From NSCLDAQ to FRIBDAQ

Giordano Cerizza

MICHIGAN STATE UNIVERSITY

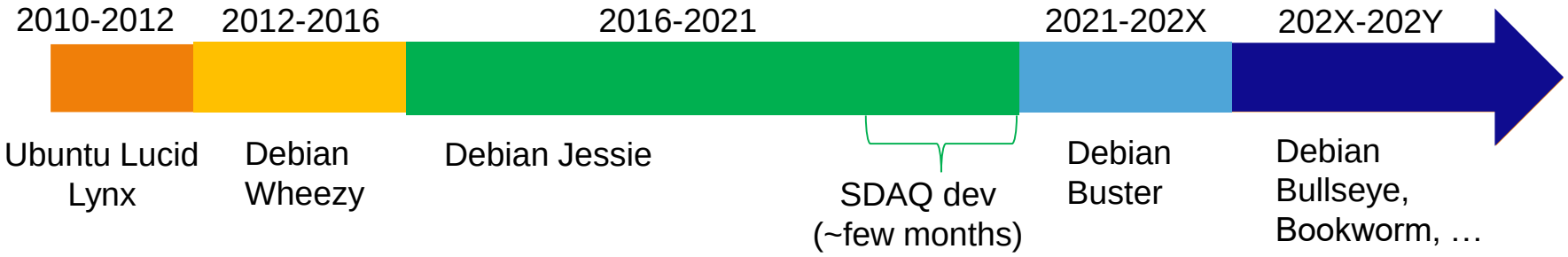
U.S. DEPARTMENT OF ENERGY | Office of Science

This material is based upon work supported by the U.S. Department of Energy Office of Science under Cooperative Agreement DE-SC0000661, the State of Michigan and Michigan State University. Michigan State University designs and establishes FRIB as a DOE Office of Science National User Facility in support of the mission of the Office of Nuclear Physics.

Why giving this talk?

- Establish a connection between developers and users where the latter could steer the work of the first ones
  - And keep the communication alive after this talk
- Showcase of what is already available
- Directions for the future
- Open discussion

NSCL/FRIB OS Distribution Timeline



- IT deploys a new OS at the end of a life cycle during shutdowns/PAC cycles
- An OS upgrade even in the same major version requires drivers to be compiled
- Critical security issues have to be released promptly but it would affect drivers

➔ How do we handle and manage IT and experimental needs?

Containers are a solution to the problem of how to get software to run reliably when moved from one computing environment to another → security VS stability problem

A little detour

- Filesystem image of Linux

/boot - kernel and its startup stuff	/sbin - system administration software	/etc - configuration files that govern how the system works

/bin, /usr/bin – programs /lib, /usr/lib, /usr/include – library
/usr/opt – NSCLDAQ, SpecTcl etc…

This is 90% of what people call a Linux version

"userland"

/user – home dirs., /mnt/events, /mnt/daqproj … data, shared code

Containers

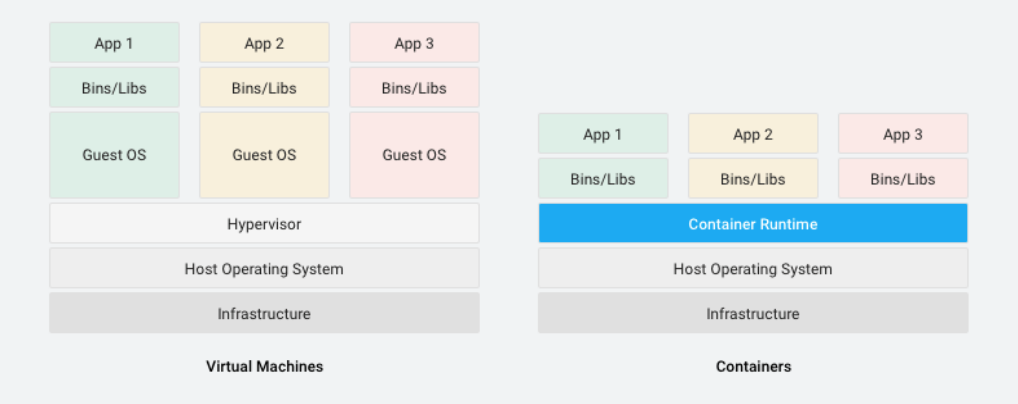
- A container consists of apps, libraries, binaries, and config files (“userland”) ALL bundled in one package.
- Container vs Virtual Machines (VM)

Virtual machine

- It includes an entire operating system.
- Hardware overhead to run multiple virtual machines at once (hypervisor + N OS).
- Size several GB, limited portability.
- Minutes of booting time.
- One and done.

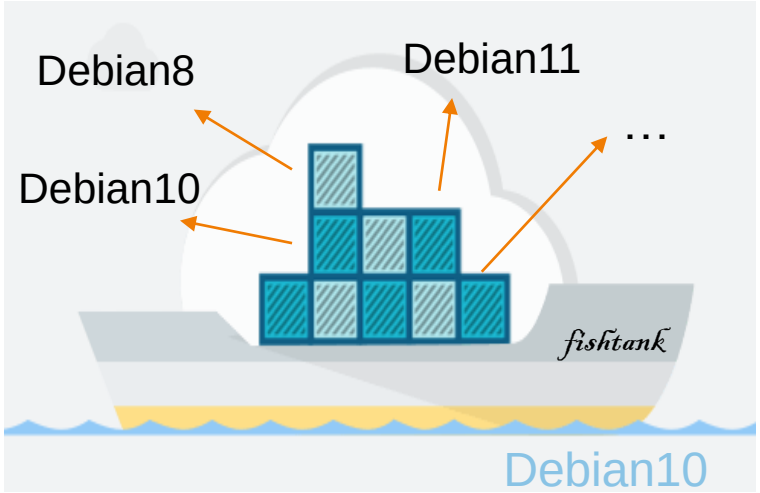
Container

- It includes only what is needed.
- One machine can run multiple containerized applications.
- Essentially no overhead.
- Instant booting
- Modularity (micro-service approach).



Why containers for FRIB?

- Stable environment
  - Predictability (i.e. analysis code just runs and runs…)
  - Productivity (i.e. no last minute issues)
- Run anywhere
  - External users can develop software and test hardware in the FRIBDAQ environment **INDEPENDENTLY** from the OS of their machine
- Isolation
  - CPU, memory, storage, and network resources STILL accessible but SEPARATED from other application (i.e. security)



Linux, MacOS, Windows, Virtual Machines, Cloud, Data Center, HPCC, …

Debian8 - Jessie

SpecTcl-3.5
nscldaq11/ReadoutShell
…

Debian10 - Buster

SpecTcl-5.13
nscldaq12/ReadoutShell
…

A working example

- We provide examples in fishtank:/usr/opt/buster.sh

```
#!/bin/bash
BINDINGS="  --bind /usr/opt/opt-buster:/usr/opt,/scratch,/mnt/misc/sw/indep \
            --bind /mnt/misc/sw/x86_64/all \
            --bind /usr/lib/x86_64-linux-gnu/modulecmd.tcl \
            --bind /usr/opt:/non-container"
IMAGE=/usr/opt/nscl-buster.img
export TZ=America/New_York

#  Exactly how we do things depends on whether or not
#  We've got apptainer or singularity:

if command -v apptainer &>/dev/null
then
    export APPTAINER_SHELL=/bin/bash
    apptainer shell $BINDINGS $IMAGE
else
    export SINGULARITY_SHELL=/bin/bash
    singularity shell $BINDINGS $IMAGE
fi
```

bindpoint-list is a list of directories you want to make visible inside of your container
(--bind /native_dir:/container_dir)

For DAQ systems, one needs to store the binding points because data taking requires that processes be run over ssh pipes in local or remote systems, i.e.
echo /usr/opt/opt-buster:/usr/opt > ~/.singularity_bindpoints

https://docs.nscl.msu.edu/daq/newsite/container-instructions.pdf

FRIB(NSCL)DAQ

- General performance improvement driven by e17011
  - Infrastructure improvements
  - Increase data rates
  - Online data handling performance (see later)
  - Near-line/offline data handling performance
  - Online/offline data volume reduction

- Manager subsystem
  - Software runnable by a single account but controlled by users logged into their own account

NSCLDAQ12

https://docs.nscl.msu.edu/daq/newsite/nscldaq-12.0pre/index.html

FRIB(NSCL)DAQ

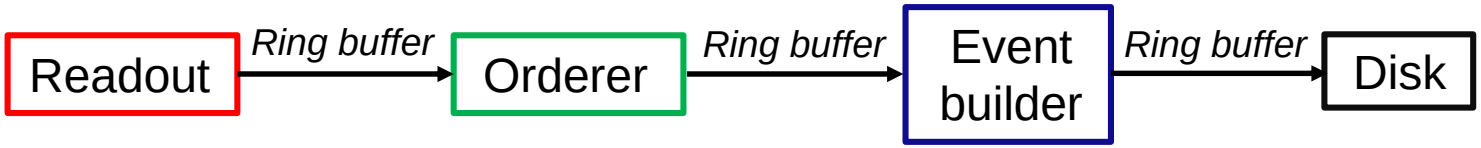
- General performance improvement driven by e17011
- Motivation & goals:
  - Sustain expected 3kHz trigger rates with XIA Pixie16 digitizers
  - Expected (aggregate) data rate 200MB/s sustained
  - Fit traces from segment CeBr3 detectors for single and double pulse shapes with analytic approximation
  - Being able to analyze nearline data at 100x incoming rate to support decision making

1) Infrastructure improvements for running experiments
- 4 new DAQ network systems: daqcompute001-004
- 4 new Office network systems: expanalysis001-004
- “Community” batch systems (upcoming)

FRIB(NSCL)DAQ

2) Increase data rates

- Separation of readout from sorting (main bottleneck due to SBC)
- Event builder optimizations
  - » Zero-copy methods (Ethernet-speed data transfer)
  - » Pipeline threading
  - » Optimized merging algorithms – dynamic switch as number of queues with remain fragments decrease
  - » Improved fragment selection via both timestamping and fragment demarcation



Outcomes:

1) Bottleneck is now the input stages of event orderer (i.e. readout)
   i.e. close to theoretical 109MB/s/crate (XIA), 60MB/s/crate (VME)
2) Ordering/sorting algorithm operates now well above 200 MB/s
3) ringFragmentSource/Orderer interface limits to about 200MB/s total rates – needs organic changes to application protocol feeding the Orderer for improvement

FRIB(NSCL)DAQ

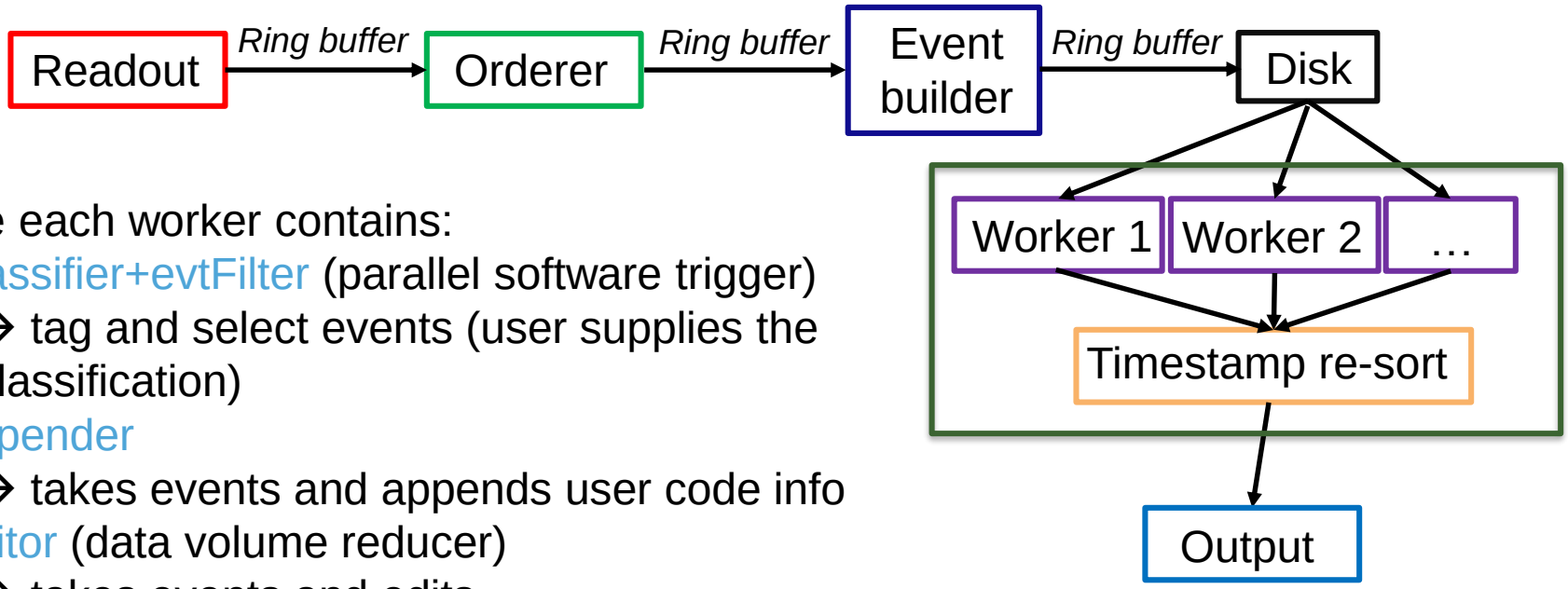
- Motivation & goals:
  - Sustain expected 3kHz trigger rates with XIA Pixie digitizers ✔
  - Expected data rate 200MB/s sustained ✔
  - Fit traces from segment CeBr3 detectors for single and double pulse shapes with analytic approximation
  - Being able to analyze nearline data at 100x incoming rate to support decision making

- Figuring out if the PIN detector has a heavy ion (and discard)
- Fitting traces to the magic shape function for one or two pulses
- Determining if the traces show one or two or more pulses
- Correlation implantation events with decay events

Parallel processing frameworks

FRIB(NSCL)DAQ

- Parallel processing analysis framework:
  - Based on several communication patterns – fan-out, fan-in, pipeline
  - Ability to run the code in threaded or MPI mode
  - Build frameworks for common operations that allow users to plug in sequential code chunks to do the work (i.e. load user code built into shared objects)
  - Ability to edit events

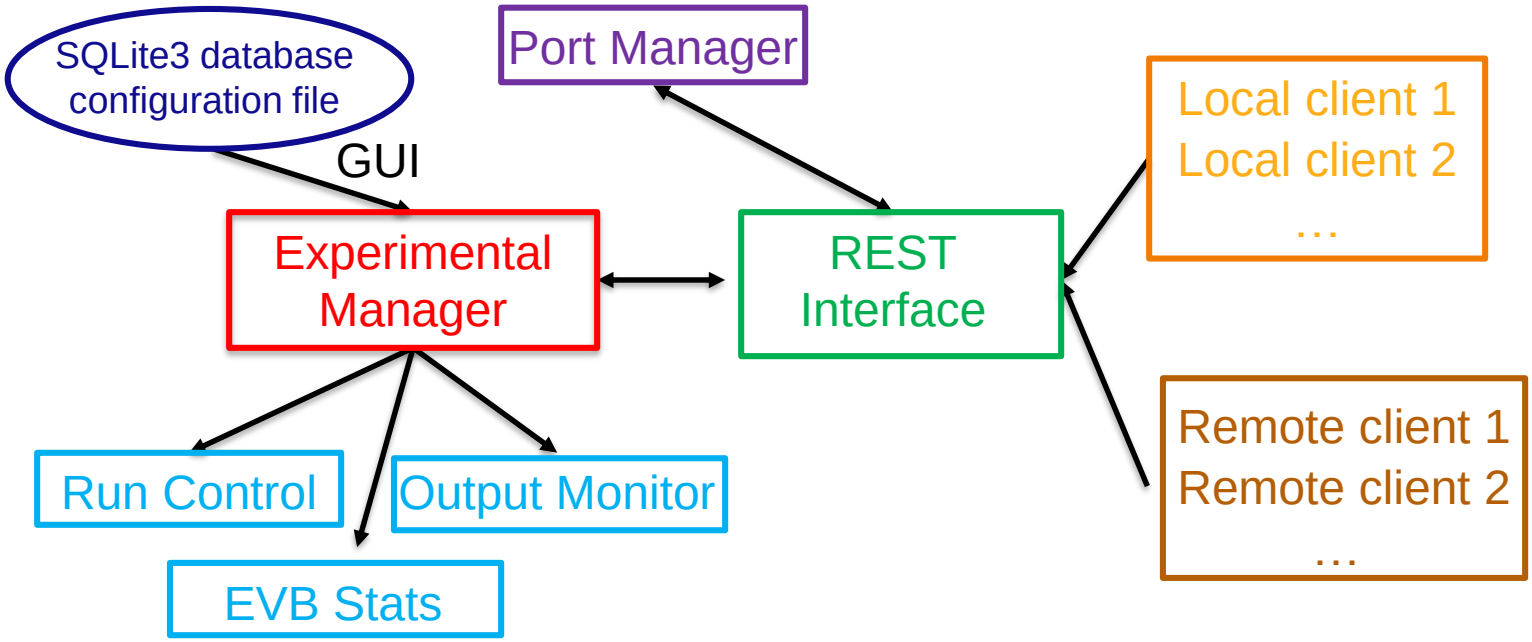


where each worker contains:

- Classifier+evtFilter (parallel software trigger)
  → tag and select events (user supplies the classification)
- Appender
  → takes events and appends user code info
- Editor (data volume reducer)
  → takes events and edits

FRIB(NSCL)DAQ

- Manager subsystem
  - The manager uses a configuration database, stored in an SQLite3 database file to drive its actions. It is a persistent server.
  - It provides an advertised REST interface that allows local or remote clients to interact with it at run time.
  - It provides an advertised port which allows clients to monitor the output and error streams of programs run by the manager.

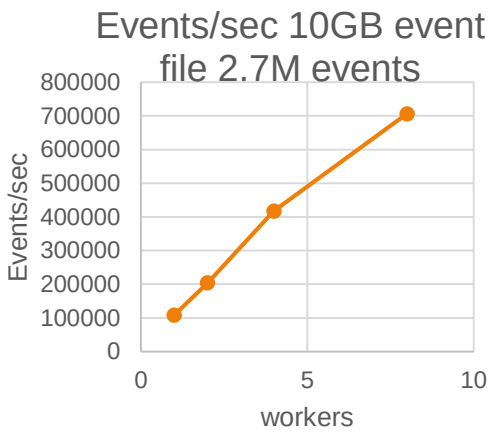


https://docs.nscl.msu.edu/daq/newsite/nscldaq-12.0pre/c748.html

SpecTcl advances

- Dynamic pipeline management
  - You can create and register event processors that are not in the analysis pipeline (see https://docs.nscl.msu.edu/daq/newsite/spectcl-5.0/pgmref/r23554.html)
- Online data handling performance
  - Threaded parallel – requires user modification of their analysis code
  - Batch Mode – serial but unpacking gets 3x performance compared to serial interactive SpecTcl (N.B. doesn't have user interface https://docs.nscl.msu.edu/daq/newsite/specbatch/c37.html)
  - MPI Mode – variation of Batch using an MPI aware TCL interpreter
    - » Each process is a complete batch SpecTcl instance
    - » Rank0 process feeds configuration and events to other instances and collect spectra
    - » Good linear scaling with # of workers
    - » Optimal for local computing cluster

    (see https://docs.nscl.msu.edu/daq/newsite/specbatch/c549.html)



SpecTcl advances

- ROOT support
  - All spectra are now actually TH* ROOT histograms
  - The Tcl Event loop is shared with the ROOT Event loop
  - New SpecTcl commands available:
    - » **rootexec** – runs a ROOT macro (i.e. instantiate a TBrowser to look at spectra and operate on them)
    - » **roottree** – create root trees from your unpacked parameters
  - Docs available here http://docs.nscl.msu.edu/daq/newsite/spectcl-5.0/cmdref/index.html

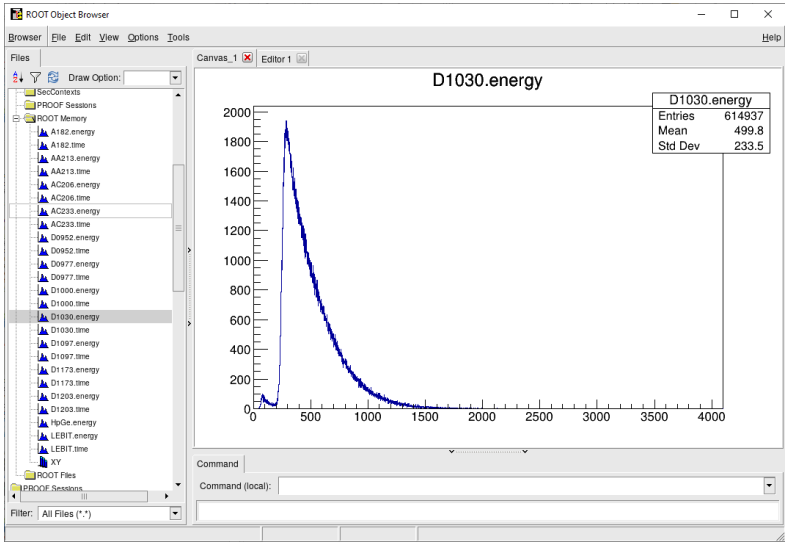
By adding those two lines to SpecTclRC.tcl

```
load $SpecTclHome/lib/libRootInterface.so
package require rootinterface
```

And creating a simple script like browser.C

```
void Browser() {
    new TBrowser;
}
```

By loading SpecTcl and launching
>rootexec Browser.C



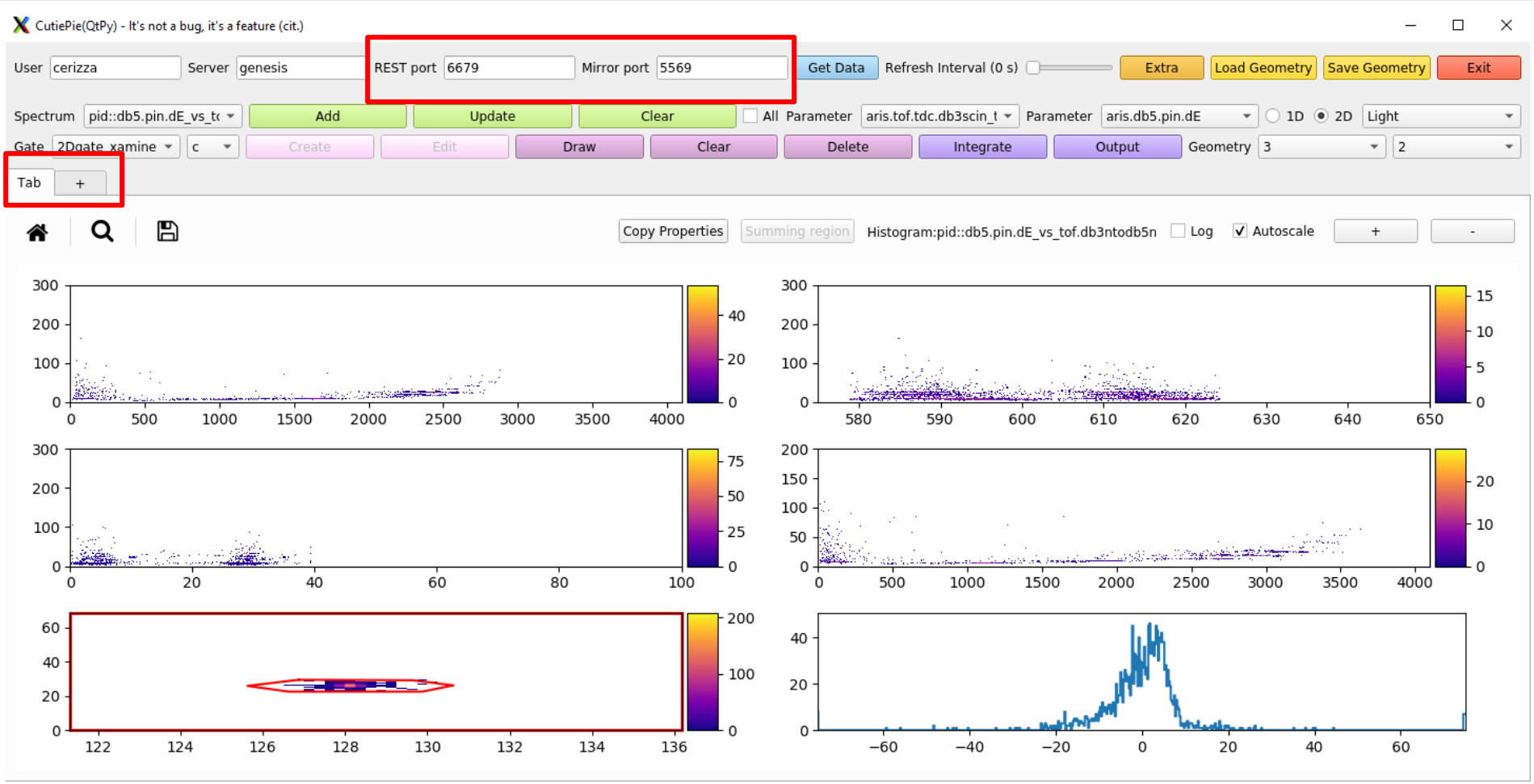
SpecTcl QtPy

- Analysis – new histogram viewer QtPy (CutiePie)
  - Alternative to Xamine
  - Linked to TreeGui in a bidirectional way
  - Based on Python/C API to extend the Python interpreter with new modules written in C/C++
  - It adopts common Python libraries for basic and more advanced numerical algorithms
  - Flexing and extensible GUI written in PyQt5
  - Expanded list of analysis tools and frameworks

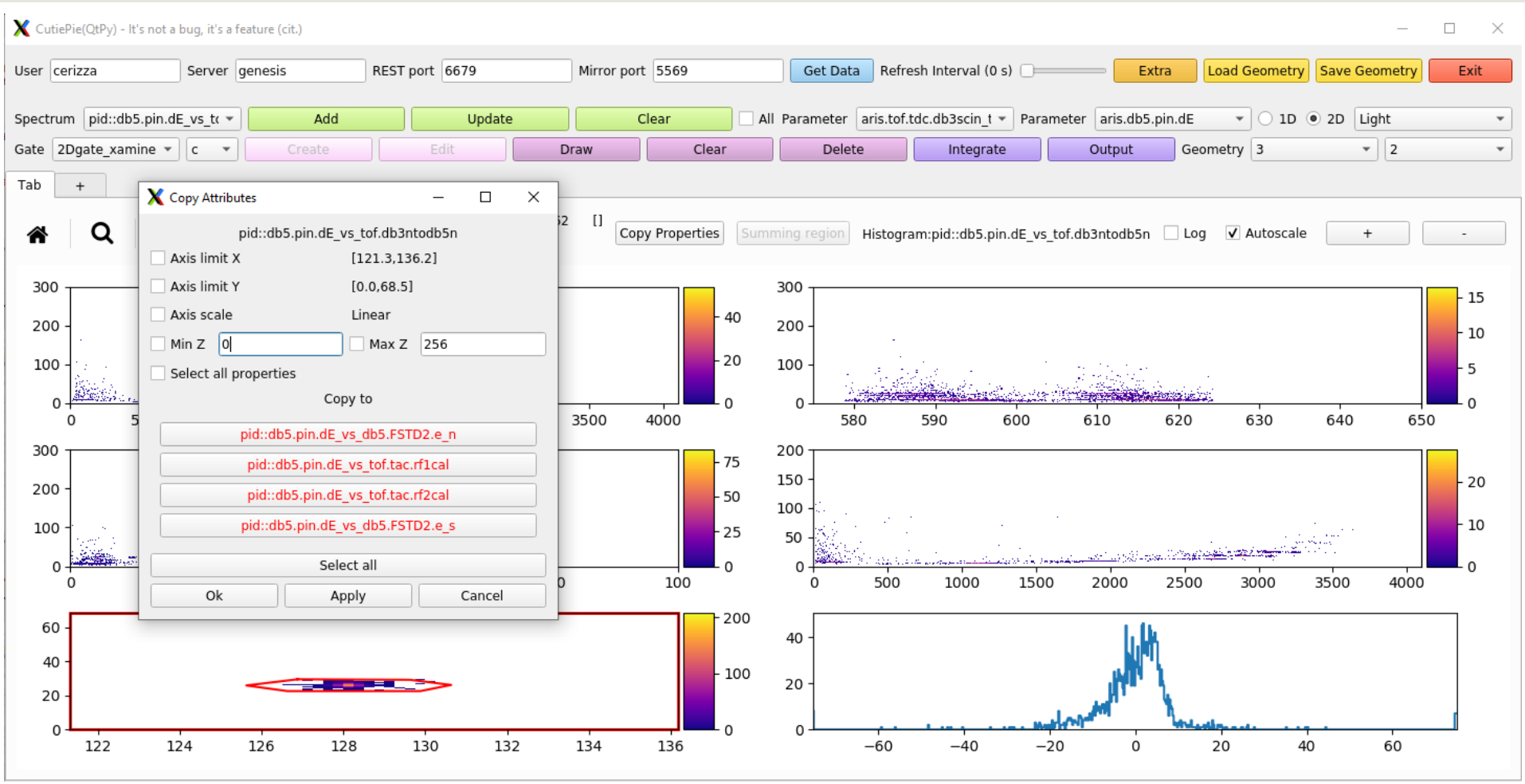
SpecTcl 5.13-000 will be available soon under the buster container env

Let's have a tour…

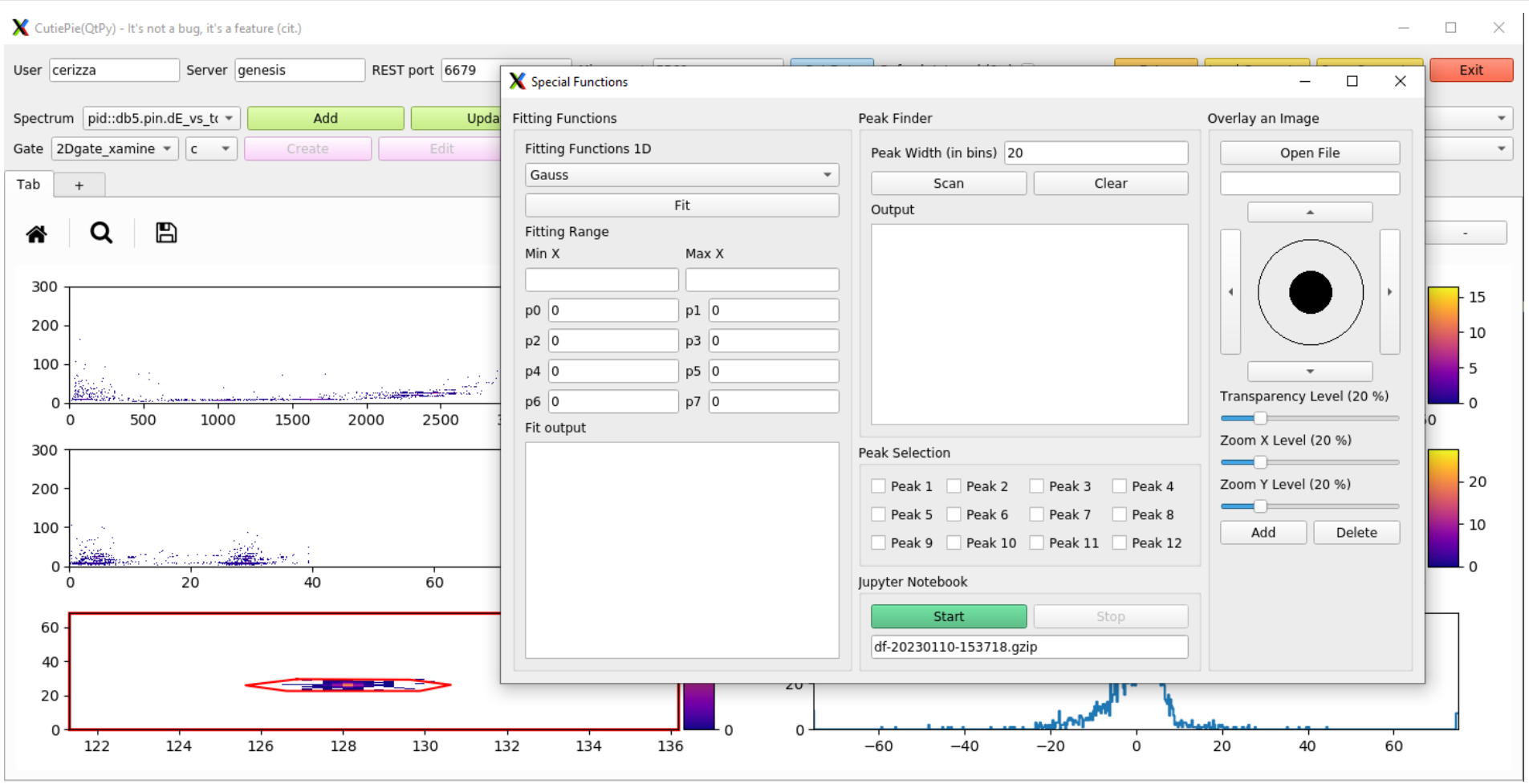
SpecTcl QtPy



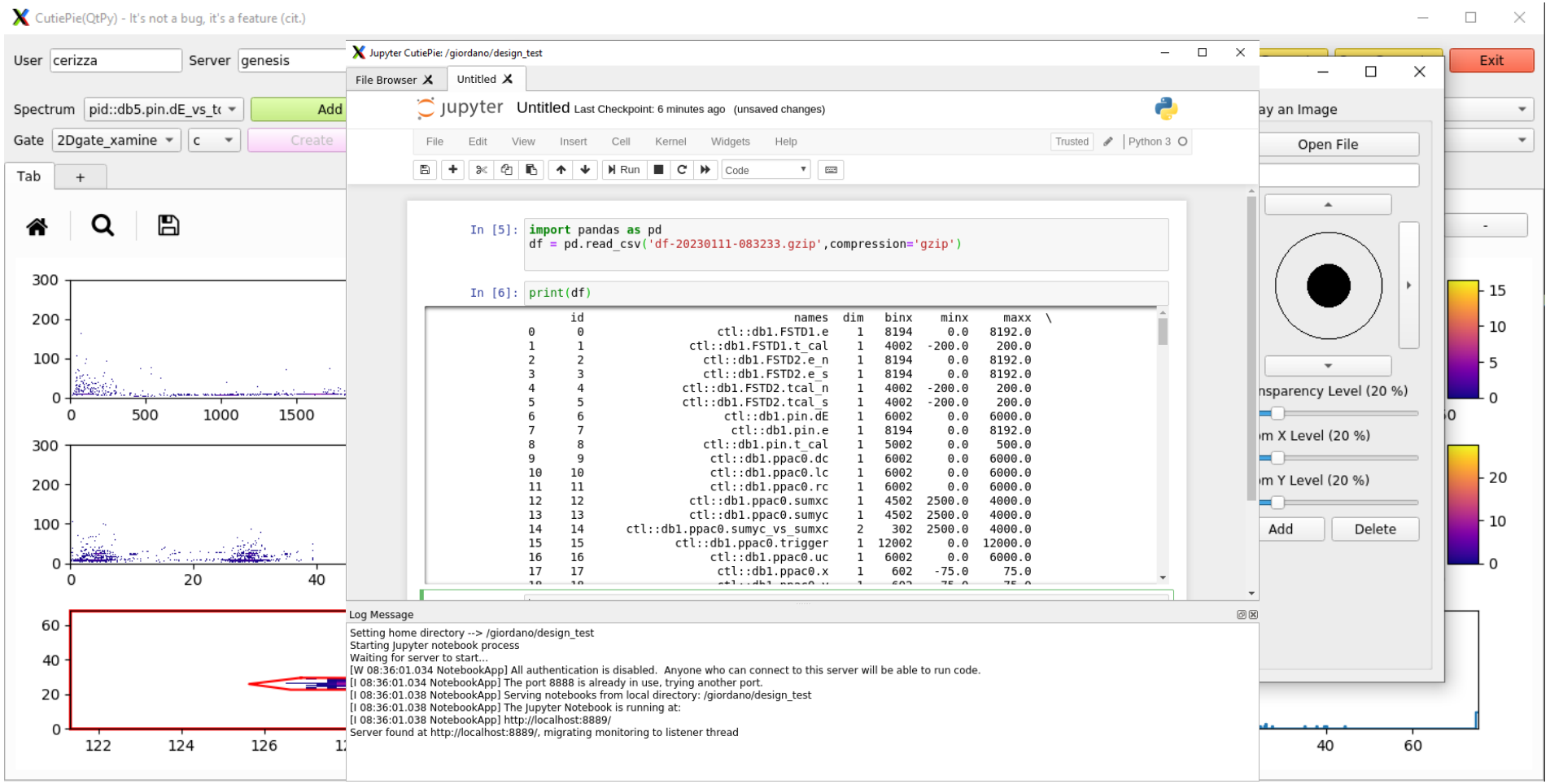
SpecTcl QtPy



SpecTcl QtPy

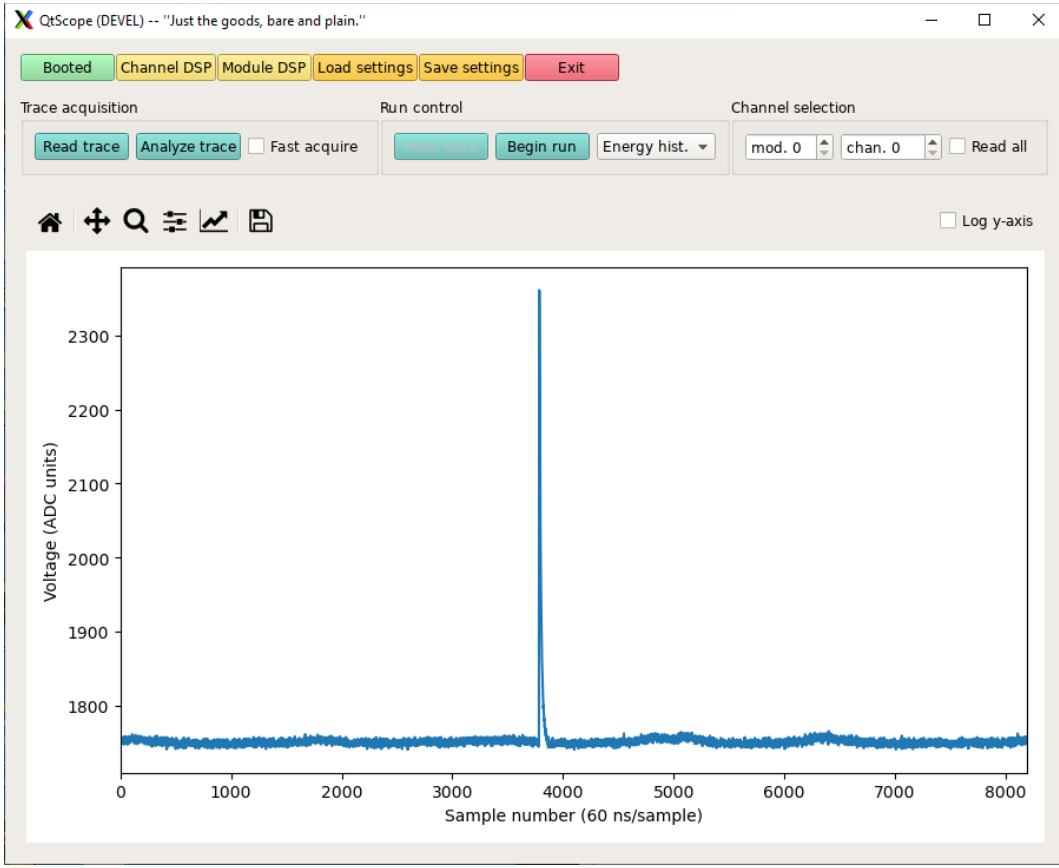


SpecTcl QtPy



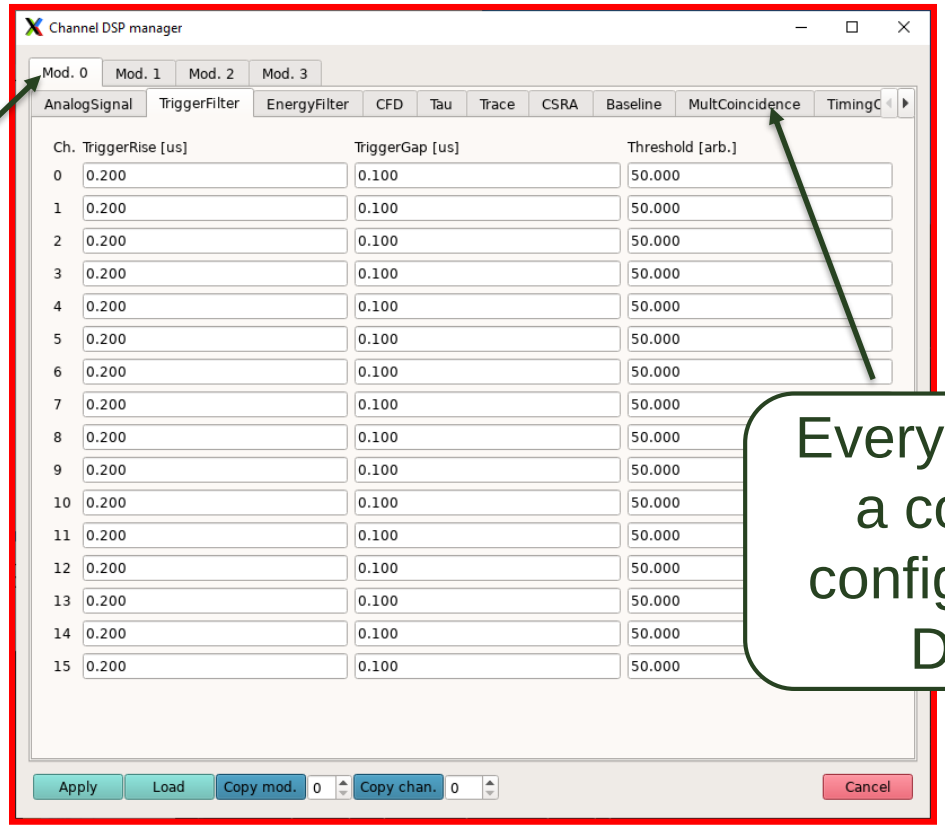
QtScope

- Updates and improves Nscope
- ROOT replaced by PyQt5 UI
- New backend for communicating with Pixie-16/32 hardware
- Improved UI blocking input when application is busy
- Back compatibility with old set files
- Beta-testing phase (need vols!)



QtScope

- DSP for modules in a single crate are configured through one menu

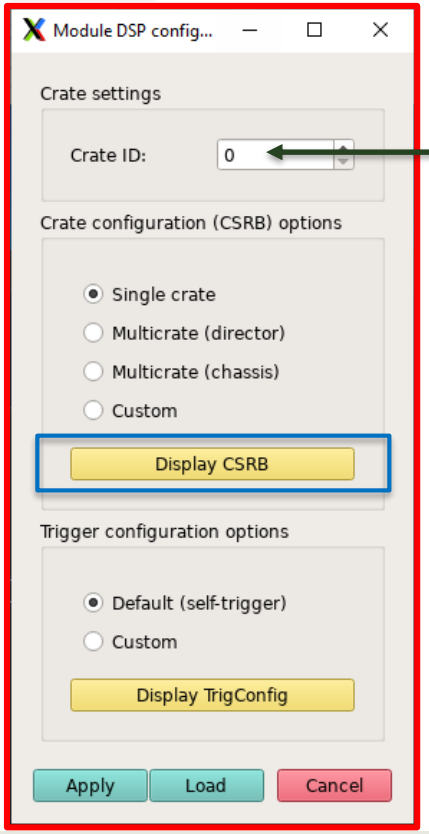


Single tab for each connected module

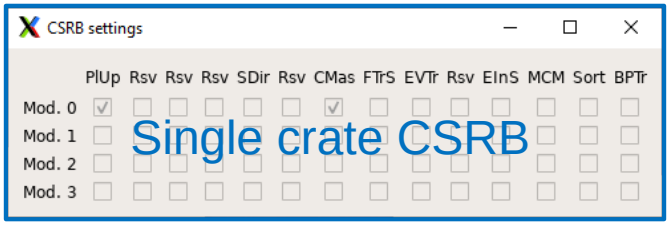
Every module tab has a complete set of configurable channel DSP settings

QtScope

- Simplified interface for configuring module parameters



Crate-wide parameters configurable in a single location/step



One-button setup and for common system configurations

QtScope – Future Dev

- Human-readable DSP settings files
- Support for new XIA hardware (PixieNet-XL, Pixie 32)
- Interactivity with Readout instances to configure modules without rebooting
- …

A look to the future

- New solution for high-number of channel detectors
  - SRS
  - CAEN FERS

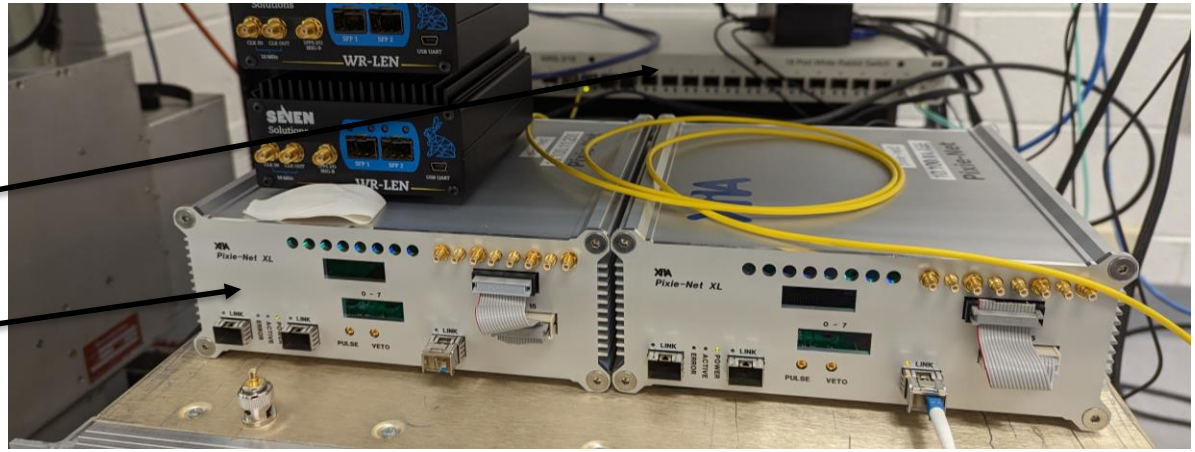
Hybrid board
(64 channels)

Scalable
Readout
System

- WR timing synchronization for XIA systems

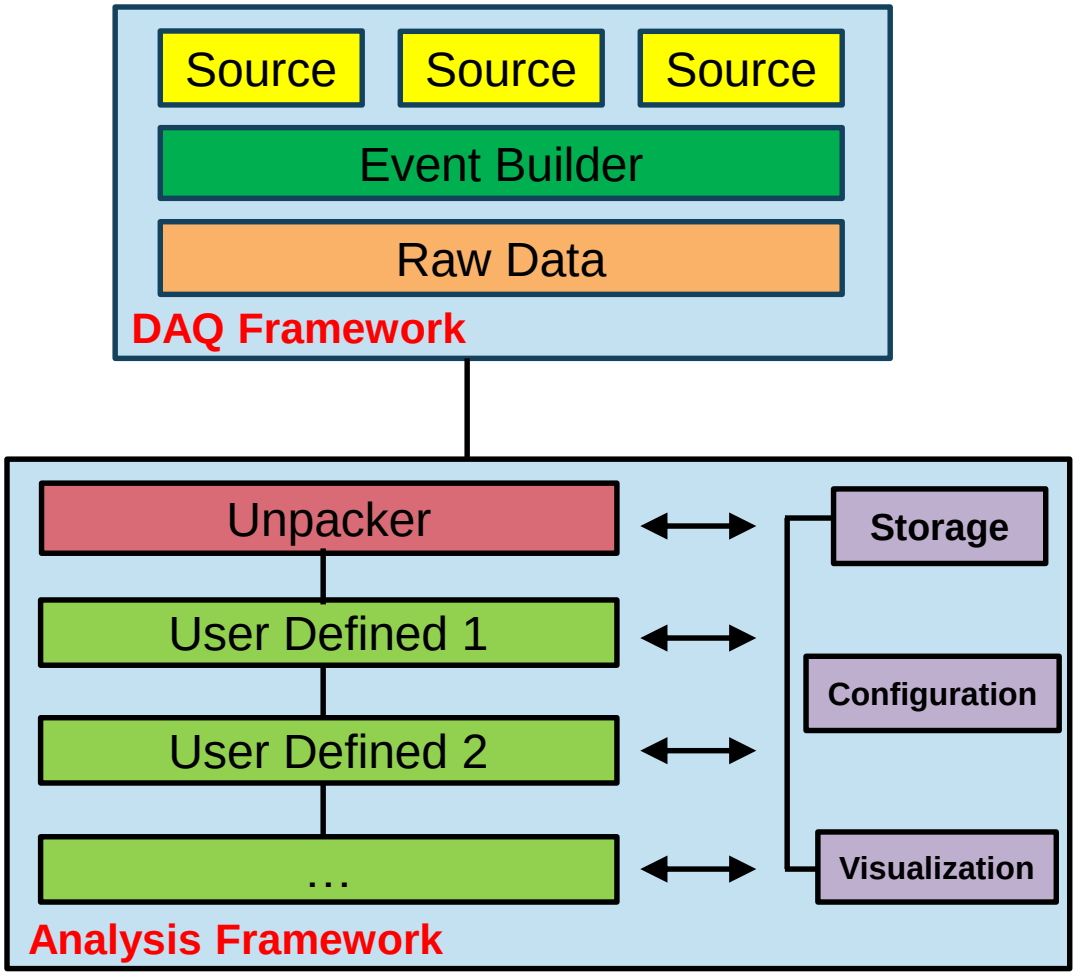
WR switch

PixieNet XL



A look to the future

- Plan to develop a new analysis framework more flexible at each stage



- All components can be parallelized
- Components can be individually enabled and disabled
- Framework is GUI controlled
- Framework will require a common set of (user) parameters and variables.
- Data can be written out of and read into each stage.
- Third-party libraries can be incorporated into user defined routines to increase functionality
- Visualization API will be defined to enable use of different visualizations (SpecTcl/ROOT/QtPy is likely to be provided by default)

Summary

- We are upgrading and modernizing what has been existing
- We are introducing new solutions for readout and analysis that minimize the effort for users to modify their existing programs
- We are open to listen to ideas/suggestions that could steer our work (i.e. as successfully demonstrated with e17011)

https://tinyurl.com/FRIBDAQ

I want to thank Ron Fox for all the work he did in the past 30 years for NSCL and FRIB. His legacy will be part of this lab forever.

“So long and thanks for all the fish”