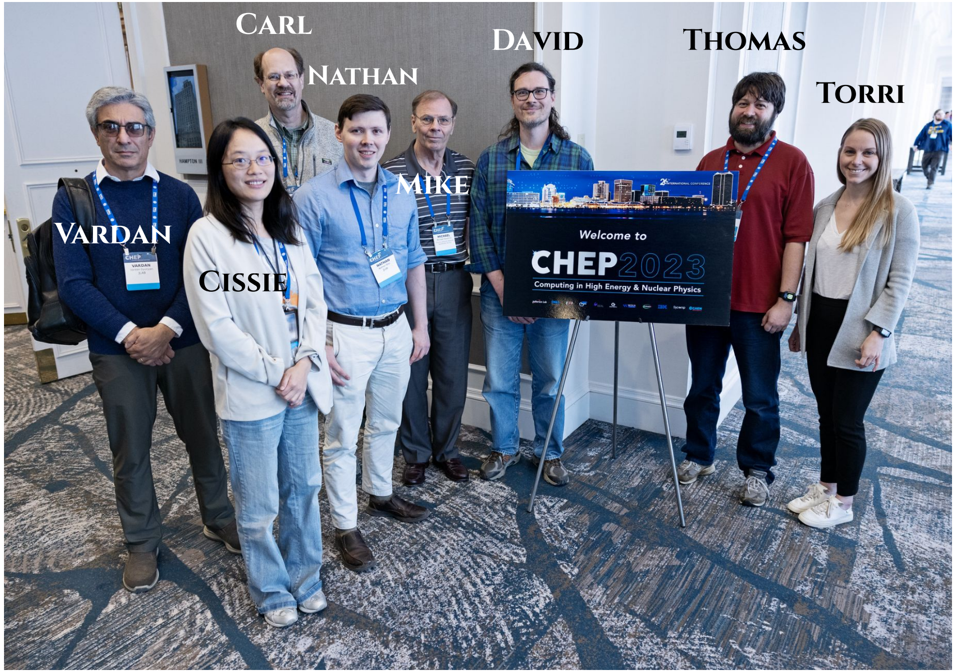
EPSCI Overview

David Lawrence

Thomas Jefferson National Accelerator Facility

Experimental Physics Software and Computing Infrastructure



EPSCI group priorities - epsciw

wiki.jlab.org/epsciwiki/index.php/EPSCI_group_priorities

Page | Discussion | Read | Edit | View history | More | Search

- Main page
- Recent changes
- Random page
- Help

Tools

- What links here
- Related changes
- Upload file
- Special pages
- Printable version
- Permanent link
- Page information
- Cite this page

EPSCI group priorities

Here are the mission statement and priorities of the EPSCI group. These are meant to help guide us in what projects to take on and which direction to take while developing them.

Mission Statement:

Identity, develop, implement, and maintain software and computing technologies in support of the Jefferson Lab Science Program.

The priorities listed below are in order. This is meant to help guide us in decision making when choices might otherwise be unclear.

EPSCI Priorities:

1. Support Experimental arm of Jefferson Lab 12 GeV Science Program through maintenance of existing software.
2. Investigate and develop new software and computing technologies to aid the 12 GeV science program at Jefferson Lab.
3. Promote the software and computing technologies developed at Jefferson Lab to benefit the science programs throughout the national lab system.

This page was last modified on 8 April 2020, at 19:22.

Privacy policy | About epsciwiki | Disclaimers

Powered By MediaWiki

Streaming Readout and Data-Stream Processing With ERSAP

Environment for Real-time Streaming, Acquisition and Processing

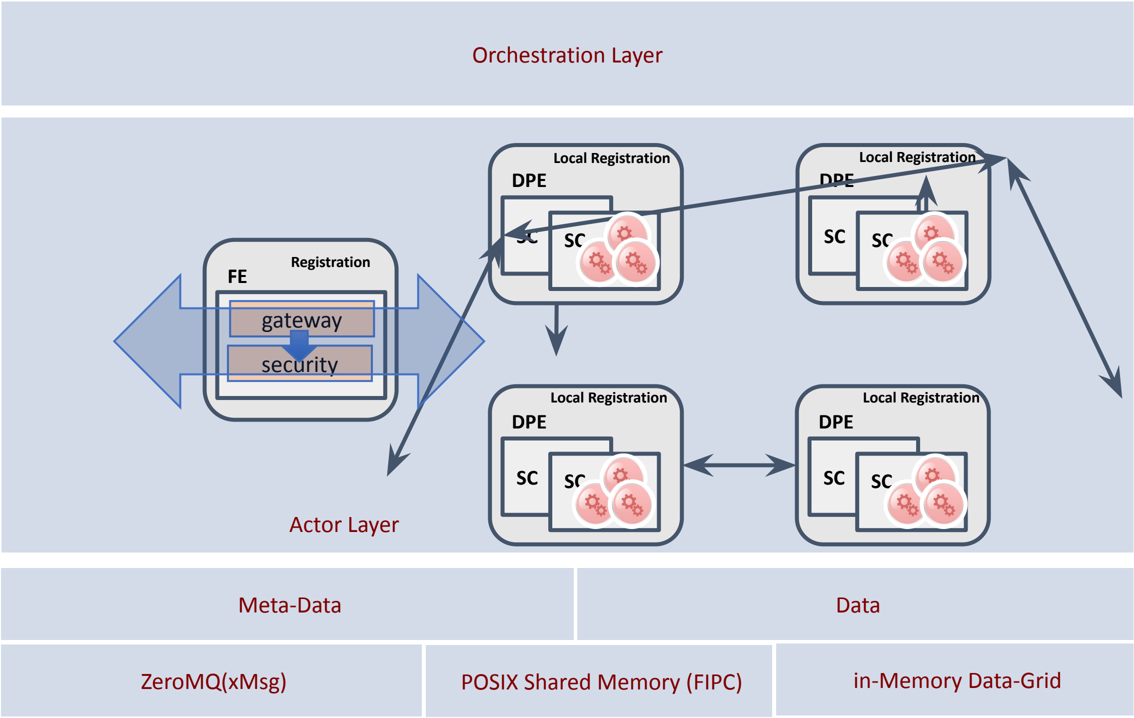
V. Gyurjyan, D. Abbott, M. Goodrich, G. Heyes, E. Jastrzembski, D. Lawrence, B. Raydo, C. Timmer

Πάντα ῥεῖ

Flow-Based Programming Paradigm

- Proposed in the late 60s by J. Paul Rodker Morrison
- “Assembly line” data processing
- Data flows through asynchronous, concurrent processors (“black box” actors)
- Actors communicate via data chunks (called information packets or data-quanta)
- Data-quanta are traveling across predefined connections (conveyor belts), where connections are specified externally to the processors.
- Data is pushed through actors, while actors are reacting on passing data quantum.
- Actors are performing independent, well-defined functions
- Simple reconfigure
- Fault tollerant

ERSAP 3-layer structure



Summary

- ERSAP is a software LEGO system
  - Encourages application design based on software artifacts (LEGO bricks)
    - Easier to understand and develop
    - Reduced develop-deploy-debug cycle
    - Easy to migrate to data
    - Scales independently
    - Independent optimizations
- Improves fault isolation
- Easy to embrace hardware as well as software heterogeneity.
- Eliminates long term commitment to a single technology stack.

Agile framework that makes easy software evolution over time!

EJFAT

ESnet-Jefferson Lab FPGA Accelerated Transport

Michael Goodrich[*], Carl Timmer[*], Vardan Gyurjyan[*],

David Lawrence[*], Graham Heyes[*],

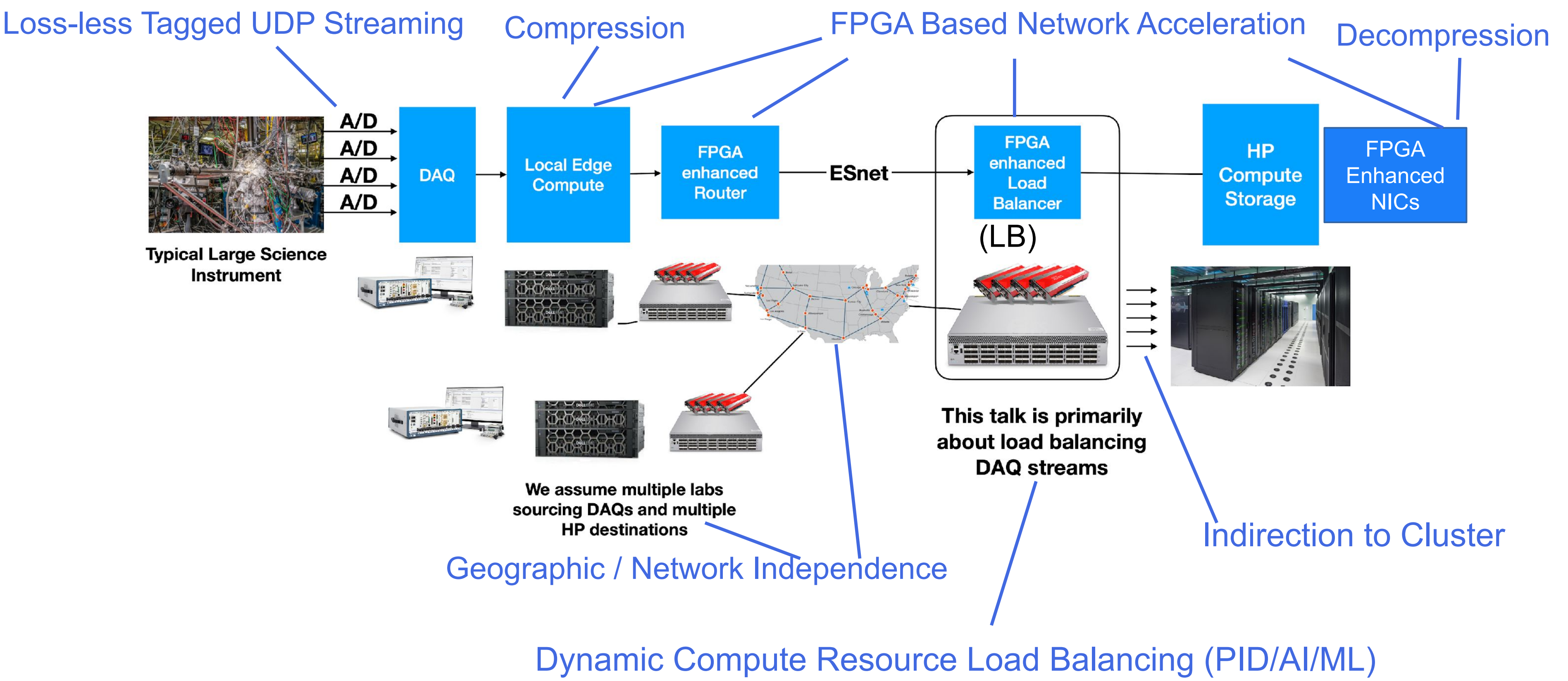
Yatish Kumar[+], Stacey Sheldon[+]

[*]JLAB
[+]ESnet

EPSCI

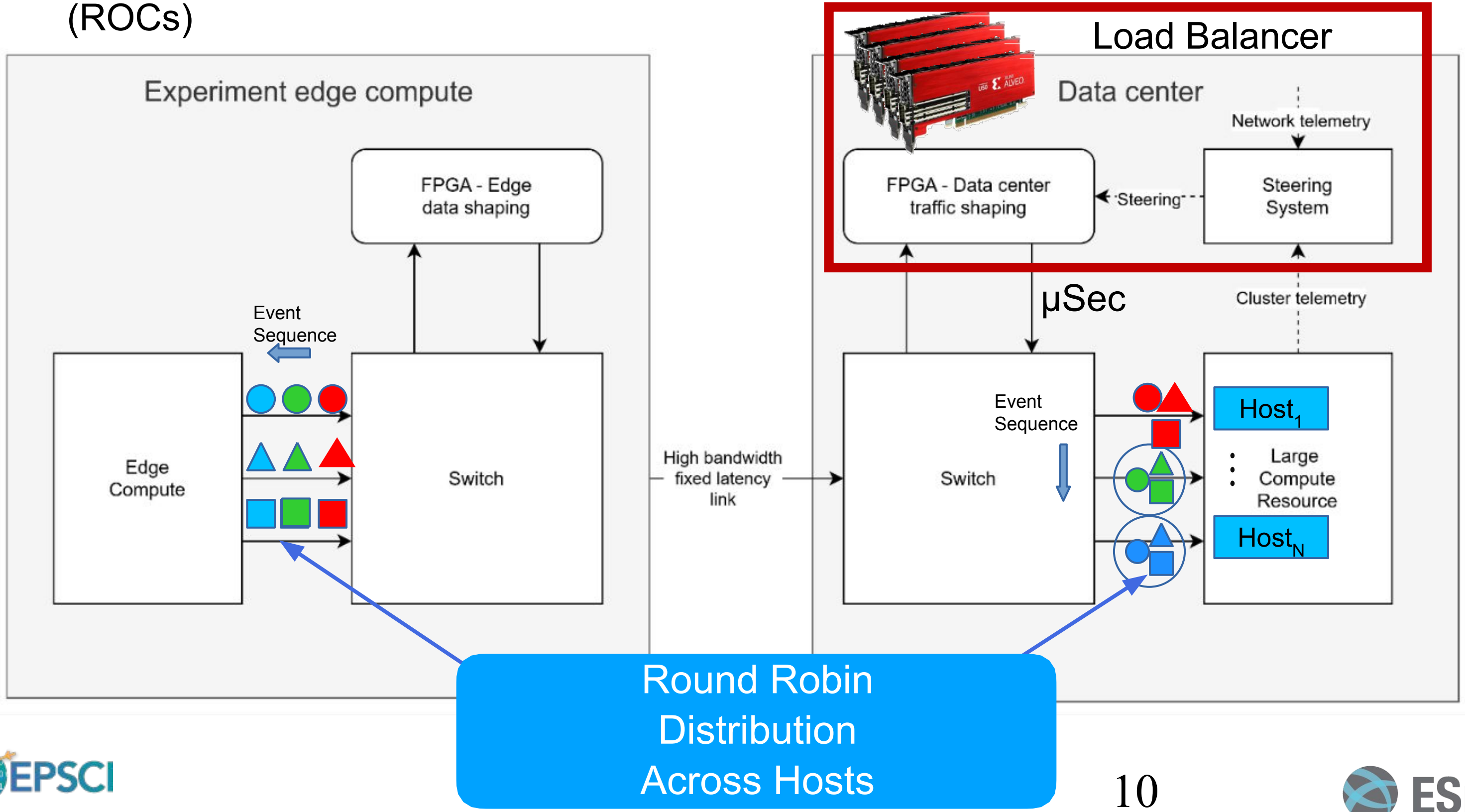
Jefferson Lab ESnet

EJFAT: Accelerated Edge to Core Workflow Steering

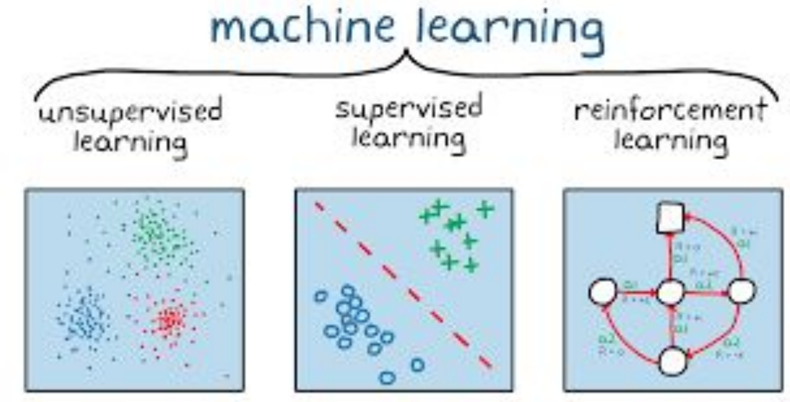


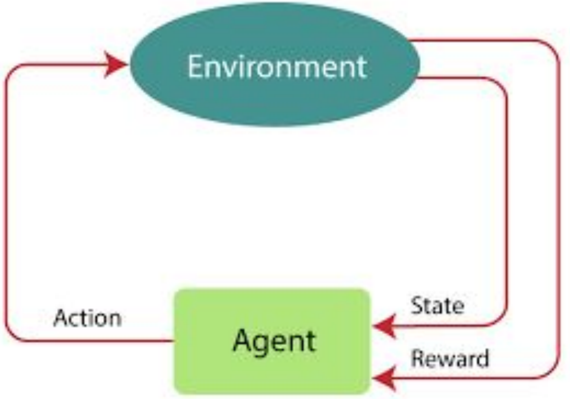
Steering/Scaling: Data Event -> Host Rotation

Colors → Events
Shapes → Channels (ROCs)

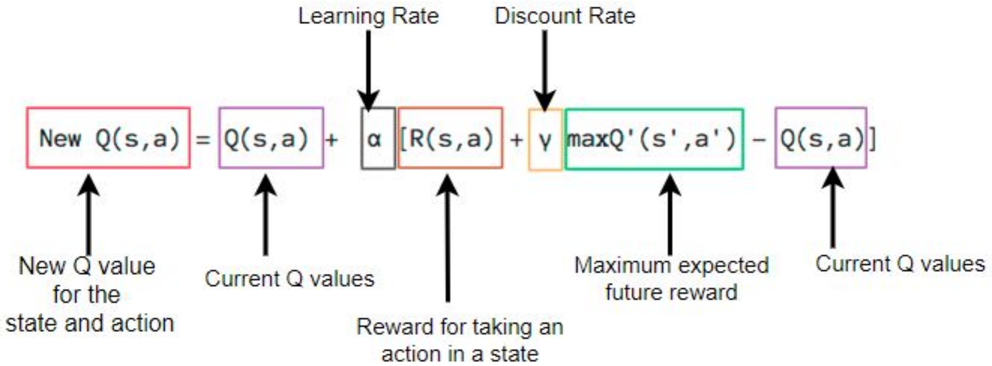


Schedule: Q (Reinforcement) Learning



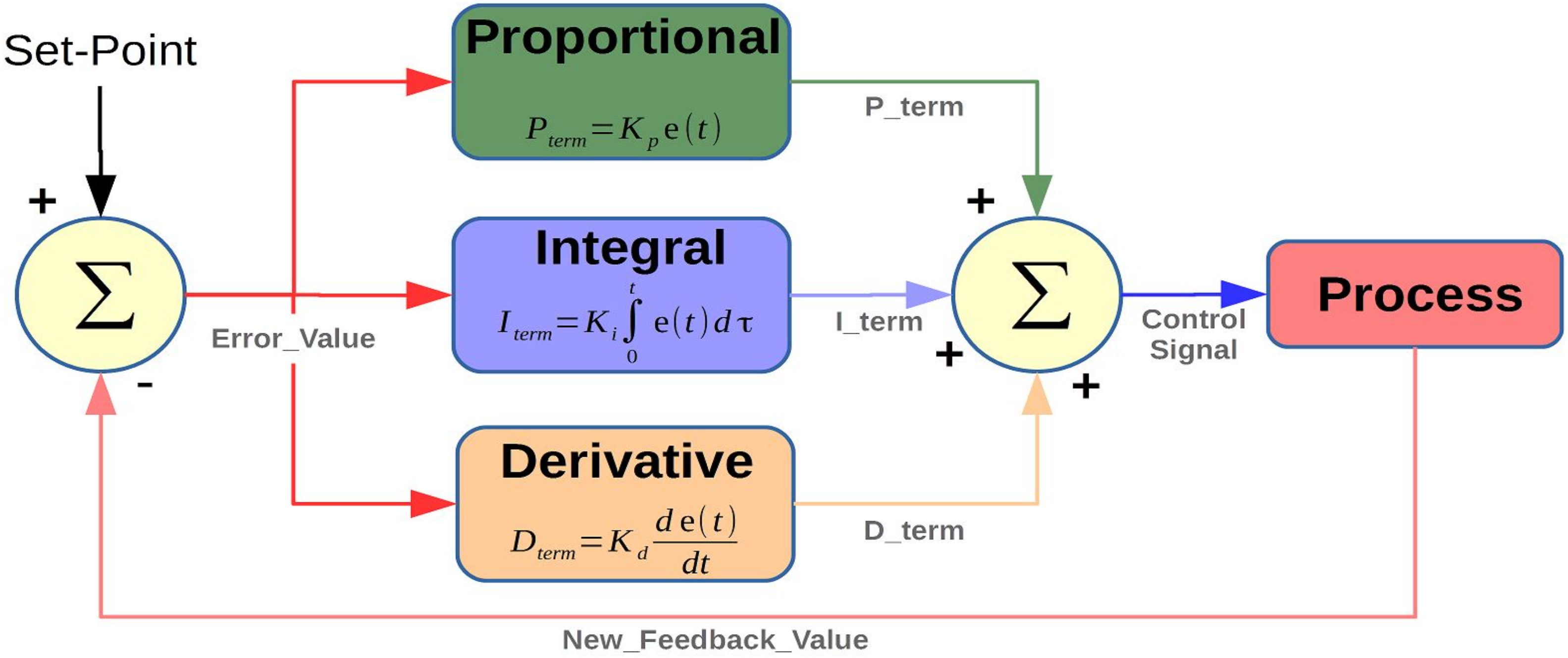


- Q Learning:
- Many Variants
- Exploration / Exploitation
- Exploration → Learning
- **Exploitation → Control**

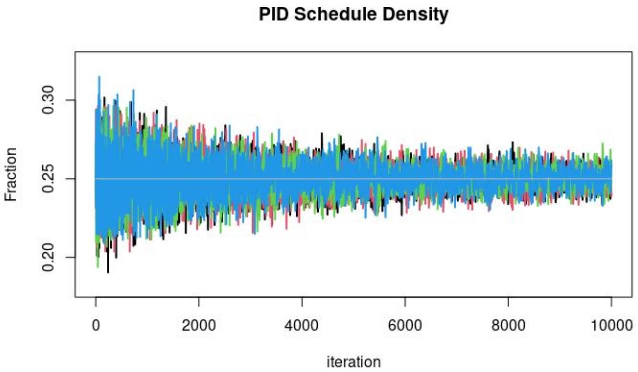


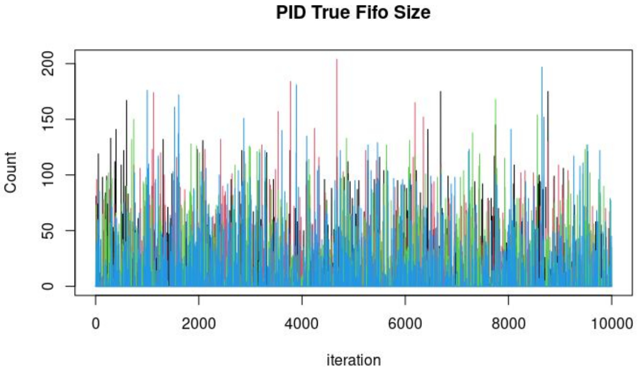
EPSCI ESnet Jefferson Lab

Schedule: PID Control

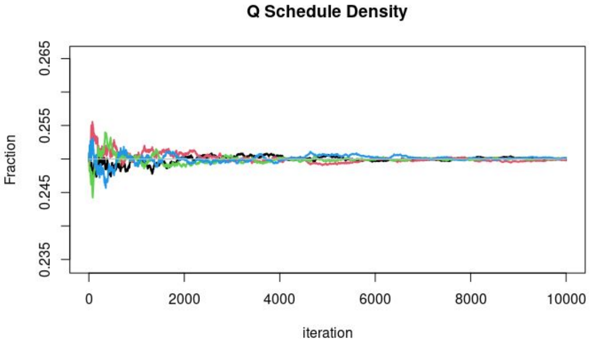


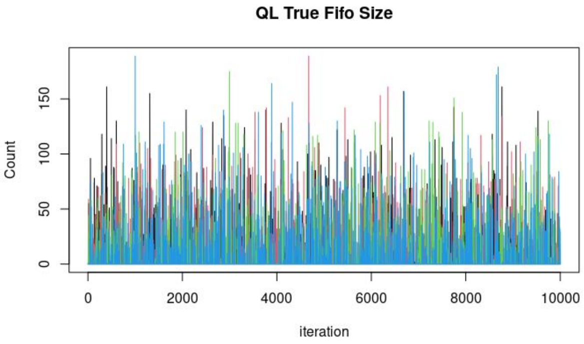
LB : Control Plane Simulation – Symmetric





Sym	N1	N2	N3	N4
EPR	500	500	500	500
FEPR	25%	25%	25%	25%
PID	**N1**	**N2**	**N3**	**N4**
SD	25%	25%	25%	25%
Db	400	400	400	400
FIFO	3.4	3.4	3.4	3.4
QL	**N1**	**N2**	**N3**	**N4**
SD	25%	25%	25%	25%
Db	400	400	400	400
FIFO	3.2	2.9	2.9	3.0

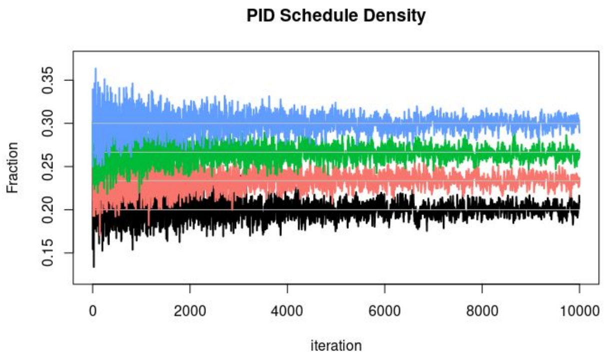


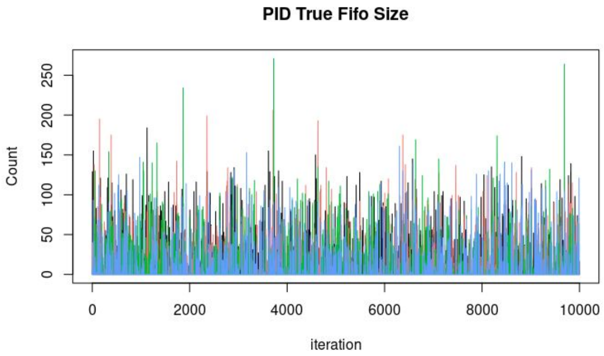


EPR=Event Processing Rate Db=Event Disbursement Rate SD=Schedule Density

EPSCI ESnet Jefferson Lab

LB : Control Plane Simulation – Asymmetric

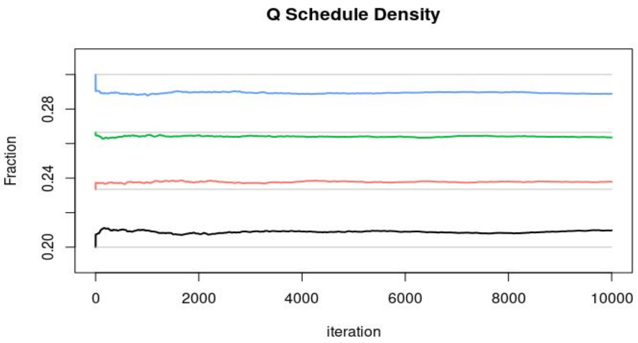


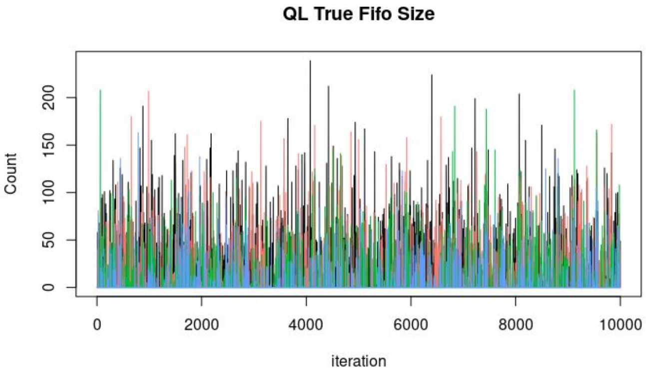


ASym	N1	N2	N3	N4
EPR	400	467	533	600
FEPR	20%	23%	27%	30%

PID	N1	N2	N3	N4
SD	20%	23%	27%	30%
Db	320	373	426	480
FIFO	4.6	3.7	3.1	2.7

QL	N1	N2	N3	N4
SD	21%	24%	26%	29%
Db	334	380	422	463
FIFO	7.4	3.9	2.2	1.2





EPR=Event Processing Rate

Db=Event Disbursement Rate

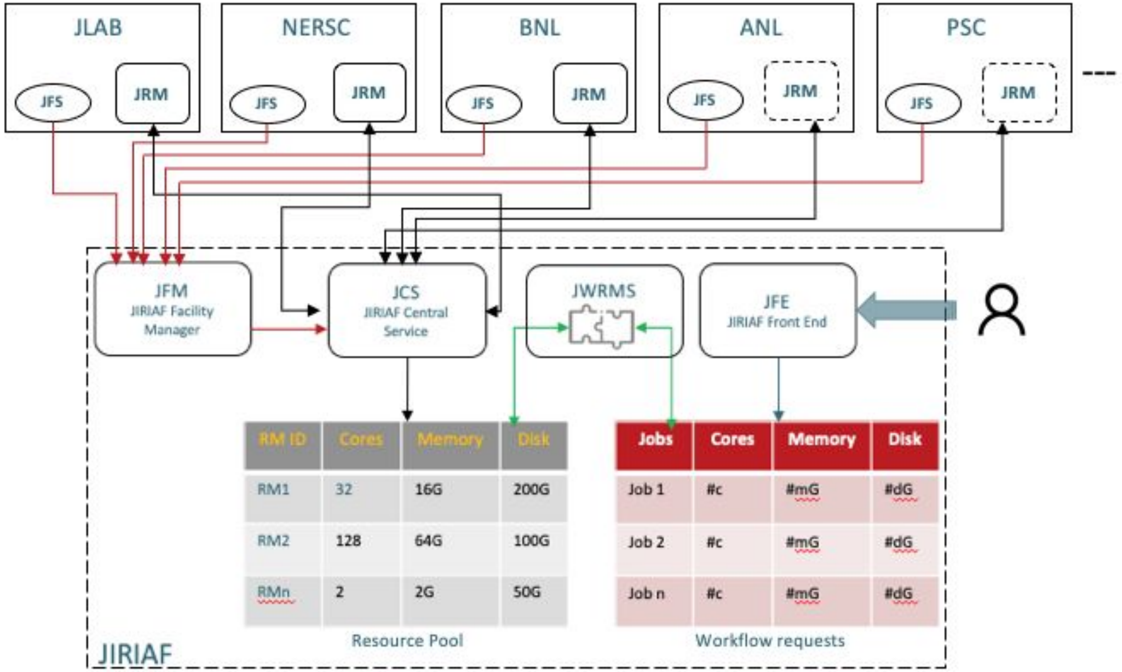
SD=Schedule Density

JIRIAF

JLAB Integrating Research Infrastructure Across Facilities

V. Gyurjyan, C. Larrieu, D. Lawrence, G. Heyes

Design Architecture



Controlling Carbon Footprint

- Improving operational efficiency of data centers can potentially delay compute resource expansion, controlling the carbon footprint of a computing facility.

Server Electricity Consumption

0.150 KW/Hour

Cooling Per Server

0.120 KW/Hour

Co_2 Emissions Per KWh (Pounds)

Coal	Natural Gas	Petroleum
2.21	0.92	2.11

→

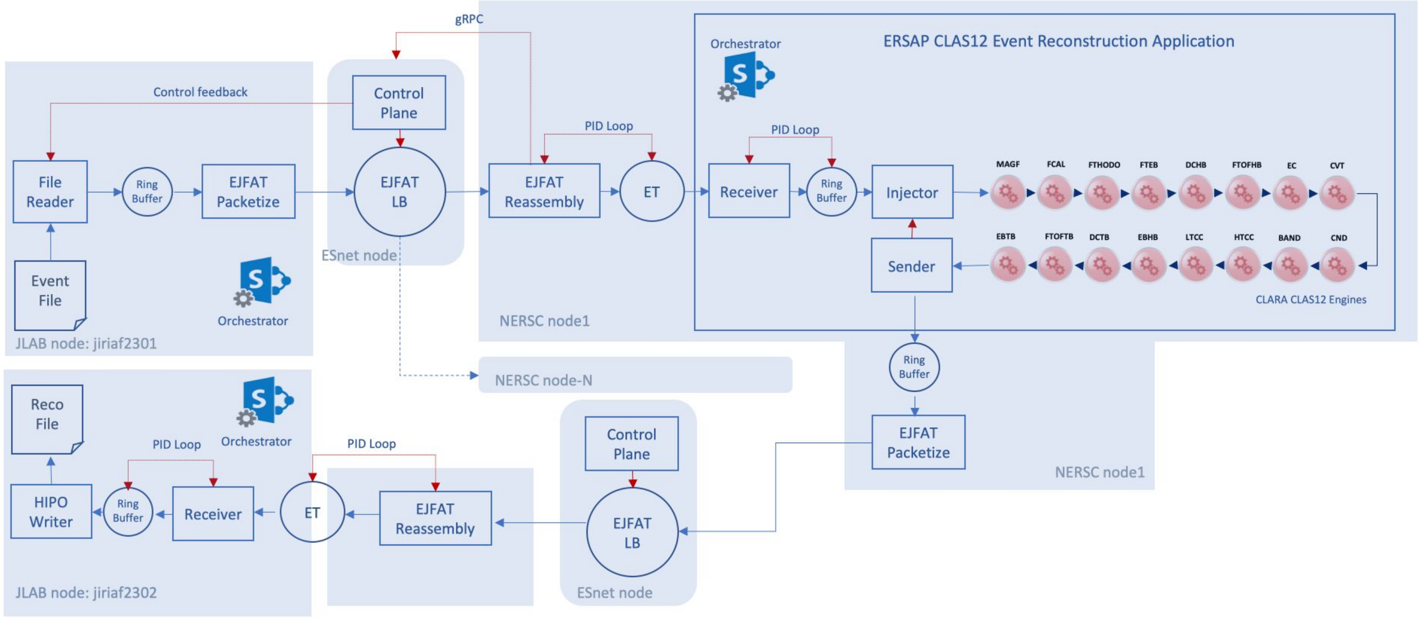
JLAB Farm
Average Yearly Carbon Footprint

Co_2 Emissions (Pounds)

Coal	Natural Gas	Petroleum
2.17M	0.9M	2.07M

Concept Validation Experiment

CLAS12 Data-Stream processing at NERSC. Stage 3



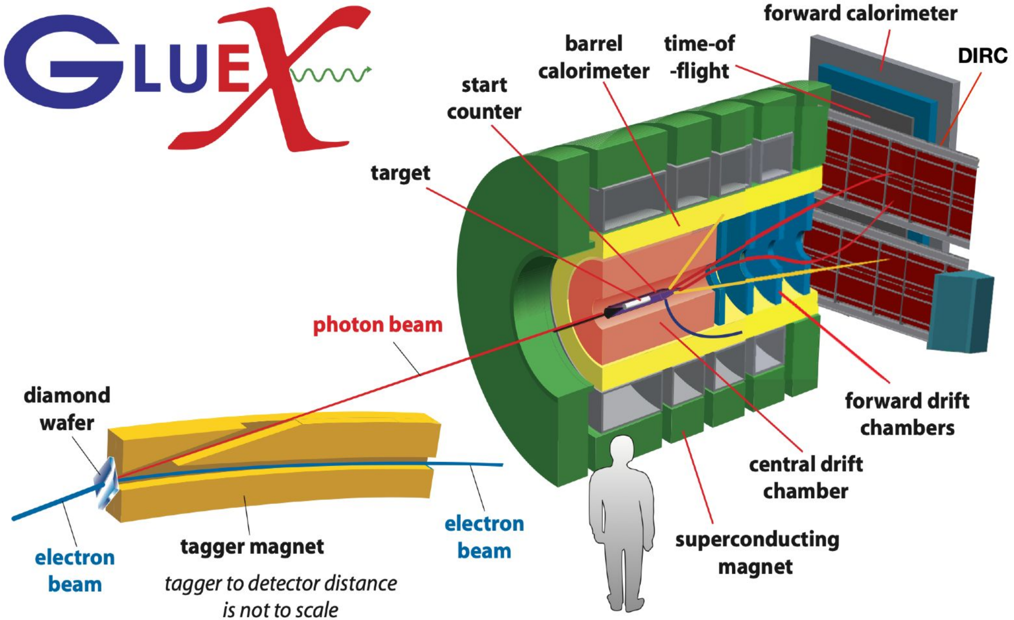
AI Driven Experiment Calibration and Control

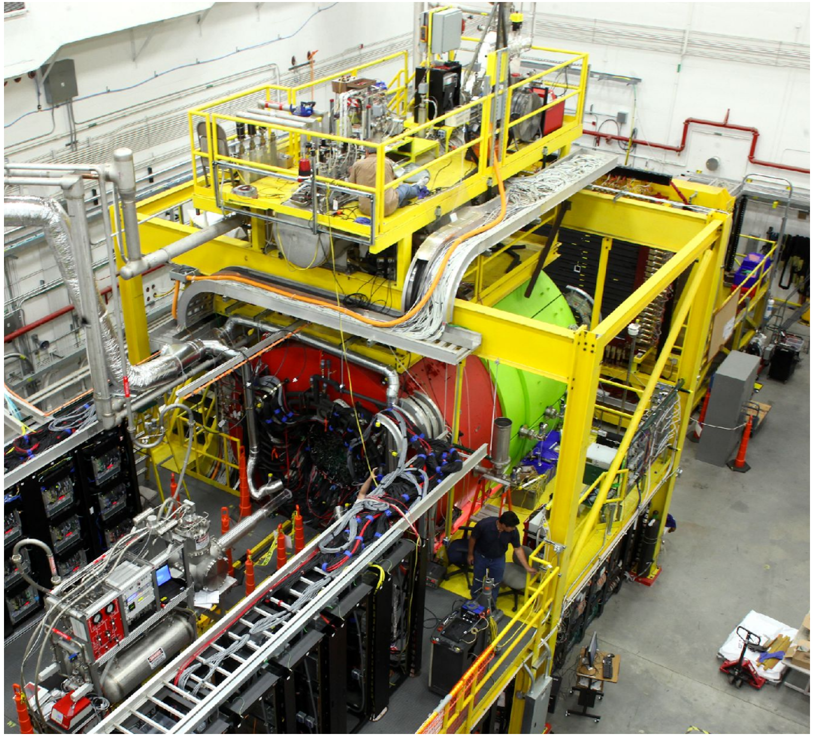
Thomas Britton
David Lawrence
Naomi Jarvis
Torri Jeske
Diana McSpadden
Nikhil Kalra

Carnegie Mellon University

The Gluonic Excitations Experiment: GlueX

GlueX detector located in Hall D at Jefferson Lab, VA

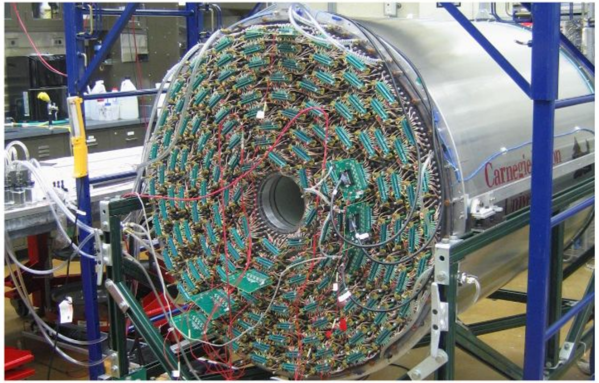




The GlueX Central Drift Chamber

Used to detect and track charged particles with momenta p > 0.25 GeV/c

- 1.5 m long x 1.2 m diameter cylinder
- 3522 anode wires at 2125 V inside 1.6 cm diameter straws
- 50:50 Ar:CO_2 gas mixture
- **Requires two calibrations: chamber gain and time-to-distance**

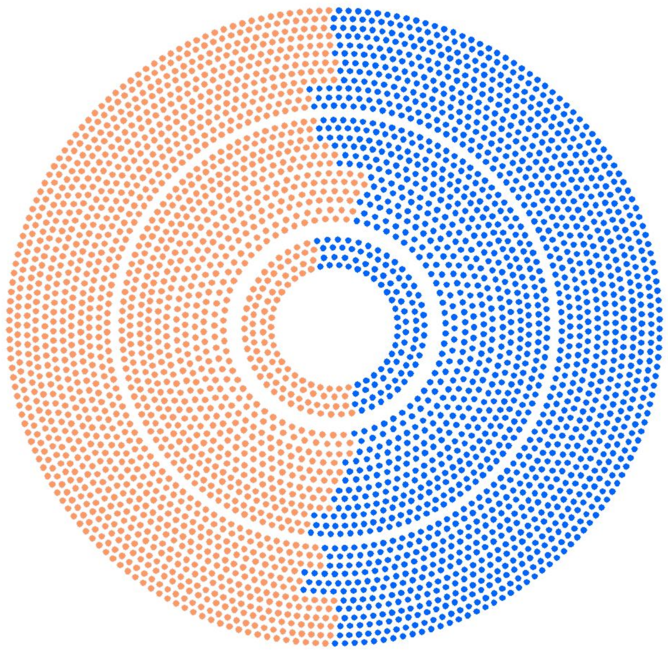


Deployment 1 – Cosmic Ray Test

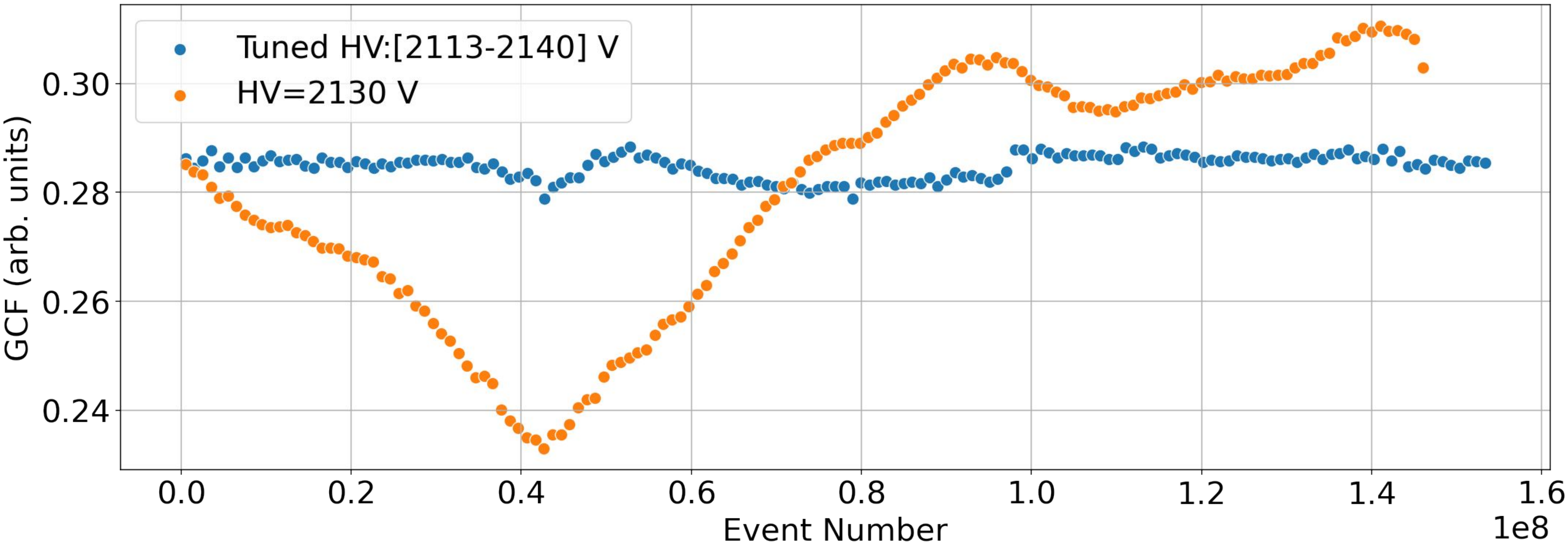
- Sorted high voltage boards (HVB) into two groups:
  - AI Tuned
  - Constant: 2130 V (5V higher than normal to compensate for no beam)

- ML
  - Update every 5 minutes
  - Completely autonomous

- Should see the gain stabilized for the Tuned group

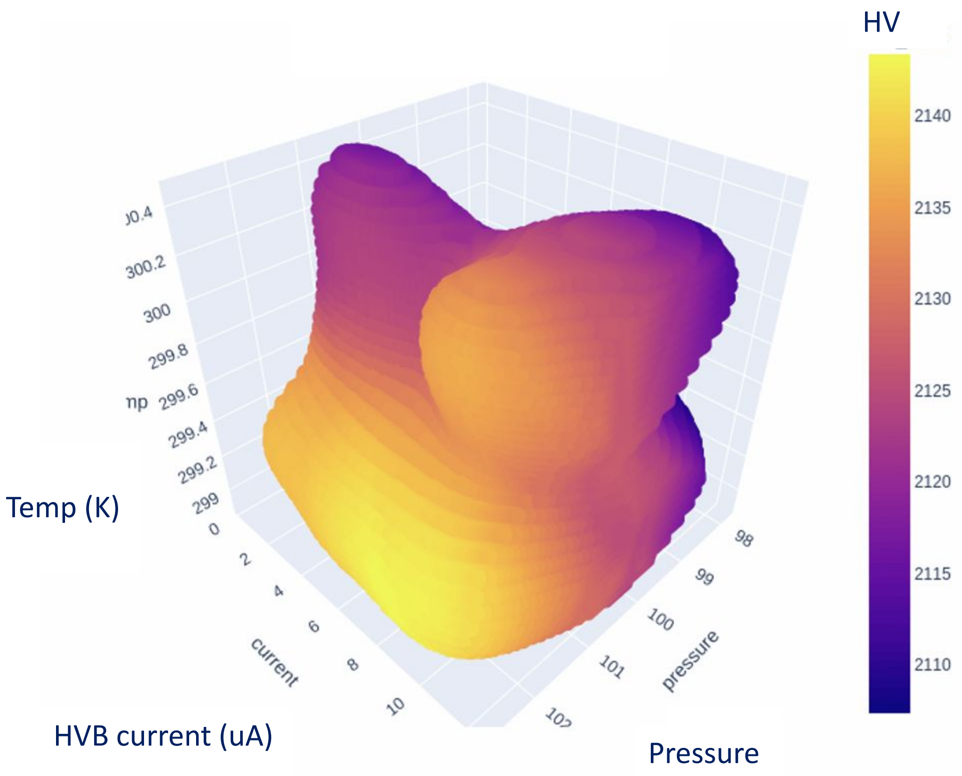


Deployment 1 – Cosmic Ray Test



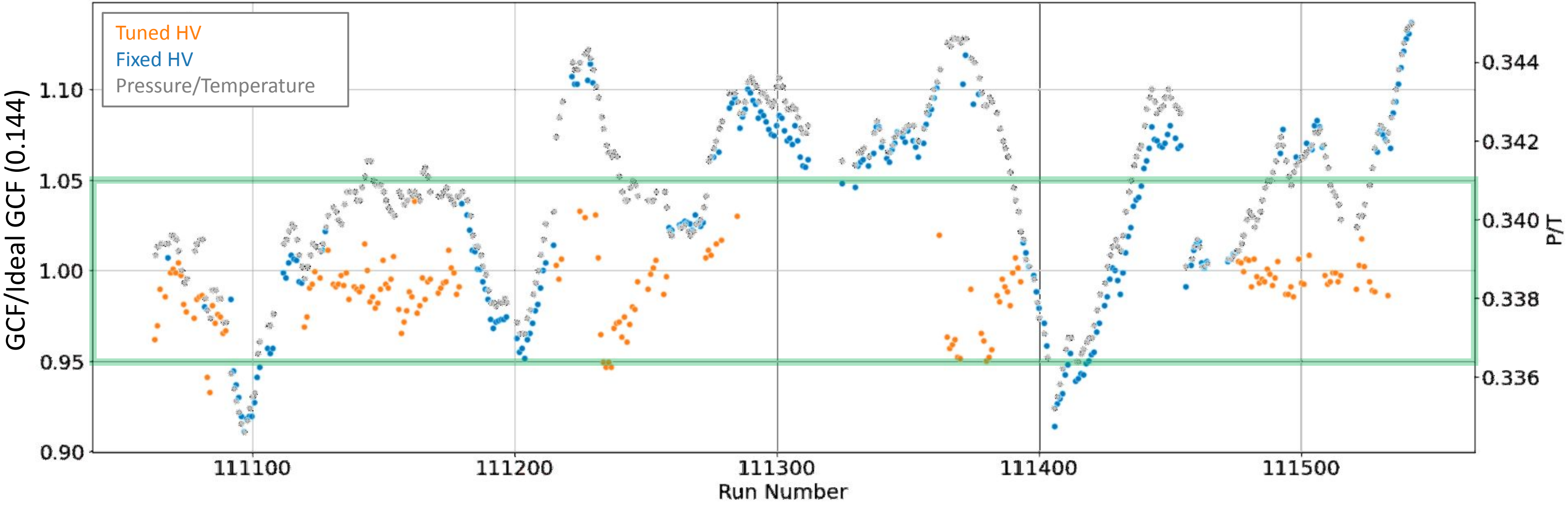
Deployment 2 – Charged Pion Polarizability May-June 2022

- RoboCDC used automatically at the start of each 2h run
- Use recommended HV if std <= 3% ideal GCF
- Otherwise, use the closest 'confident' HV in Euclidean distance on the uncertainty mesh
- Reverted to 2125V for empty target runs
- Low stakes - CDC not critical for CPP run period
- CPP: unusual running conditions
    - Different target in different location
    - Low beam current



Deployment 3 – PrimEx-η June-Dec 2022

- GCF obtained from dE/dx after the run
- Preliminary results show GCF predominantly within 5% of ideal value for runs with tuned HV
- Plot of GCF/ideal for tuned HV and fixed HV also shows pressure/temperature



Calibration of the Forward Calorimeter

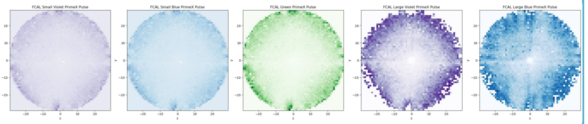
Diana McSpadden, Cullan Bedwell, Abhijeet Chawhan, Julie Crowe

University of Virginia

Traditional Calibration:

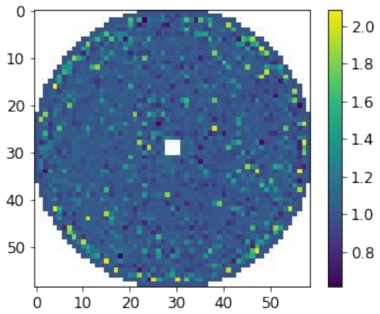
- iterative over π^0s
- Requires particle reconstruction
- Statistics sometimes difficult

Can we use the LED monitoring system and Machine Learning?

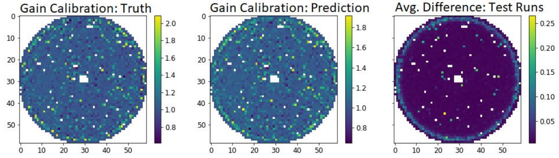


=

Gain calibration values

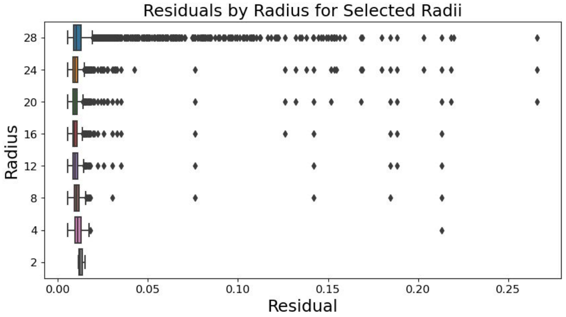


Can ML learn traditional calibrations?



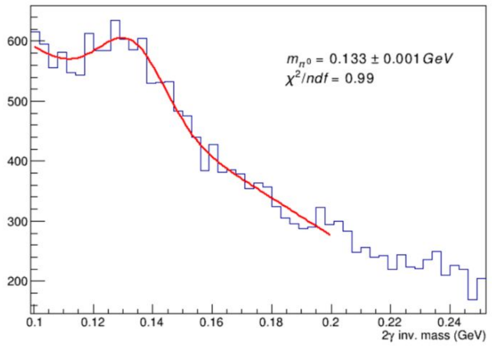
Average results over 5-fold cross validation

dataset	fold idx	average residual ↓	mape ↓	mse ↓
unmasked	average	0.258	23.848	5.183
masked	**average**	**0.027**	**2.370**	**0.004**



Initial Physics Comparison

- Does prediction accuracy result in good physics results?
- We have an initial π^0 analysis
  - Single run, entire FCAL

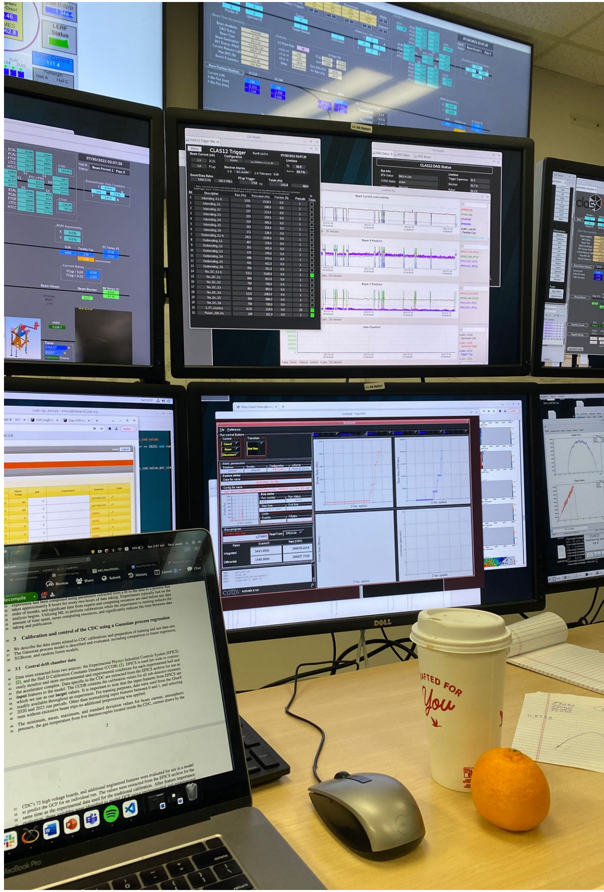


- π^0_{PDG} mass: **134.98 MeV**
- Using our calibrations: **133.31 MeV**

HYDRA

COMPUTER VISION FOR REAL TIME DATA QUALITY MONITORING

CHEP 2023



TORRI JESKE
roark@jlab.org

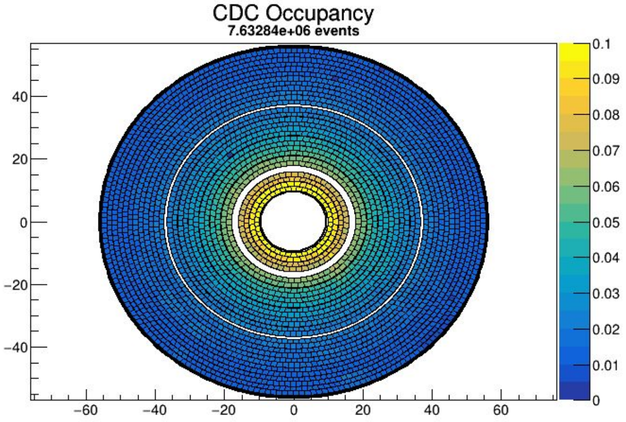
THOMAS BRITTON
tbritton@jlab.org

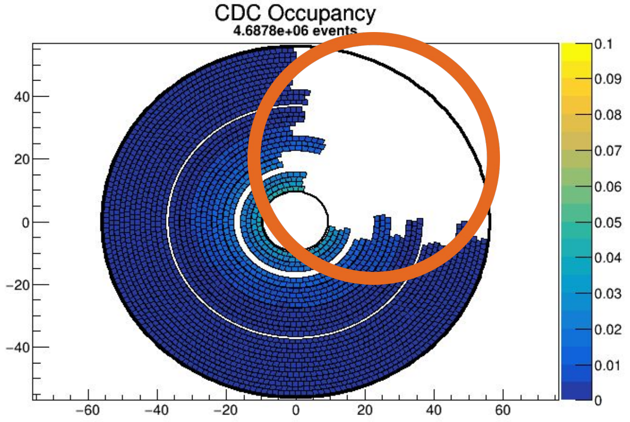
DAVID LAWRENCE
davidl@jlab.org

KISHANSINGH RAJPUT

Human Classification

What influenced your decision?





1/ Probably good

We expect to see higher occupancies closer to the beam line. This appears to look consistent with other monitoring histograms.

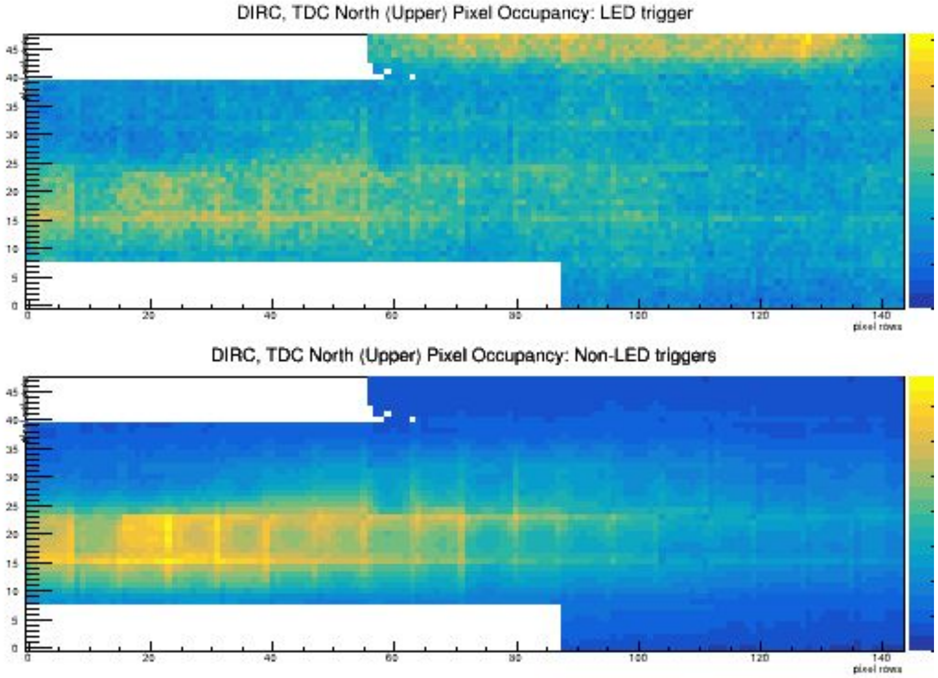
2/ Probably NOT good

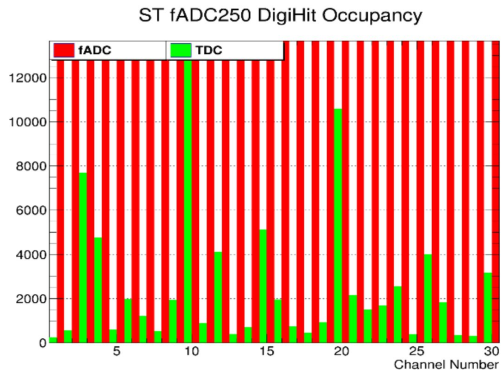
Turns out this giant hole is *fine* if you temporarily donate some electronics to another detector.

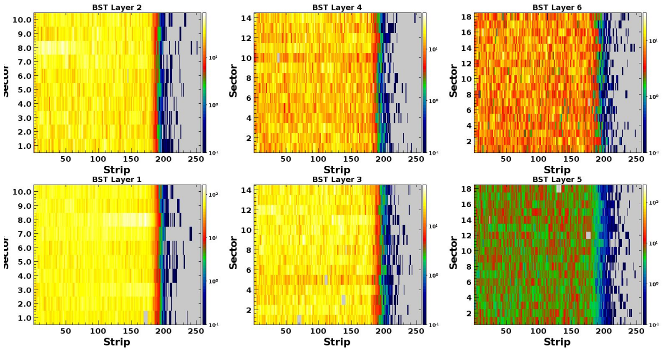
This is an obvious example!

Less Obvious Examples

It's hard to tell right away if an image is bad!







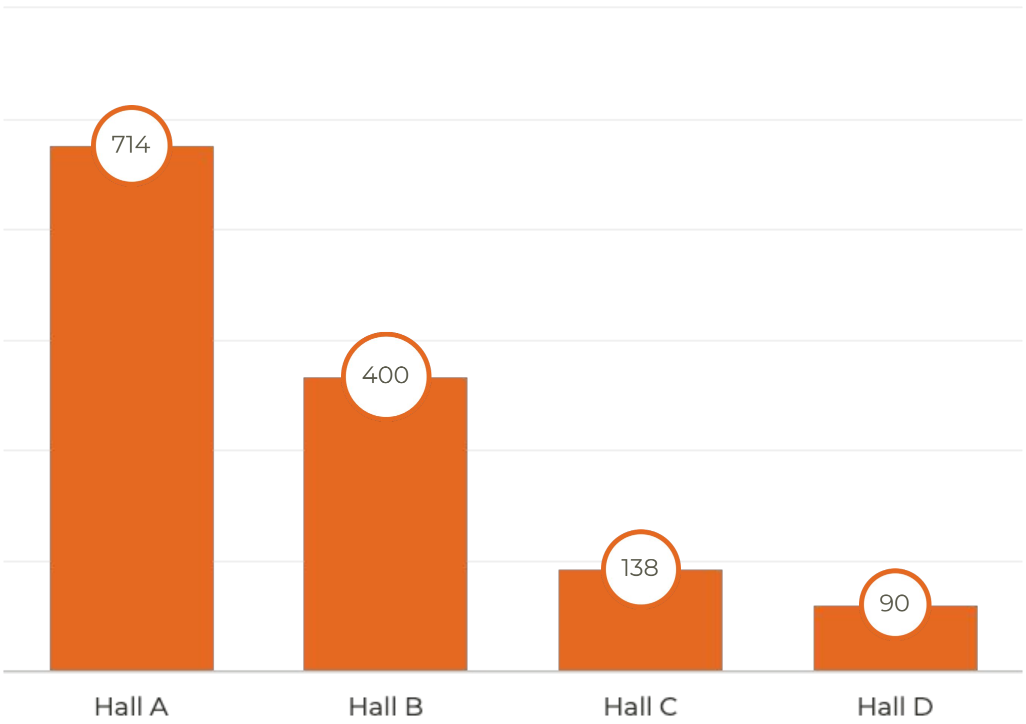
Online monitoring is **tedious.**

Varying levels of expertise

Inconsistent monitoring

Multiple plots per detector system

Probably too many plots to look at



Approximate **number of individual histograms per experiment per run**, monitored by the shift crew for each experimental hall.

Can we use computer vision to mimic the Shift Crew?

Reduce inconsistencies inherent with human-based monitoring.

Correctly classify monitoring images quickly and more frequently than humans

Explainable predictions: can the model tell us what influenced it's decision?

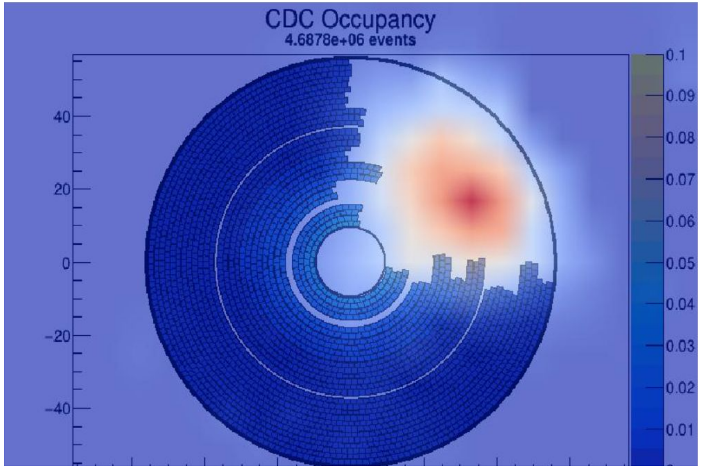
Shift Crew 🤝 AI

PREDICTED: [('N02112137', 'CHOW', 4.611241), ('N02124075', 'EGYPTIAN_CAT', 4.3817368)]
HTTPS://KERAS.IO/EXAMPLES/VISION/GRAD_CAM/

Why did the model make that prediction?

- Deep neural networks have great performance but are hard to interpret.
- **Interpretability matters**, especially when implementing smart systems into our typical work flows.
- Visual explanations with Gradient-weighted Class Activation Maps (GradCAM)

  Uses the gradients of any target concept flowing into the final convolutional layer to produce a map that highlights important regions in the image for the prediction

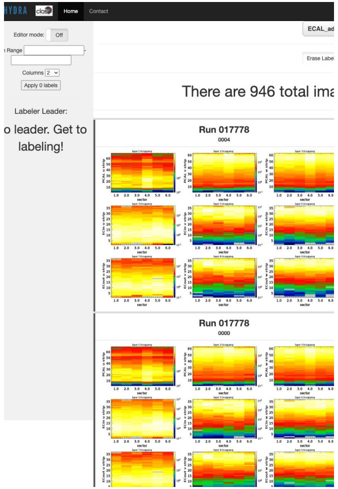


We are usually much more interested in what makes an image bad than good.

Reference: https://arxiv.org/pdf/1610.02391.pdf

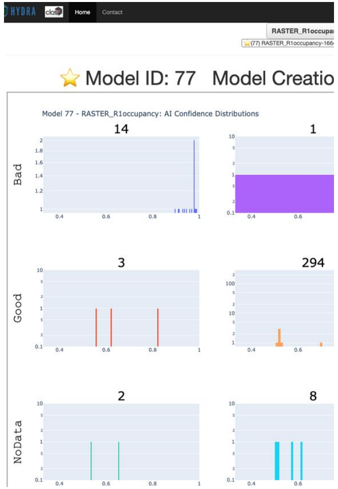
HYDRA: Front End

Web based for user convenience.



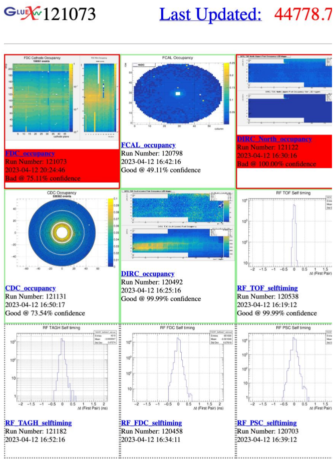
Data Labeler

Efficiently label hundreds (thousands) of images



Library

Contains enhanced confusion matrix, thresholds, active model designations

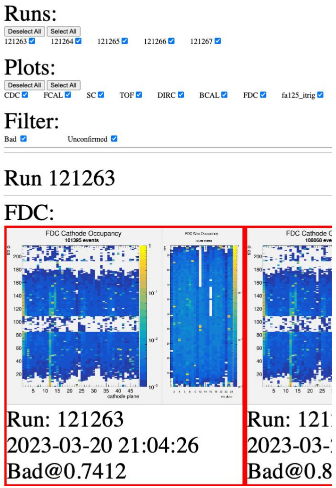


Run

See near real time predictions

Grafana

Dashboard displays all predictions over time



Log

Display concerning plots sorted by detector from previous day

PHASM: Parallel Hardware viA Surrogate Models

ACAT 2022

Nathan Brei, Xinxin Mei, David Lawrence

Jefferson Lab

October 26, 2022

What is PHASM?

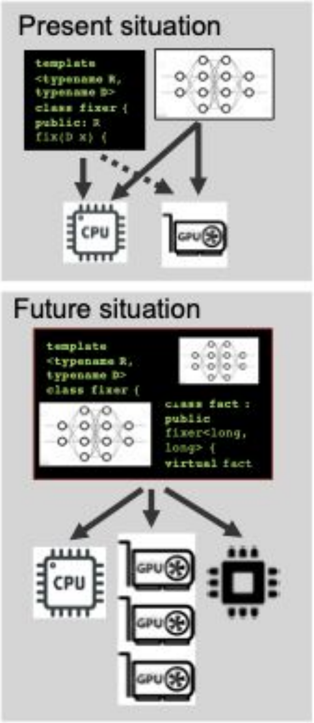
- LDRD project at Jefferson Lab
- 1 year old, 2-3 people
- Proof of concept

Basic Idea

Make it as easy as possible to train a neural net surrogate model to mimic and replace an arbitrary piece of existing numerical code. Systematize and formalize the process from analysis to deployment.

Perspective shift

A neural net surrogate model of an algorithm is a *transformation* of that algorithm. Eventually, classical numerical methods and their data-driven analogues will be understood under a unified theory.



Surrogate API: Simple example

```
double f(double x, double y, double z) {
    return 3*x*x + 2*y + z;
}

phasm::Surrogate f_surrogate = phasm::SurrogateBuilder()
        .set_model(std::make_shared<phasm::FeedForwardModel>())
        .local_primitive<double>("x", phasm::IN)
        .local_primitive<double>("y", phasm::IN)
        .local_primitive<double>("z", phasm::IN)
        .local_primitive<double>("retval", phasm::OUT)
        .finish();

double f_wrapper(double x, double y, double z) {
    double res = 0.0;
    f_surrogate.bind_original_function([&](){res = f(x,y,z);})
               .bind_all_callsite_vars(&x, &y, &z)
               .call();  // $PHASM_CALL_MODE controls this
}
```

Optics

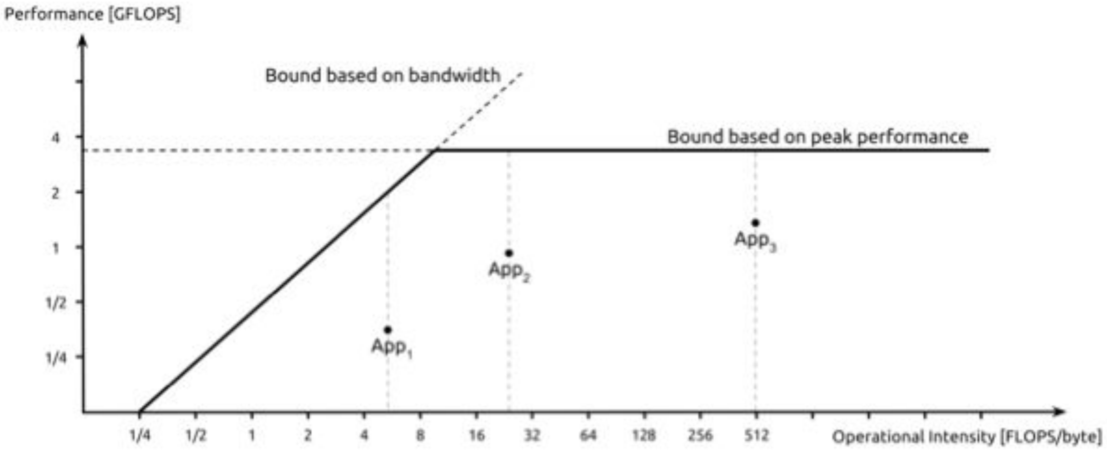
- *Profunctor optics* are used to perform simple and effective two-way data transformation between C++ types and tensors.
- Profunctor optics are composable 'getter and setter' objects that are generalized to support type conversions, missing, and multiple values.
- Correctly handles nested datatypes, e.g. an array containing a struct containing an array of doubles translates to a 2D tensor of doubles, without writing any loops!
- Currently PHASM supports primitives, fixed-length arrays, variable-length arrays, structures, and unions.
- In progress are STL collections and objects with invariants

Surrogate API: Nested data example

```
struct MyStruct { double x, y; };
double sum = 0;
void sum_all_x(MyStruct* s) {
    for (int i=0; i<4; ++i) { sum += s[i]->x; }
}
phasm::Surrogate sum_all_x_surrogate = phasm::SurrogateBuilder()
    .set_model(std::make_shared<phasm::TorchScriptModel>("model.pt"))
    .local<MyStruct*>("s")
        .array<MyStruct>(4)
        .accessor<double>([](MyStruct *s) { return &(s->x); })
        .primitive<double>("x", phasm::IN)
        .end()
    .global_primitive<int>("z", &z, phasm::INOUT)
    .finish();

void sum_all_x_wrapper(MyStruct* s) {
    sum_all_x_surrogate.bind_original_function([&](){f(s);})
                .bind_all_callsite_vars(s)
                .call();  // $PHASM_CALL_MODE controls this
}
```

Performance analysis: Roofline model



Experimental Physics Software and Computing Infrastructure

EPSCI is leading several projects:

- ERSAP — modular streaming readout platform
- EJFAT — smart, dynamic traffic shaping
- JIRIAF — HPC/HTC resource optimization and rollover
- AIEC — AI for Experimental Controls
- Hydra — AI Data Quality Monitoring
- PHASM — surrogate model integration tools
- JANA2 — multi-threaded reconstruction framework

Several languages are used

- C++
- Python
- Java