Modernizing Nuclear Physics Data Processing

S.V. Paulauskas, Ph .D, PMP

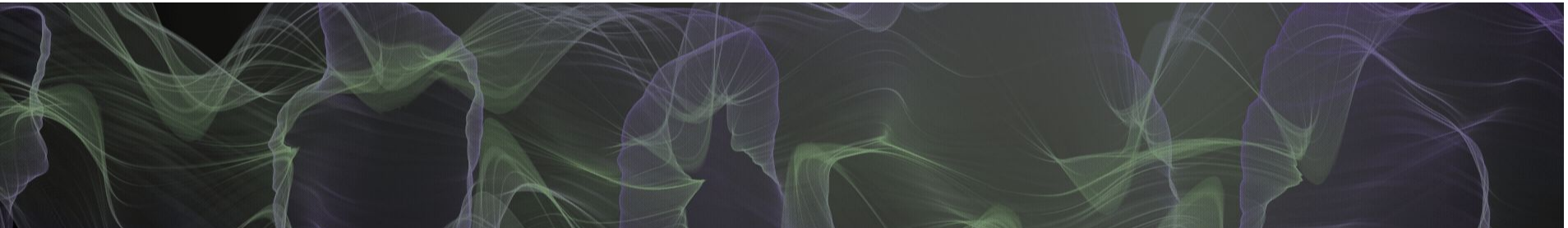
Brief Bio

“I wonder if I've been changed in the night. Let me think. Was I the same when I got up this morning? I almost think I can remember feeling a little different. But if I'm not the same, the next question is 'Who in the world am I?' Ah, that's the great puzzle!”
— Lewis Carroll, Alice in Wonderland

- B.S., Physics, TTU (2007)
- Ph. D., Nuclear Physics, UTK (2013)
- Ran experiments at
  - ORNL
  - NSCL/FRIB
  - Notre Dame
  - Ohio University
  - CERN/ISOLDE
- Working at Jewelry Television (JTV)
  - Building “big data” platform
  - Working with streaming data
  - Applying ML techniques to enhance business value (whatever that means)

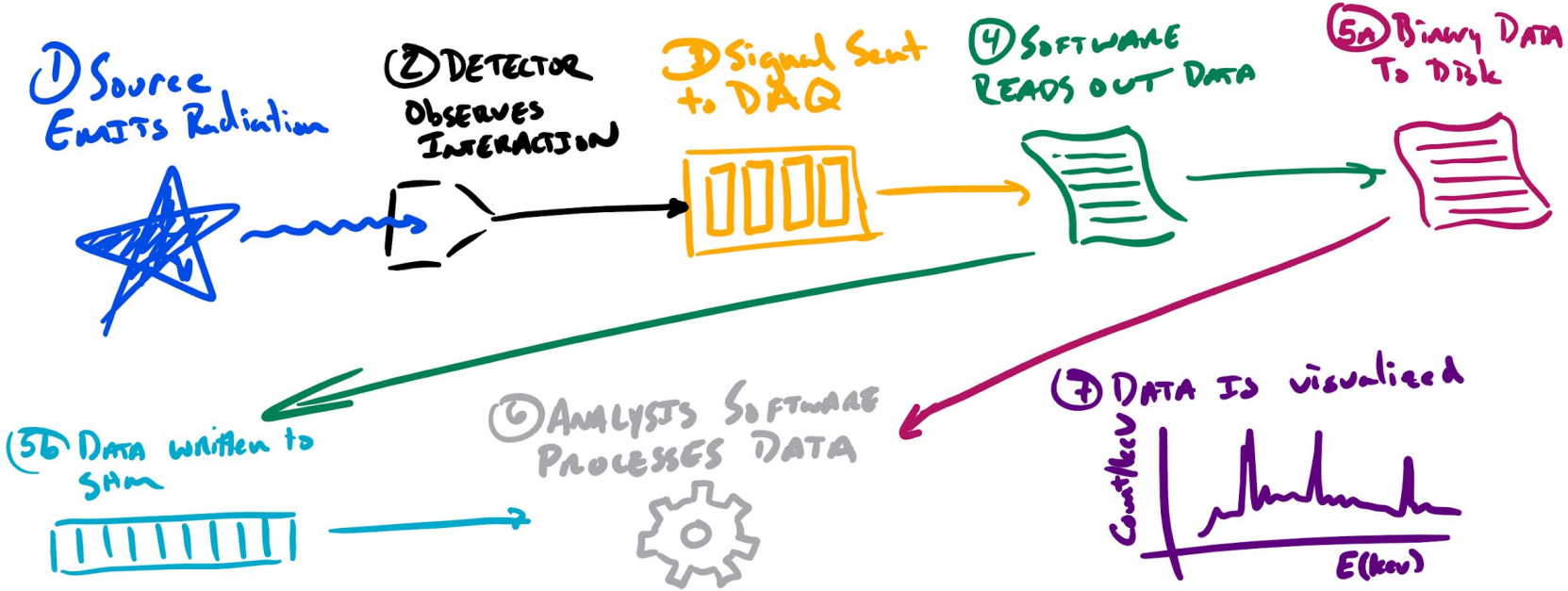
Overview

Many research groups lack a modern data acquisition (DAQ) and analysis system. Most use the same software for years, hacking away with students and post-docs.



Objective

Modernize our workflow using open, supported software solutions as a framework.



Typical Data Flow Model

Pros and Cons

- Solid, well-established workflow
- Feedback loop is short for small experiments
- Visualization is simple since it's an aggregate
- Can replay data from disk

- Serial processing of data can be slow
- Correlation across systems difficult
- Not easily parallelizable due to nature of software
- Joins across data segments can be impossible
- Need to unpack data every time we analyze

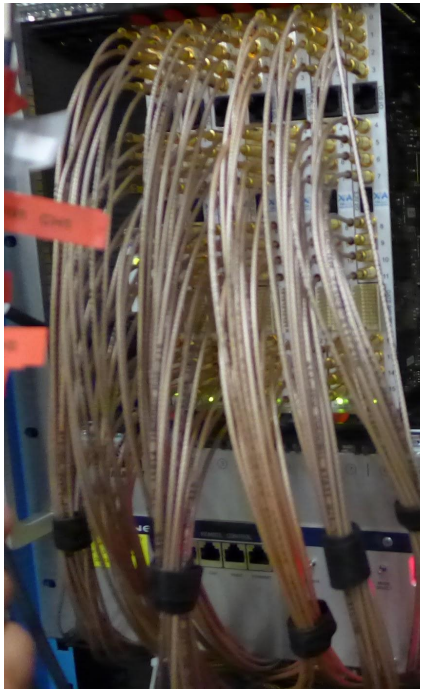
Data Acquisition

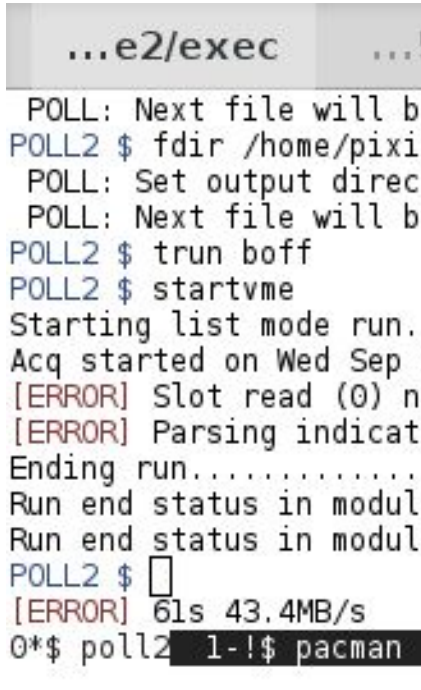
PAASS - ORNL / UTK: DAQ

A framework that started development in the 70's - 80's.
Maintained by the folks at UTK/ORNL.

Features:

- Written in C++ by tons of authors over many years
- User Datagram Protocol for network communication
- Text based / CLI interface and controls
- Limited to consuming from a single data producer
- Limited to producing to two (ish) data sinks
- Based around XIA LLC's digital electronics



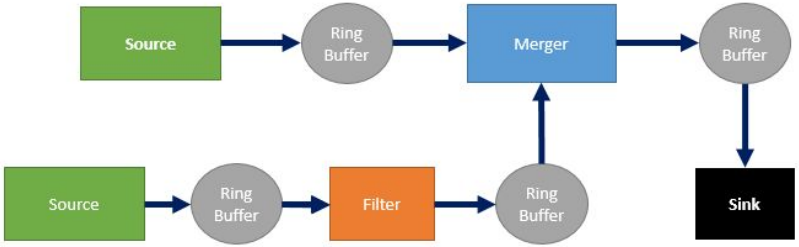


NSCL/FRIB:
DAQ

Framework developed at NSCL/FRIB. Maintained by a full-time staff of physicists.

Features:

- Covers any system used at the lab
- It uses ringbuffers for shared memory
- Data are binary EVT files
- Extensible by users
- Supports both digital and analog electronics (mutually exclusive)



http://docs.nscl.msu.edu/daq/newsite/nscldaq-11.2/c5.html#AEN8

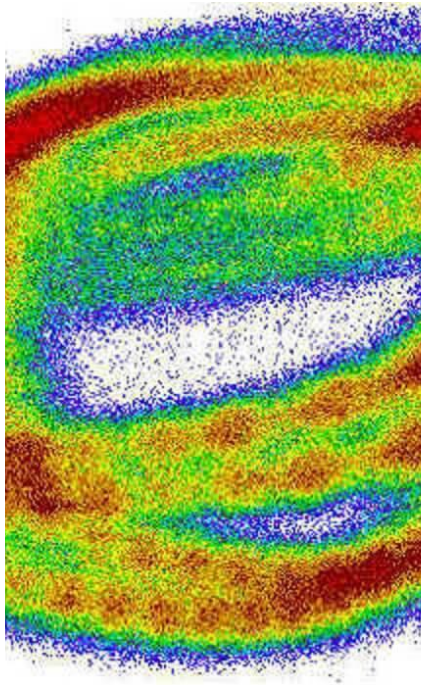
Data Analysis

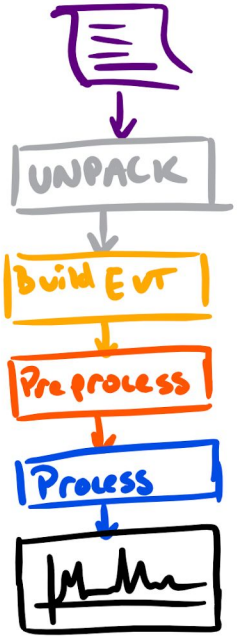
PAASS - ORNL / UTK
Analysis / Visualization

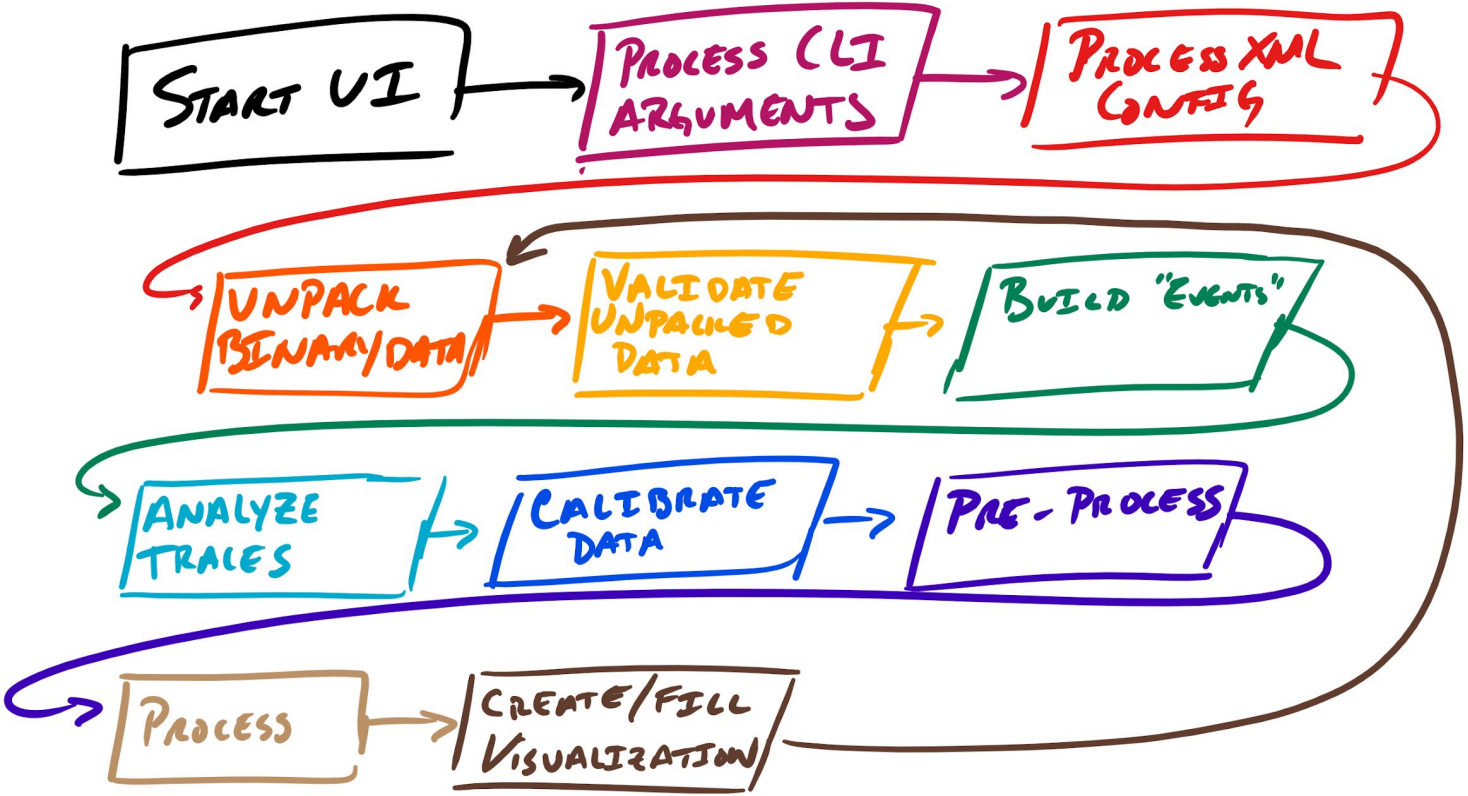
A framework that started development in the mid 2000's. Maintained by the folks at UTK/ORNL, only under internal development at this time.

Features:

- C++ software that relies on serial processing
- DAMM for visualization (FORTRAN from 80's)
- Supports User Defined Processors







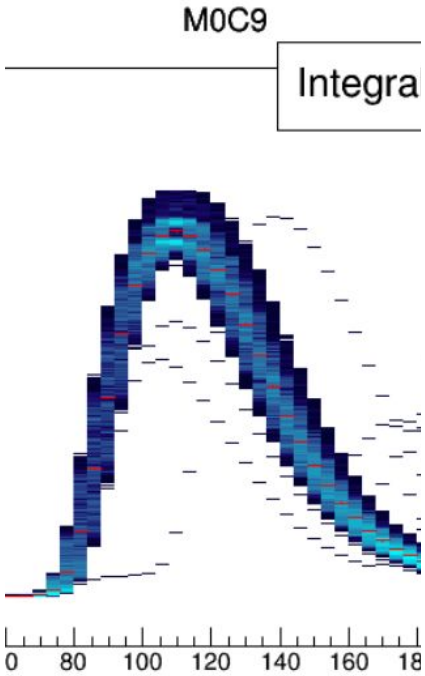
PAASS - Workflow

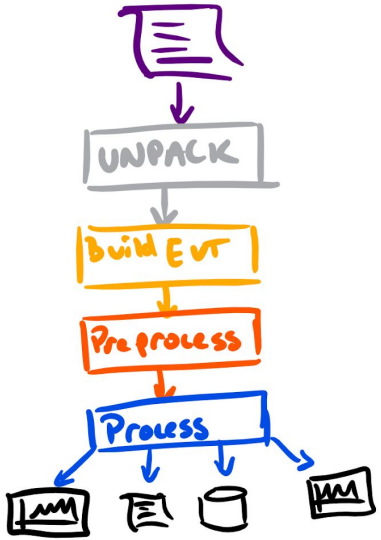
PAASS-LC - Project Science
Analysis / Visualization

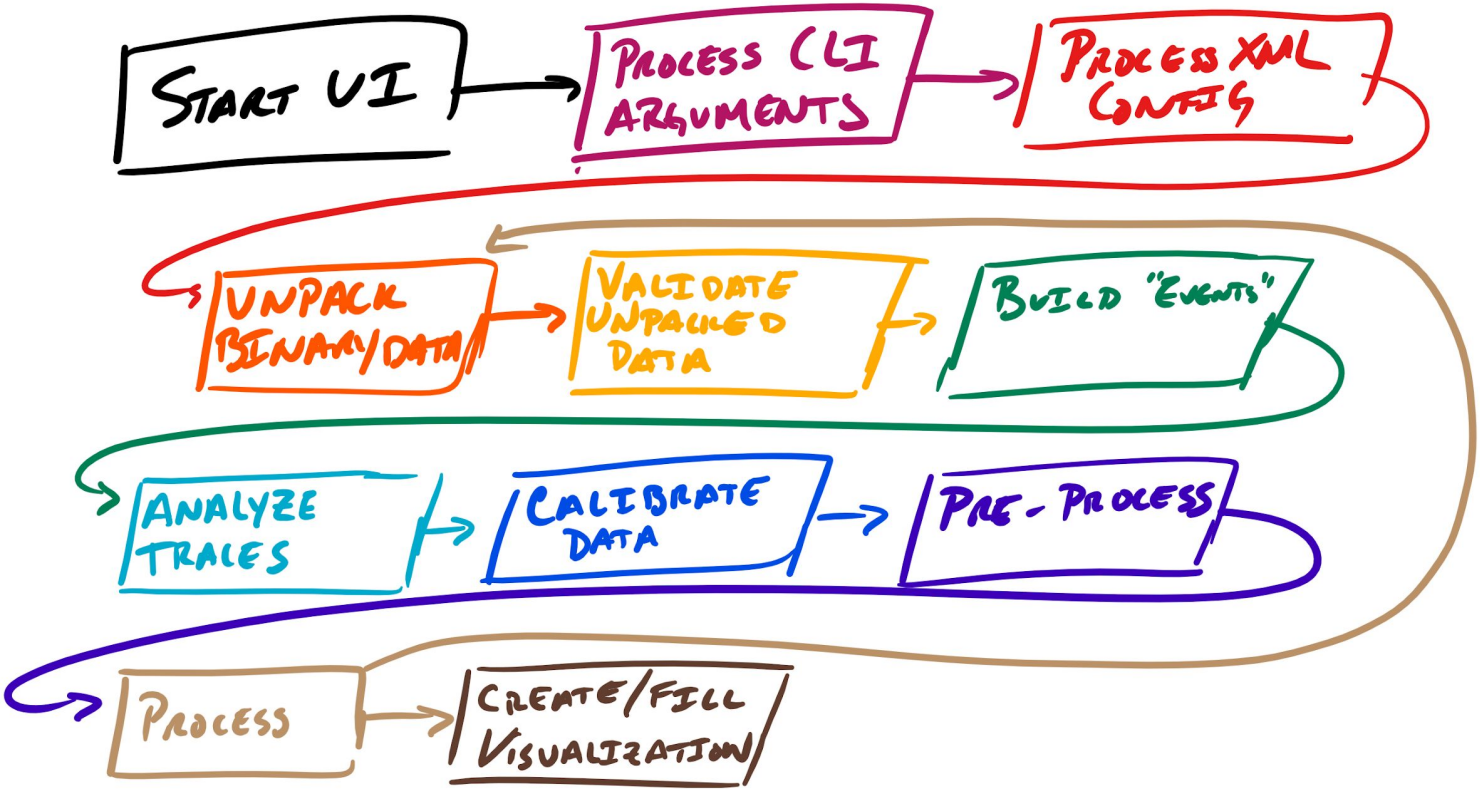
Based on the UTK software. It's developed and maintained by Project Science. Will soon be deprecated.

All features of PAASS, plus:

- Parallel processing
- Native ROOT visualization
- CI server
- Expanded unit tests







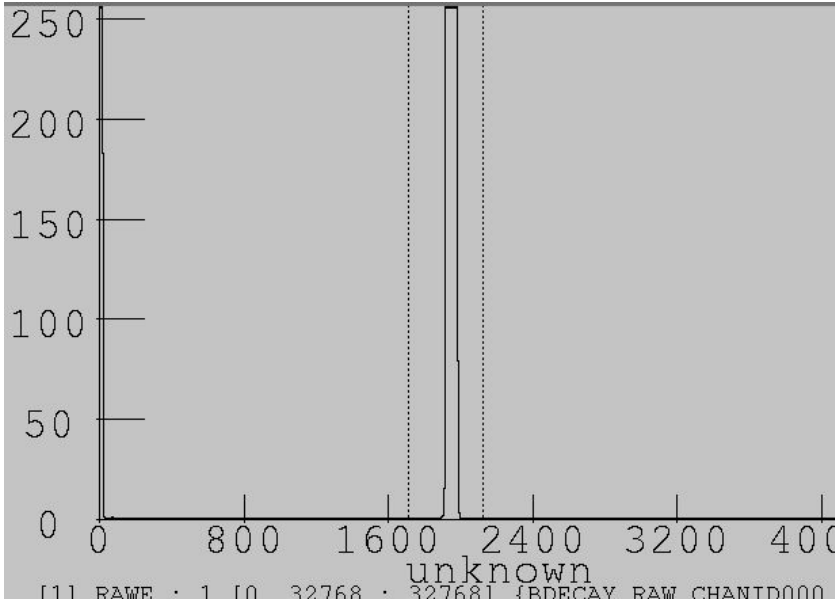
PAASS-LC Workflow

SpecTcl - NSCL/FRIB
Analysis / Visualization

Framework developed at NSCL/FRIB. Maintained by a full-time staff of developers (physicists).

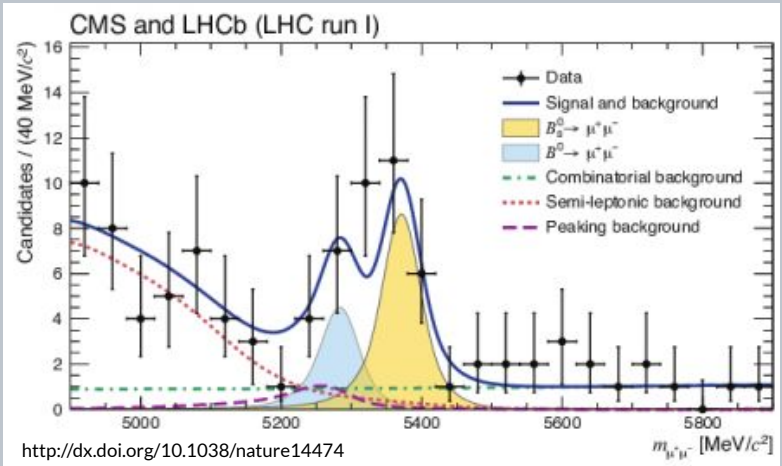
Features:

- Written in TCL
- Expandable with C++ code
- GUI
- Serial Processing



Visualization

ROOT - A data analysis framework

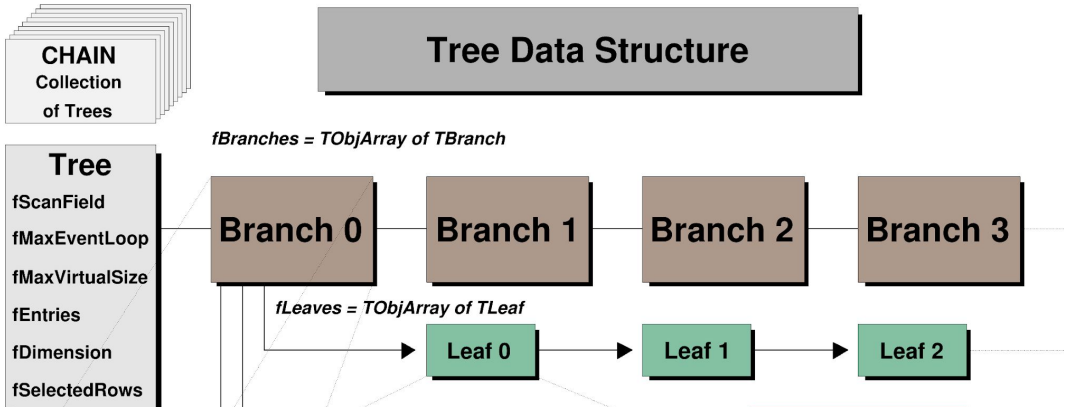


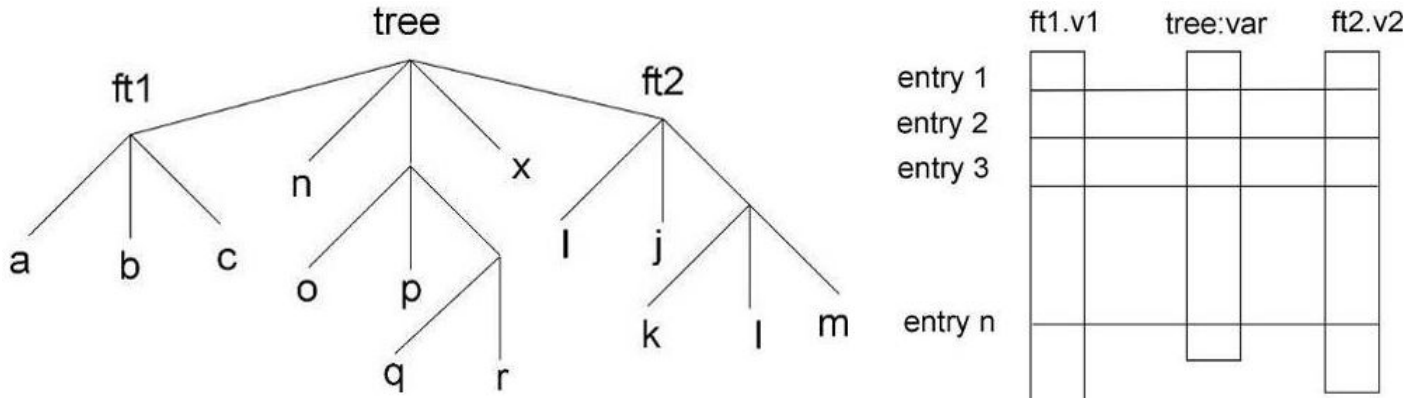
http://dx.doi.org/10.1038/nature14474

- C++ based software package from CERN
- Incredible number of prepackaged algorithms / data processing
  - 2d and 3d histograms
  - Least-squares / Maximum Likelihood
  - Fourier transforms
- Produces publication-quality plots
- ~50 million lines of code
- Devs on the C++ Standards committee

ROOT TTrees

- Custom NoSQL DB framework
- Row-oriented storage
- All cross-tree correlations require full Tree scans.

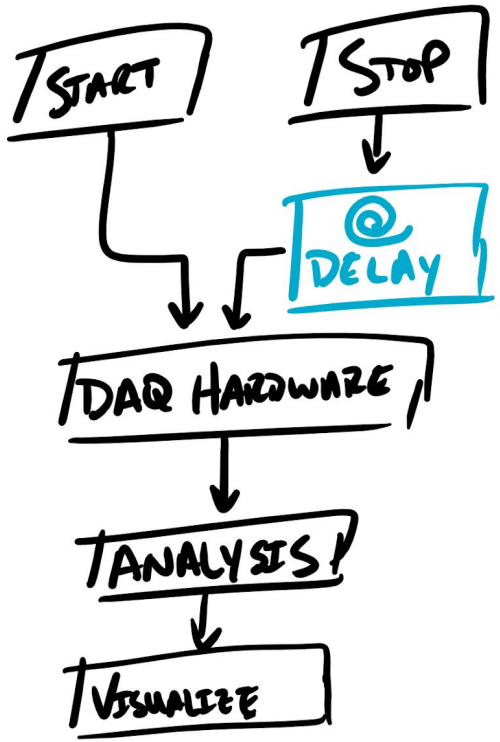




Let’s roll this all up and look at some actual analysis techniques.

High-Resolution Timing Analysis

Let’s consider a simple time-of-flight model.

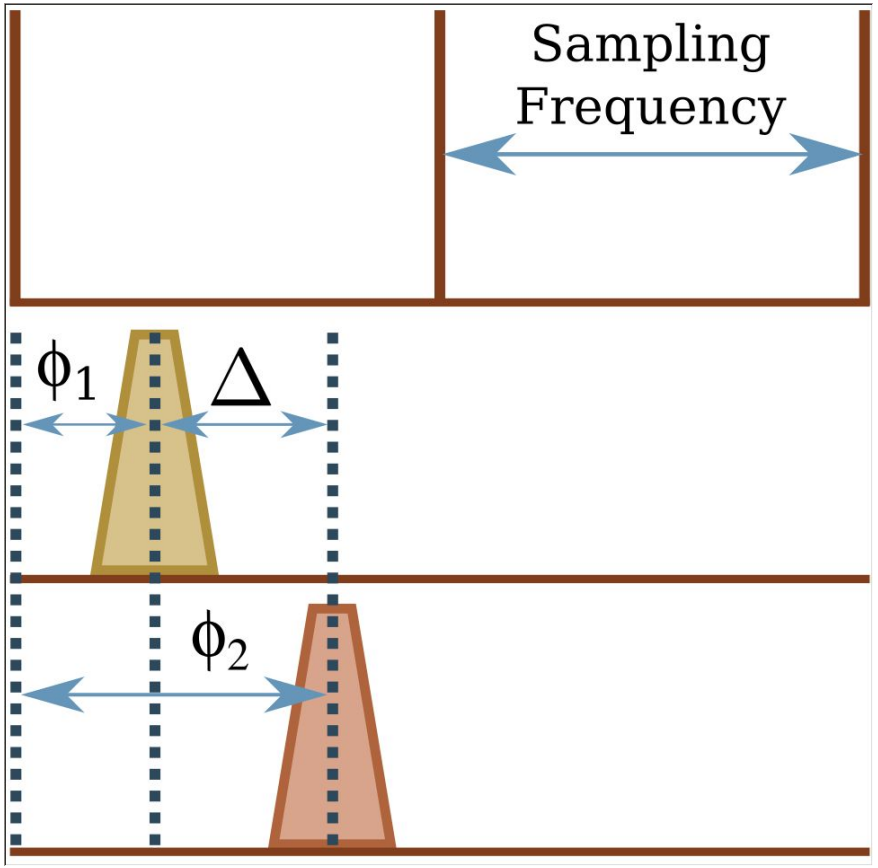


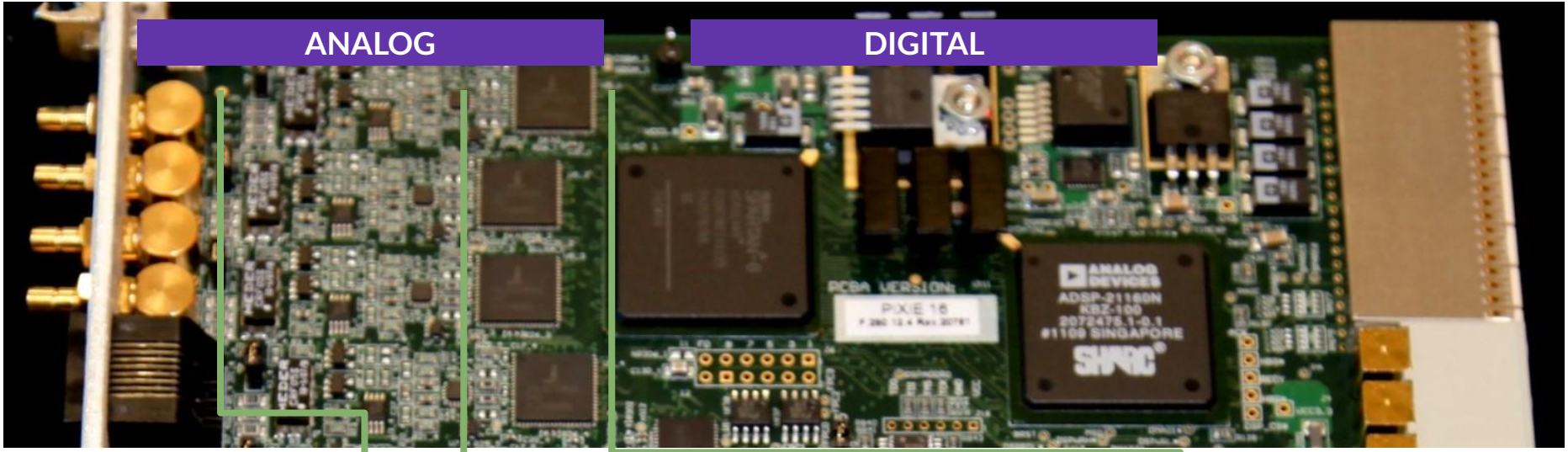
Signal Relationships

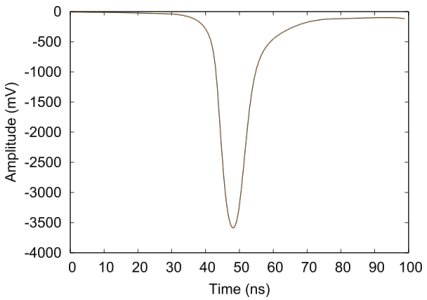
Consider two signals separated by some time delay. We expect

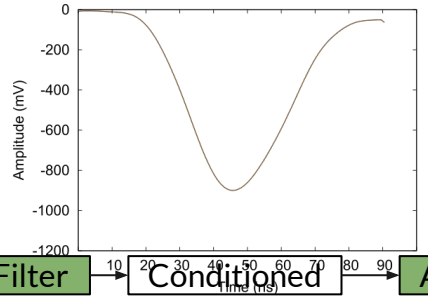
$$\phi_1 - \phi_2 = \Delta.$$

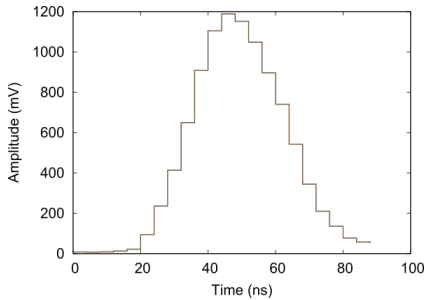
All signal transformations **must** preserve this relationship!

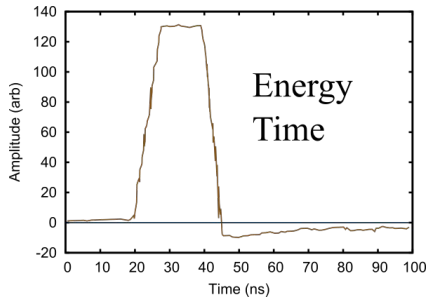












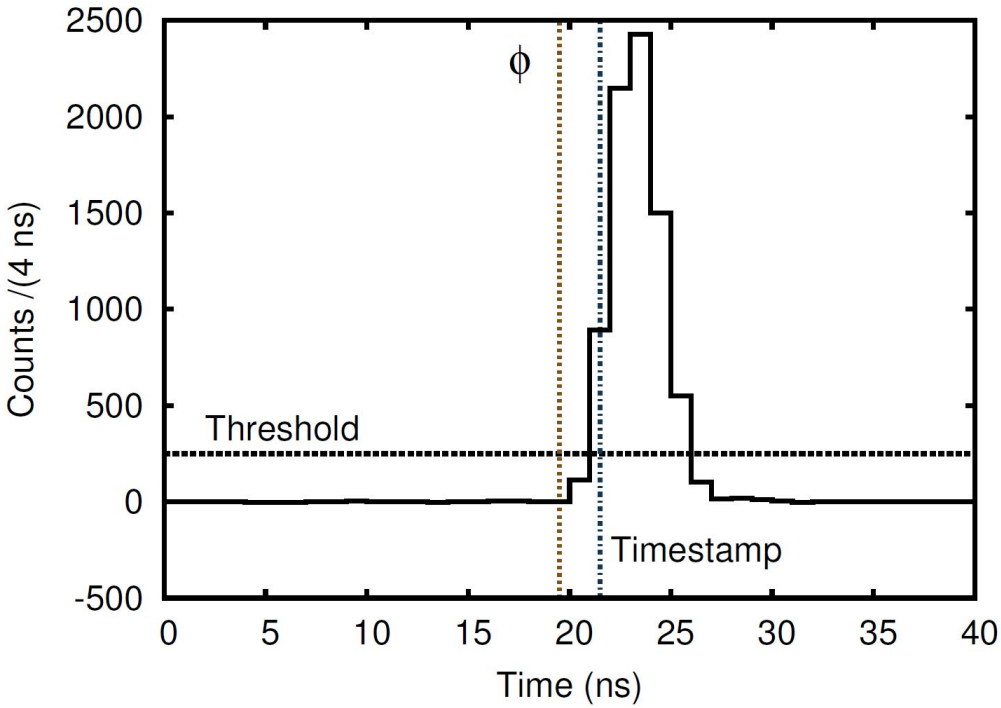
Input → Low-Pass Filter → Conditioned → ADC → Sampled → Digital Filter → Filtered

Anatomy of a Digitized Signal

- Full signal called a trace
- Region of interest called waveform

ADC for this system samples every 4 ns.

Question: What is the timing resolution of the system?

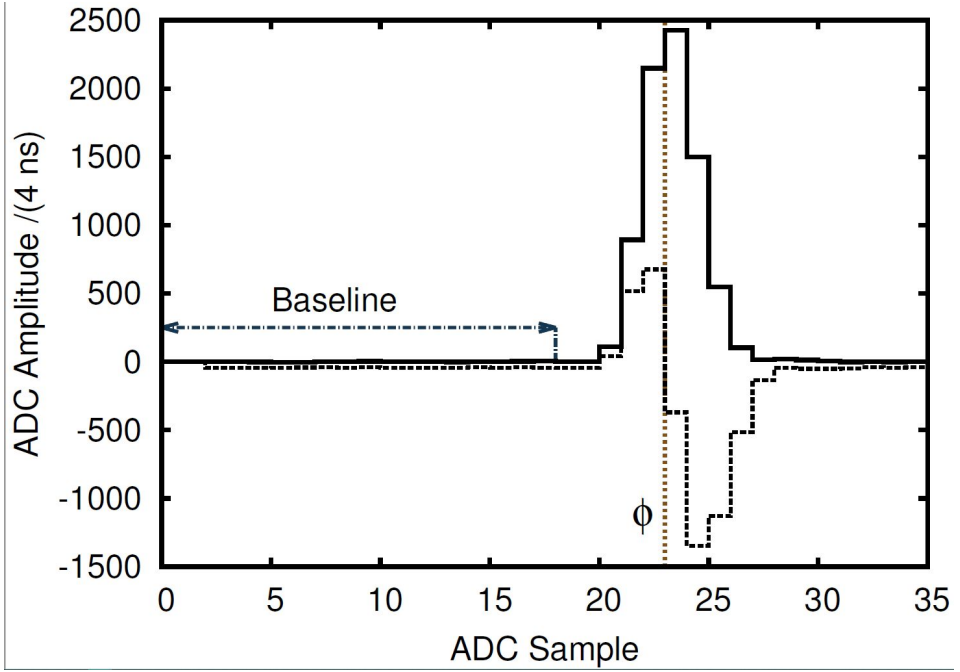


Constant Fraction Discriminator (CFD)

It sums an inverted shifted signal with the original to produce an amplitude invariant arrival time.

```
def tradCFD(trace, f, d, l):
    return [f*(trace[i] - trace[i + d]) for i in range(len(trace)-d)]
```

What are the problems with this type of analysis?



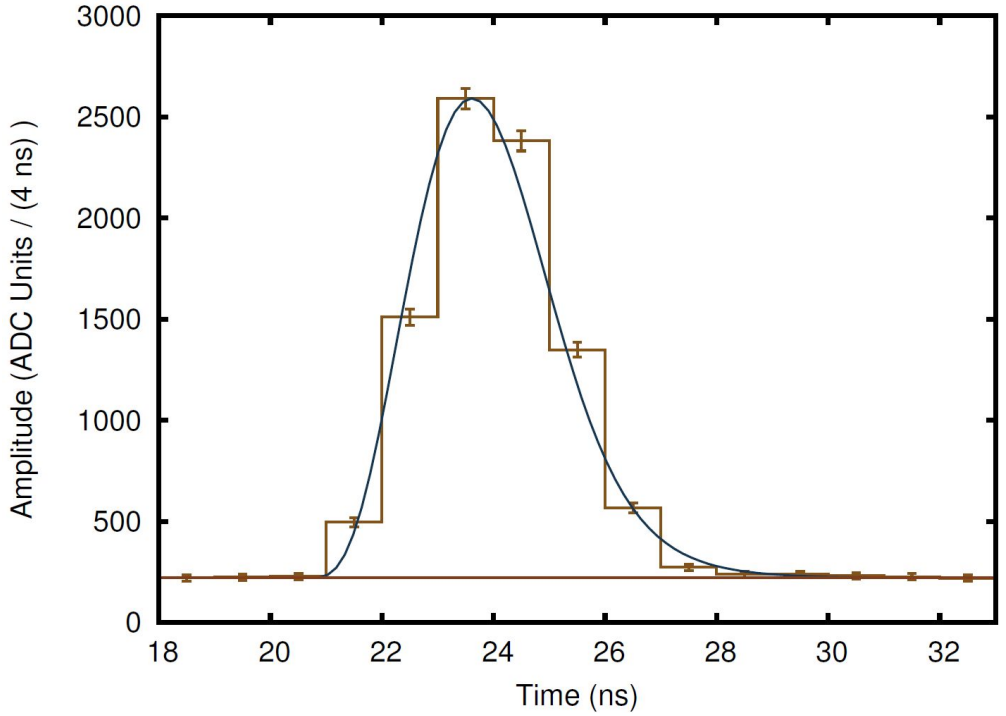
Fitting Algorithm

CFDs do not take into account non-linearity in the signal. A fitted function explicitly takes this into account.

$$f(t) = \alpha e^{-(t-t_0)/\beta}\left(1 - e^{-(t-t_0)^4/\gamma}\right)$$

We fix the shape (β and γ) allowing the algorithm to determine amplitude/phase.

Allows for **sub-sampling** time resolutions!

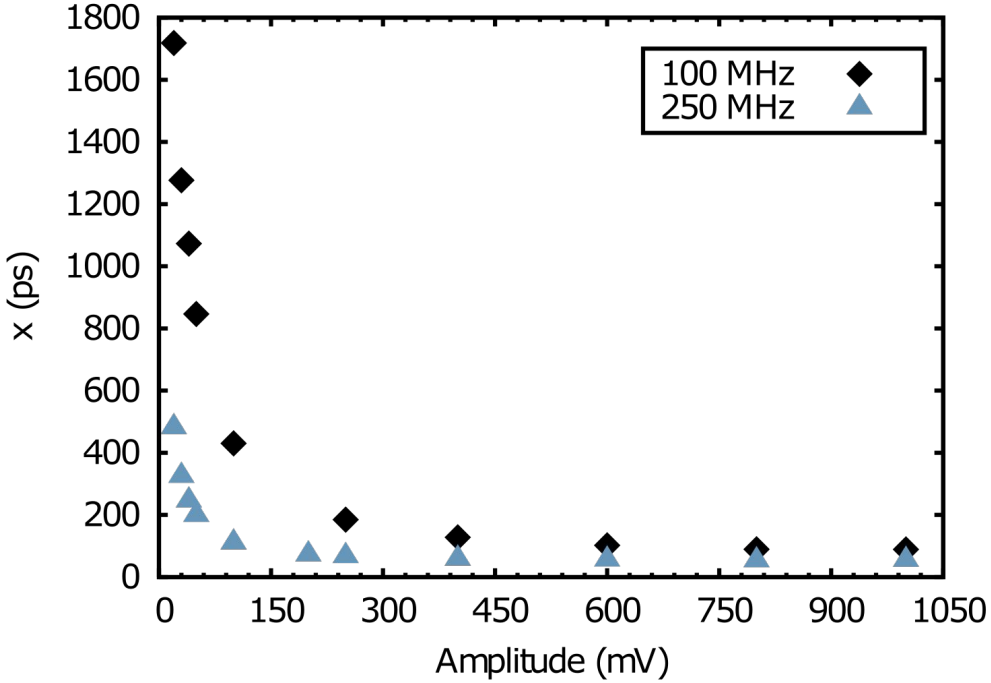


Algorithm Results

For ideal signals, we obtain results consistent with jitter in the digital system (~51 ps FWHM).

You can see the sampling-frequency effect on the time resolution.

This is great, but is it right?

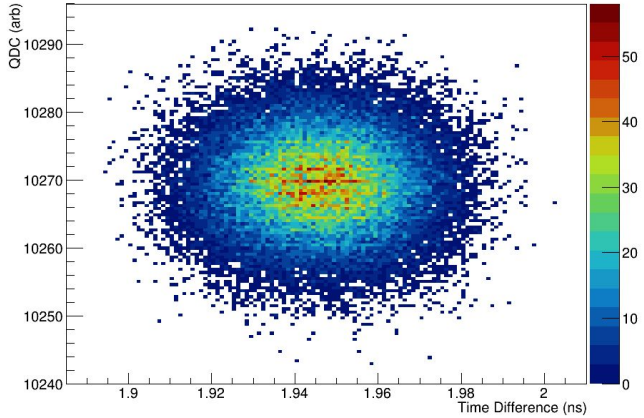


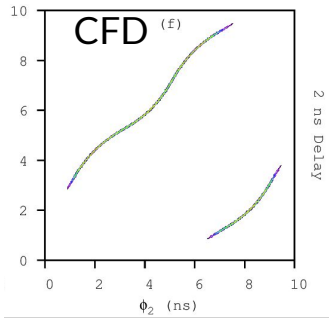
Algorithm Validation

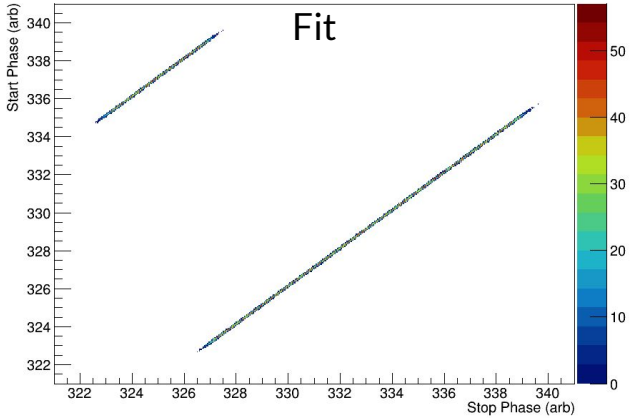
As mentioned, the phases must have a linear relationship.

Algorithms that don't take into account non-linearity in the signal produce non-linear results!

For more: [S. V. Paulauskas NIMA 737 (2014) 22-28]







We do the same work in physics that everybody else does...

Trends

There are many more choices out there. Each lab/group seems to have their own package. Most manufacturers provide some sort of framework (XIA, CAEN, etc). Few of these use software they didn't write. They completely ignore industry.

Physics

- Processes streaming data from detectors
- Analyzes data in real time
- Needs high-fidelity data storage
- Needs to analyze data from disk

Industry

- Processes streaming data from event streams
- Analyzes data in real time for business decisions
- Needs high-fidelity data storage
- Analyzes CSV, database entries, etc.

… How do we fix our process?

Problems to solve

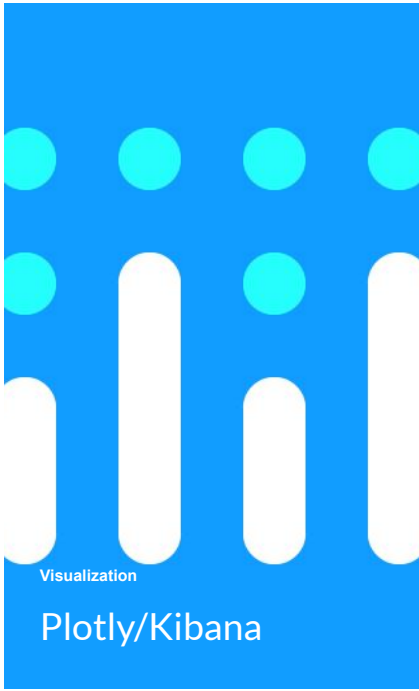
1. How do we provide an extensible, scalable, robust DAQ that integrates multiple systems?
2. What data storage solutions exist to replace binary data formats?
3. How can we implement a data-analysis framework that doesn't limit concurrent operations?
4. What frameworks exist that let us translate our model into "big data" stacks?

We have three main components: DAQ, Analysis, and Visualization.

(surprised no one in this room...)

Team

Each division has its own core leader.

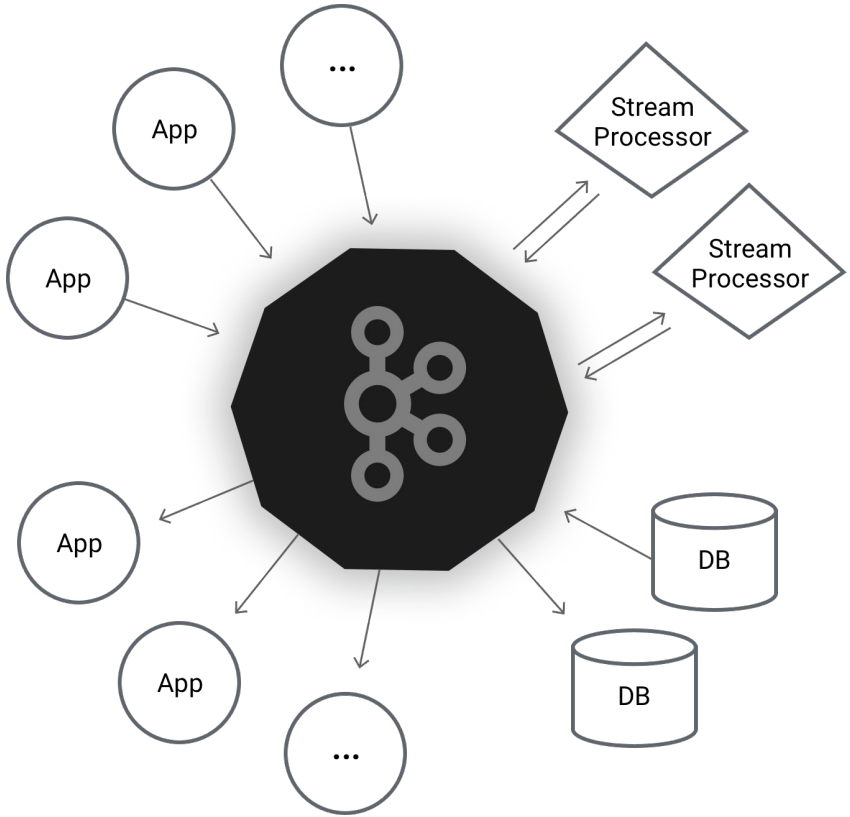


DAQ

Apache Kafka

Kafka is a messaging framework that allows us to manage communication between all of our different data sources. It allows for a multi-producer, multi-consumer model. We can collect and analyze in real-time.

- Fault-tolerant, replicated data streams
- Input : 1 M msg / sec | Output: 2 M msg / sec
- Huge amount of community support
- Could be used as data storage (not that I would)

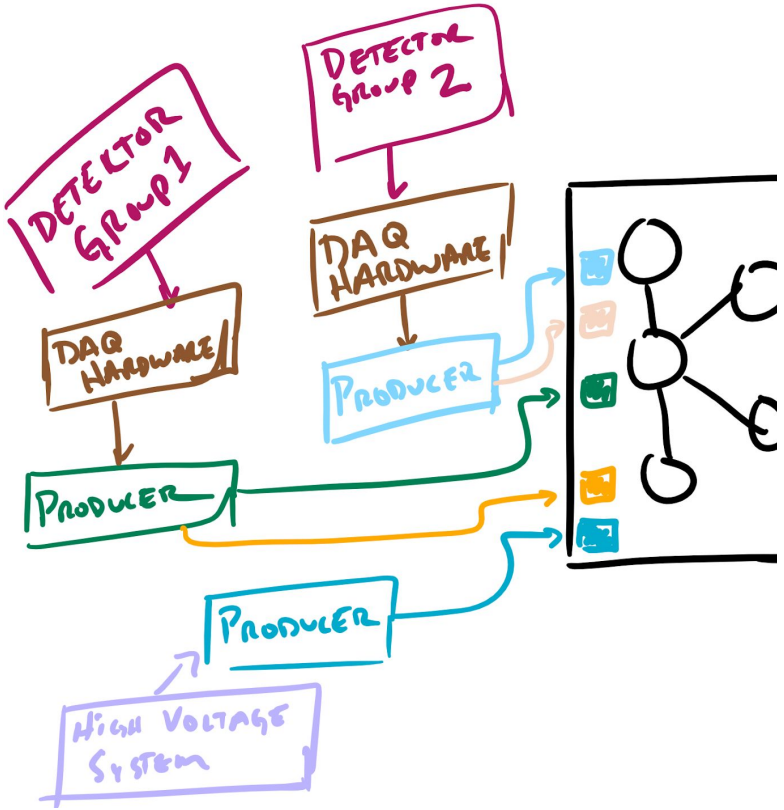


Producers

Using Kafka as a central messaging system, we combine previously distinct data sources.

We can aggregate from any number of auxiliary systems with the same timestamp.

Horizontally scales our workload, don't need beefy DAQ machines.

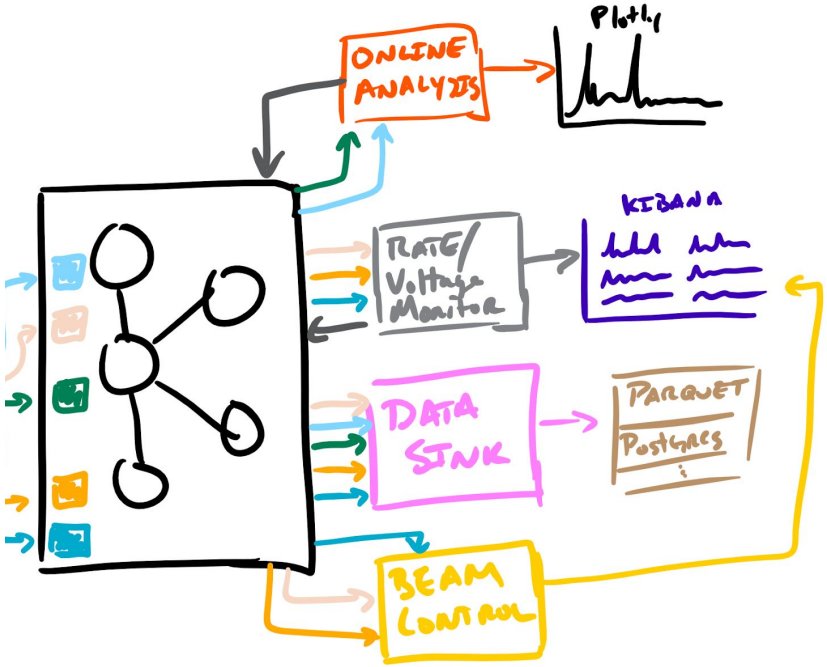


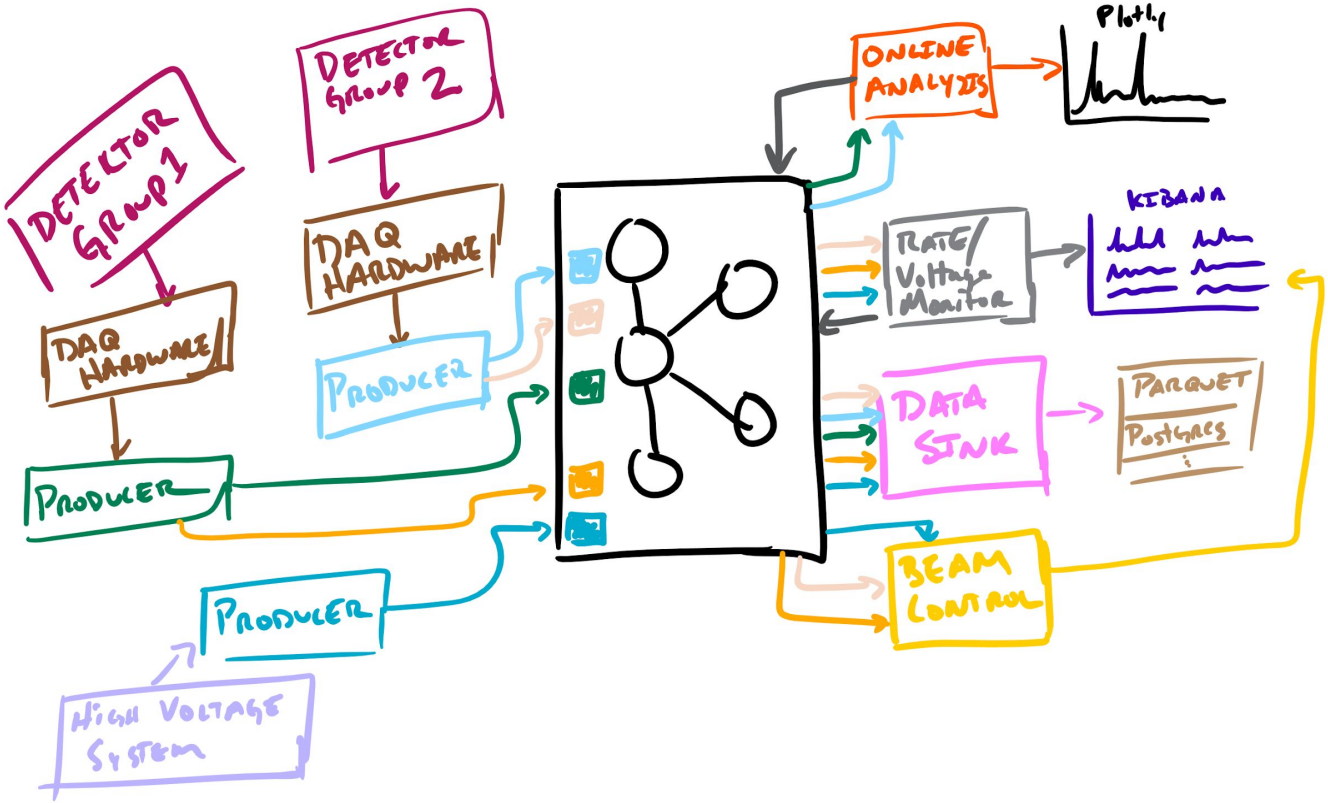
Consumers

We again scale horizontally. No monster analysis machines (though they help).

Can provide visualization and monitoring via industry-standard applications (Plotly, Kibana).

Researchers take their data home, Parquet /Avro files offer ideal storage.





Looking at the big picture

Analysis

Python

The de facto data-analysis programming language. We already have interfaces for all of the other tooling that we could consider. Incredibly simple to perform advanced analysis of data (e.g. ML). We'd be fools not to consider it as our main framework.

- I really shouldn't have to convince you.
- Let's take a breather to answer any questions, drink some water, etc.
- We're in the home stretch now.

Visualization

Plotly|Kibana

The “best” framework for visualization is really going to be application dependent. I suspect that a lot of the end-users are going to end right back up in ROOT. MicroStrategy may be useful. We’ll use an ELK stack for log aggregation and visualization.

- Plotly creates interactive plots
- Plotly can be used in Jupyter Notebooks
- Kibana provides trivial rate monitoring
- MicroStrategy provides reporting

Prototype rate monitor using a MIDAS consumer in Kibana from a real experiment on April 12th.

Plot a Histogram!

Crate

0

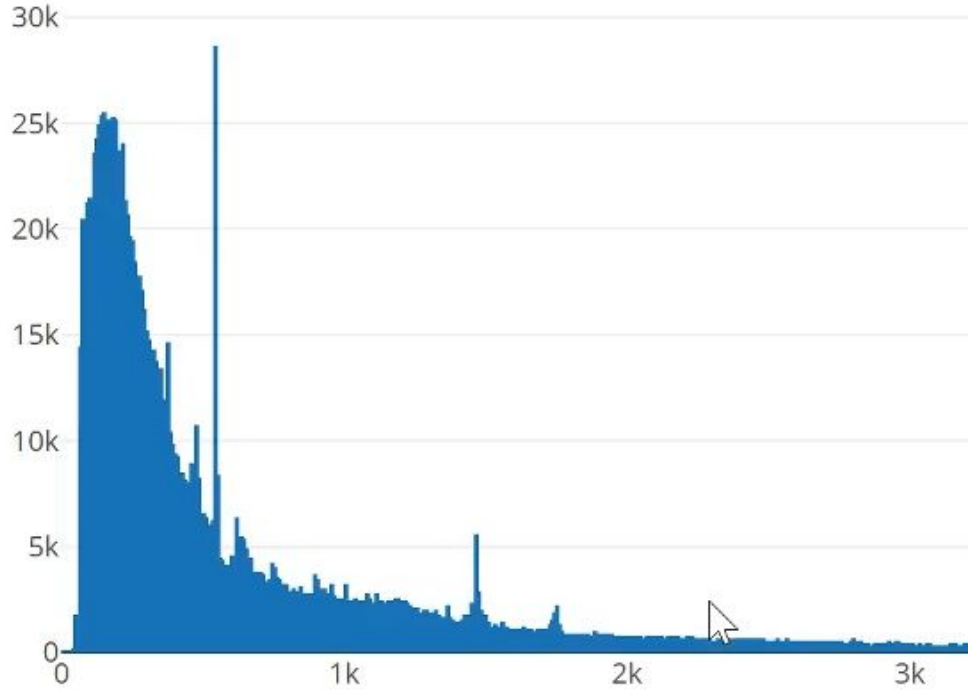
Slot

2

Channel

5

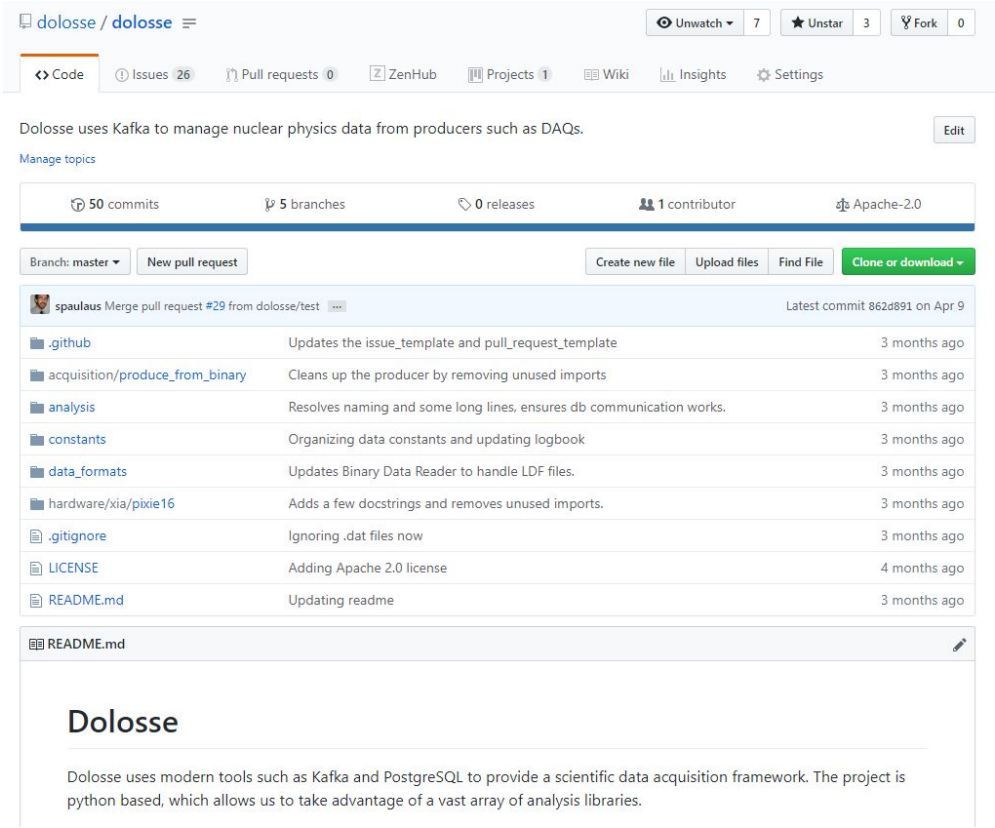
Display Energy Histogram



Simple form UI to visualize data w/ Plotly from a Postgres DB. This is just MVP.

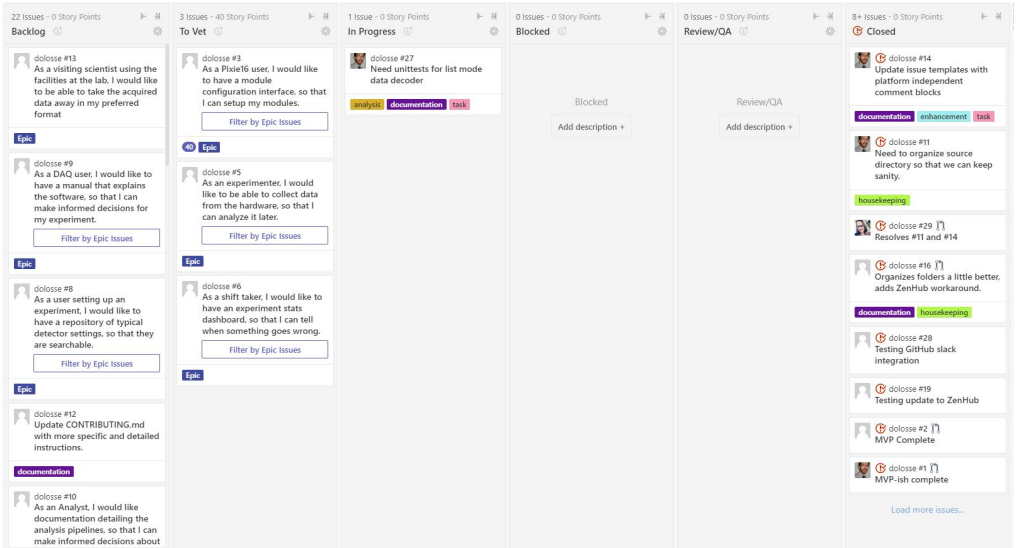
Dolosse: A streaming data acquisition and analysis framework

- Combines the previous slides
- License: Apache 2.0
- Collaboration:
  - Project Science
  - iThemba LABS
  - University of Stellenbosch
  - University of Western Cape
  - Vanderbilt University



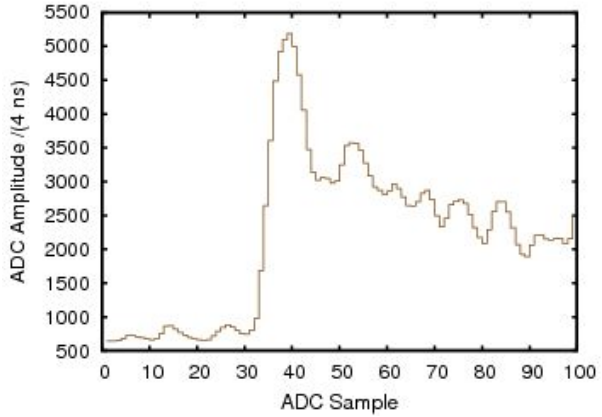
Project Status

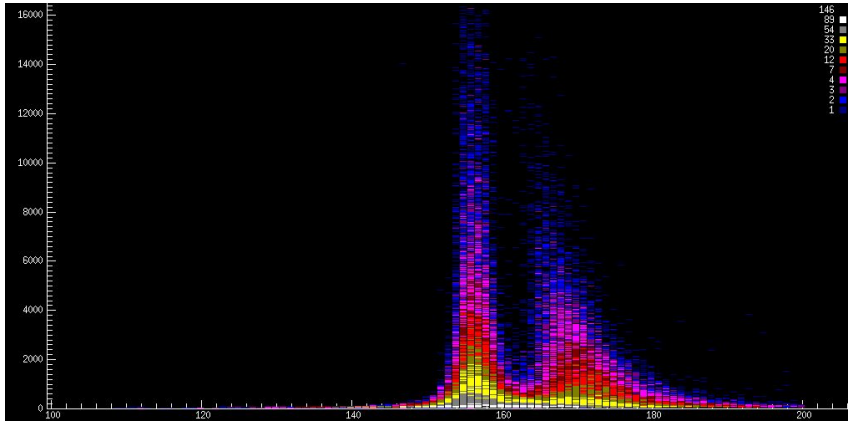
- Working through architecture
- Writing hardware interfaces
- Building out Kafka Servers
- Creating consumers/producers
- Building out backlog
- Actively recruiting!



Future Projects

- Clustering for particle identification
- Anomaly detection for ill traces
- Spark data processing
- Whatever else we dream up...





Quick Summary

- Data requirements are not unique.
- Physics has an unfortunate development cycle.
- It can be difficult to throw out everything and start anew.
- Dolosse aims to provide a strong open-source presence to physics DAQ and analysis.
- We'll use cutting-edge software and analysis techniques.


Thank you.