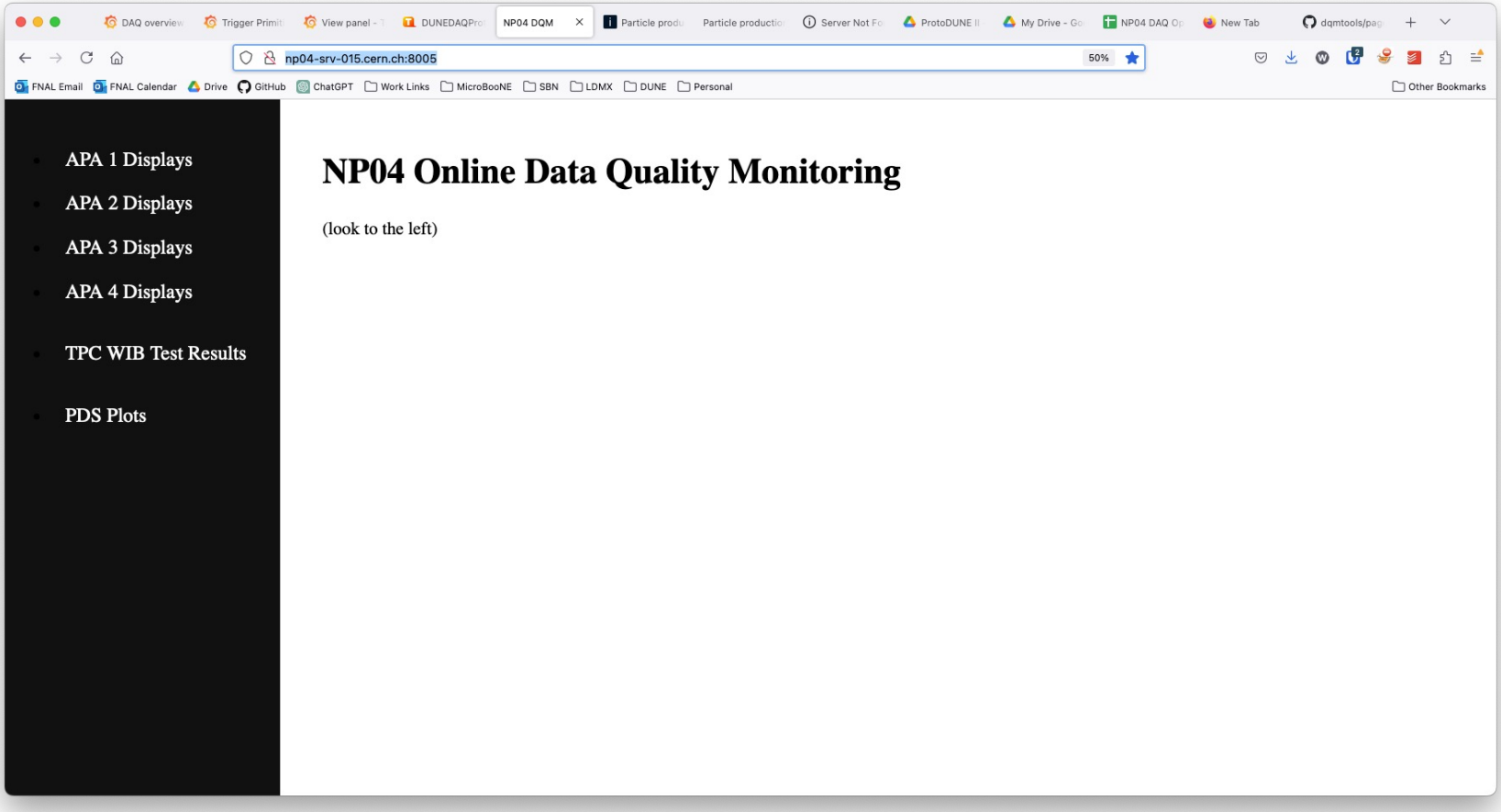
PDS Plot examples



Web display

- In order to have a simplified display showing these, have a very simple flask-based web server running
  - Python code and templates here
- Features …
  - Functions implemented to look into directories, parse file names, and find the ‘most recent’ based on run and trigger record number
    - Expects to be pointed to top-level directory, with hard-coded subdirectories containing the desired plots
  - Templates for showing pages
    - For event display, have addresses per APA and plane available to make loading times faster
  - Auto-refresh
    - I think? Maybe not optimally …
  - A navigation bar that isn’t the ugliest thing ever
- Running in a tmux session on np04-srv-015
  - http://np04-srv-015.cern.ch:8005/

A very impressive landing page



Fermilab DUNE

Things to do …

- Minor improvements in plots
  - E.g. add what trigger type we see on event displays
- Add new plots / images ???
- Cleaning up deployment for operations
  - Some probing questions:
    - Do we want to use rscratch for hosting the data? If not, where?
    - Relatedly: what server do we want to run on?
    - Do we want to run out of tmux sessions still? Can we / should we make these ‘microservices’ that run?
      - And can run ~anywhere?
      - If so, how do we deploy/redeploy/etc. as we expect changes to occur?
  - We should try to ~finalize this for the restart of beam (so, end of next week)
- Extending elsewhere
  - Even more so than justintime, this is largely assuming ProtoDUNE-HD for now
  - Do we need to push to make it more extendible ~now?