NOvA Computing & Needs for Concurrency

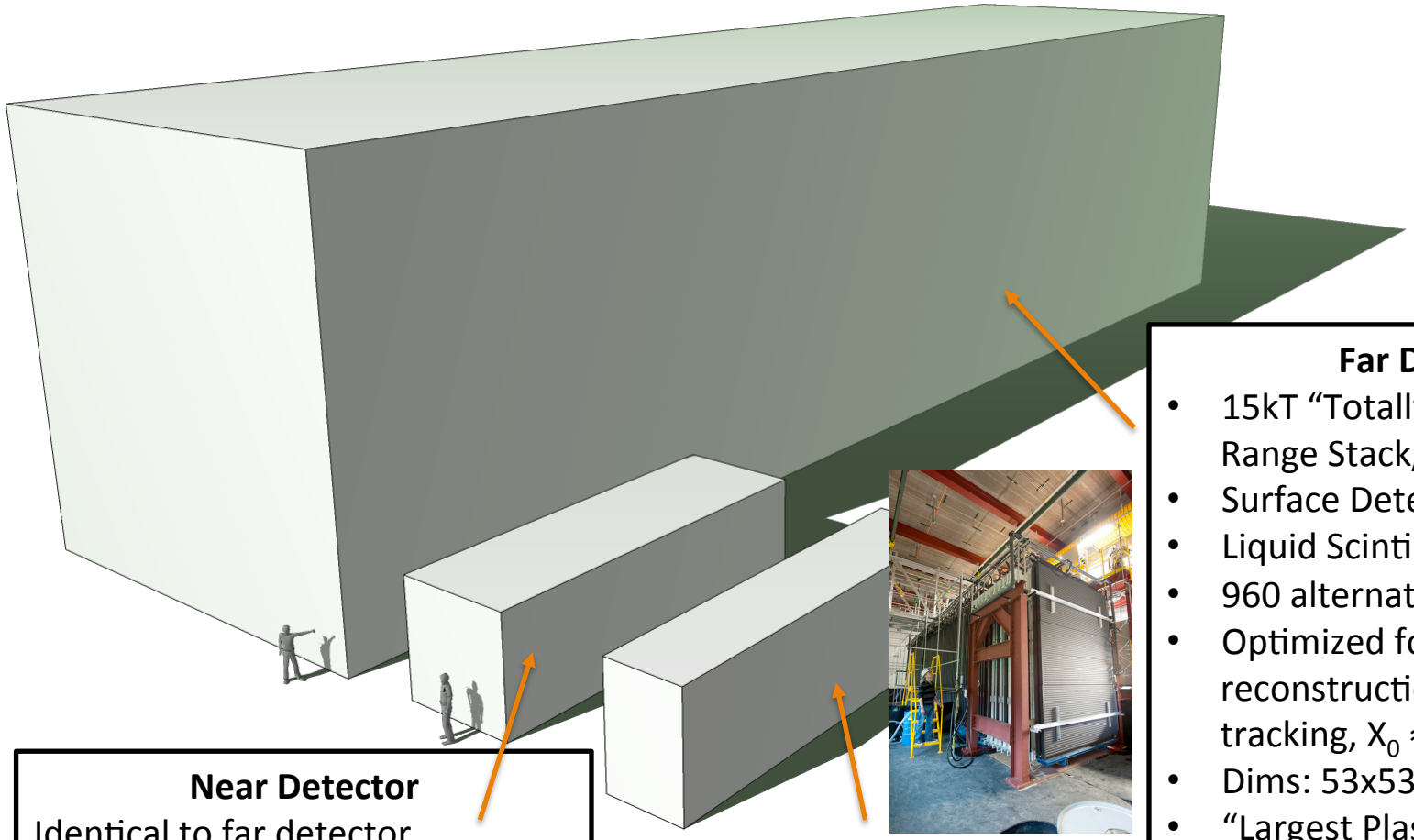
Andrew Norman, Fermilab

Annual Concurrency Forum 2013

NOvA

- NOvA is a program to investigate the properties of neutrinos
- It includes:
  - Doubling of the Fermilab NuMI beam power to 700 kW
  - An 15kTon totally active *surface* detector, 14 mrad off axis at 810km (first oscillation max for 2GeV)
  - A 220 Ton totally active near detector
- It has been optimized as a segmented low Z calorimeter/range stack to:
  - Reconstruct EM showers
  - Measure muon track
  - Detect nuclear recoils and interaction vertices
- It presents a challenge for modern computing and data acquisition

The 3 NOvA Detectors



Near Detector

Identical to far detector
1:4 scale size
Underground Detector
Optimized for NuMI cavern rates
-- 4x sampling rate electronics

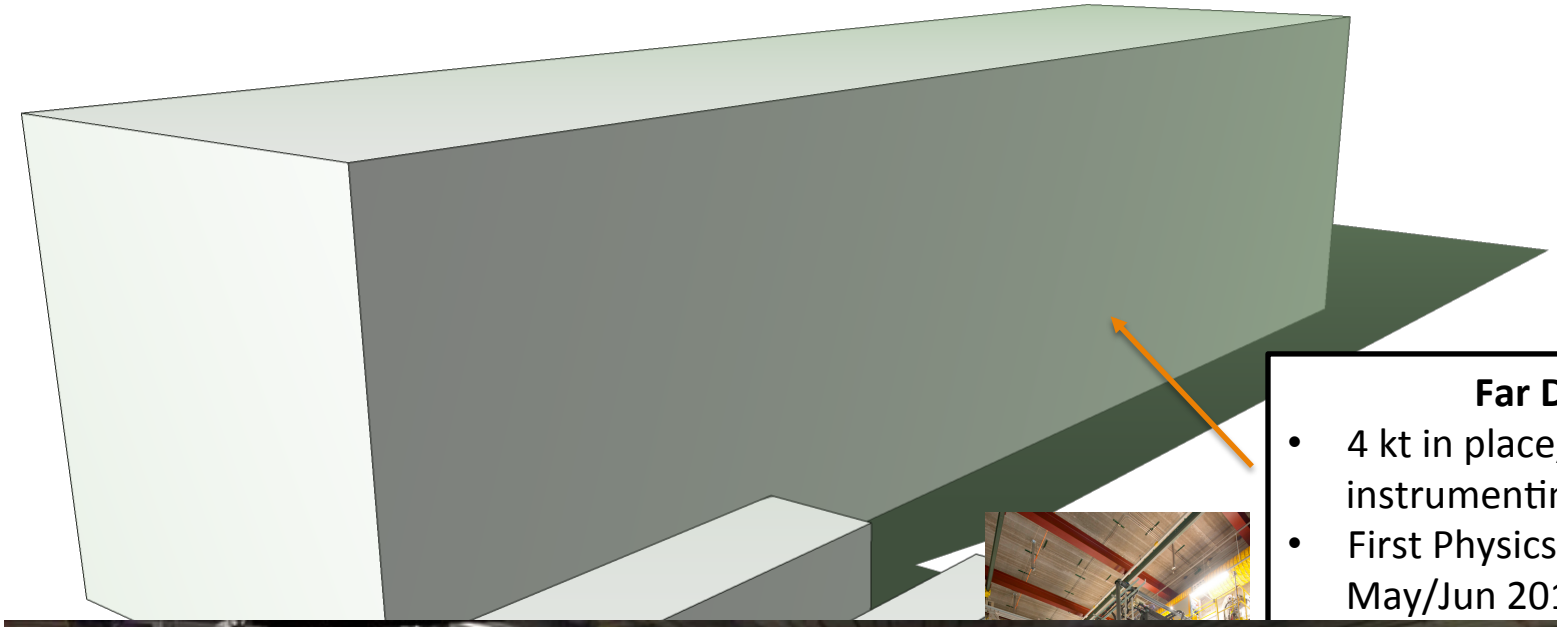
Near Det. Protype

In operation 2010-Present on surface at FNAL in NuMI and Booster beam line

Far Detector

- 15kT "Totally Active", Low Z, Range Stack/Calorimeter
- Surface Detector
- Liquid Scintillator filled PVC
- 960 alternating X-Y planes
- Optimized for EM shower reconstruction & muon tracking, X_0 ≈40cm, R_m≈11cm
- Dims: 53x53x180ft
- "Largest Plastic Structure built by man"
- Began construction May 2012
- First operation est. Feb. 2013 (cosmics)

The 3 NOvA Detectors



Far Detector

- 4 kt in place, in process of instrumenting
- First Physics Data May/Jun 2013

FAR DETECTOR (FEB 2013) ≈4 KT

Nova Computing Overview

- Realtime data processing (trigger)
  - Continuous data stream off detector (peak 4.3 GB/s)
  - 3200 core L3 style cluster
  - ARTDAQ framework used for realtime analysis/trigger processing
  - Maximum latency to trigger decision ≈20 s

Time critical, requires vectorization, multi-core, many-core to meet trigger goals

- Offline data processing
  - Need to process ≈ .75 PB/yr
  - Use the ART framework

Embarrassingly parallel at event level

- Monte Carlo generation
  - Want ≈ 10^9 evt/yr (30x beam data)
  - Use Genie, Geant4 with ART framework

Current model uses job level parallelism and grid computing

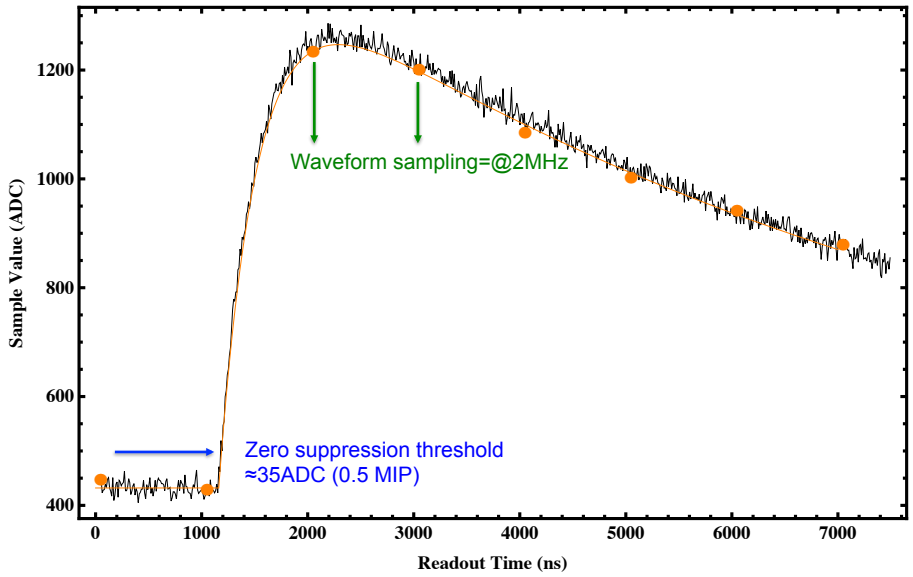
NOvA Examples:

- Realtime Hit unpacking
  - Vectorization
- Pattern Recognition (Hough Transforms)
  - Ideal parallelism mappable to many-cores
- Event Classification (Library event matching)
  - Shared resources in a parallel environment

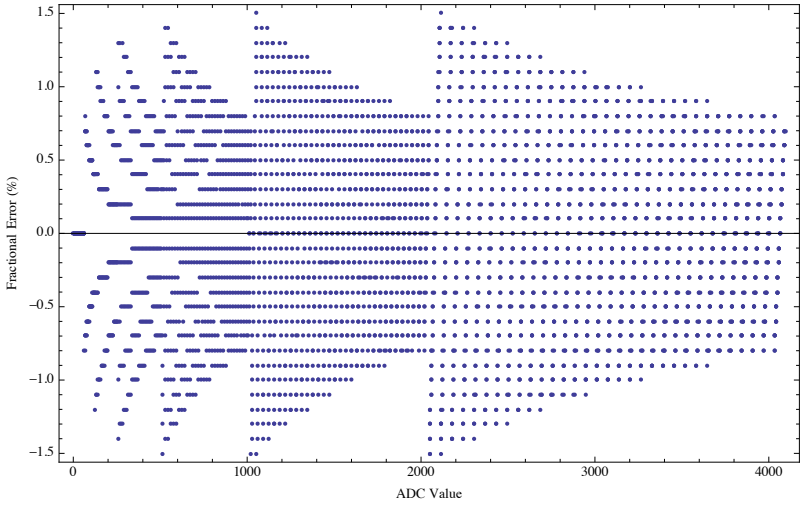
- Timing & Pulse height extraction on waveforms
  - "Fitting" w/ GPUs

Readout Decoding

- To save bandwidth hit data is compressed
  - Convert raw 12 bits ADC values into a 8 bit representation (lossy compression)
  - Allows for 16 sample (8 μs) waveform readout
  - Induces a small (1.5% fractional error on single samples)
  - Improves timing resolution by >10x



Fractional Error in ADC Compression



- Requires that data be "decompressed" within the online
- Realtime decoding is required to be as fast as possible in order to match maximum trigger latencies

Vectorization – Readout Decoding

- NOvA hit data is ideally packed to take advantage of vectorization
- Small computational kernel:
  - To small for thread based solutions
  - Data aligned on 32/64 bit word boundaries
  - Potential to give a 4x – 16x speed up in decode using SSE3 SIMD

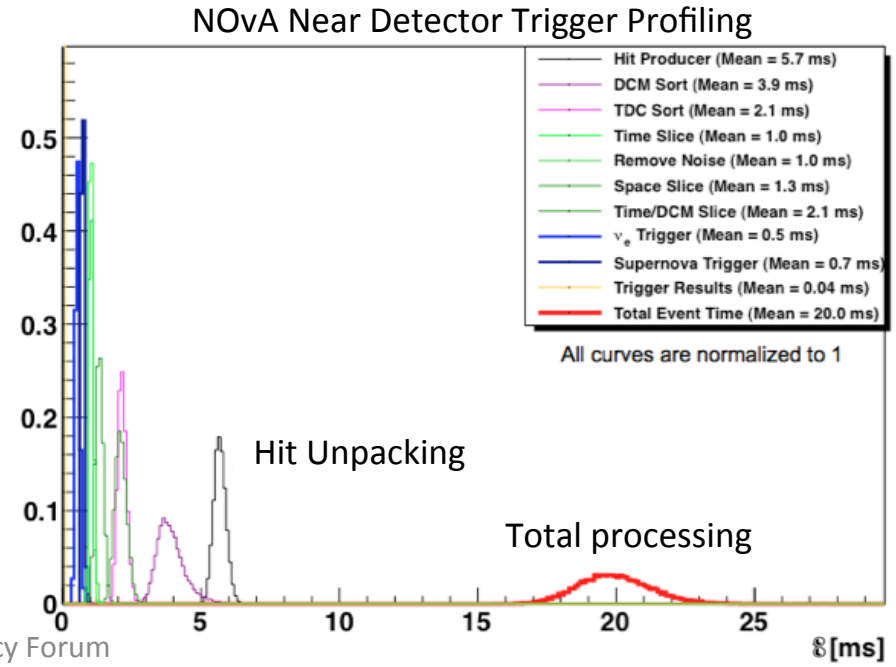
```
out_vector = ((0x1 << ((in_vector >> 5) - 1)) *
                    ((in_vector & 0x1f) + 32.5)) -0.5;
```

- Many similar hit processing tasks that are ideal for vectorization and are required to be fast for triggering

Projects underway within NOvA to understand how to exploit vectorization

31 24 23 16 15 8 7 0			
Nanoslice Header			
Timestamp counter (Sample 2)			
Baseline	ADC-1	ADC-2	ADC-3
ADC-4	ADC-5	ADC-6	ADC-7
ADC-8	ADC-9	ADC-10	ADC-11
ADC-12	ADC-13	ADC-14	ADC-15

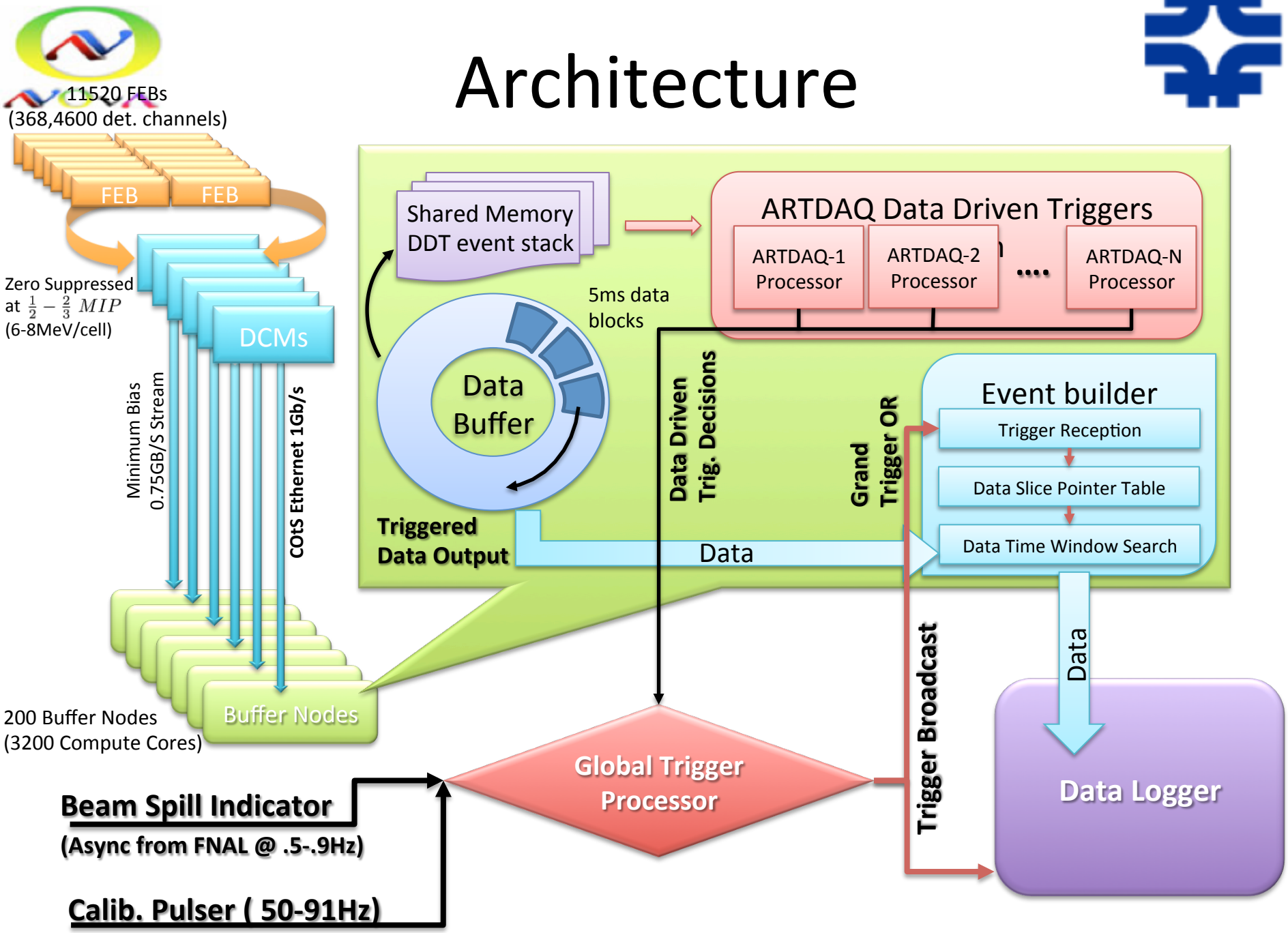
Figure 6: Nanoslice Format version 2.1, Compressed Waveform Readout



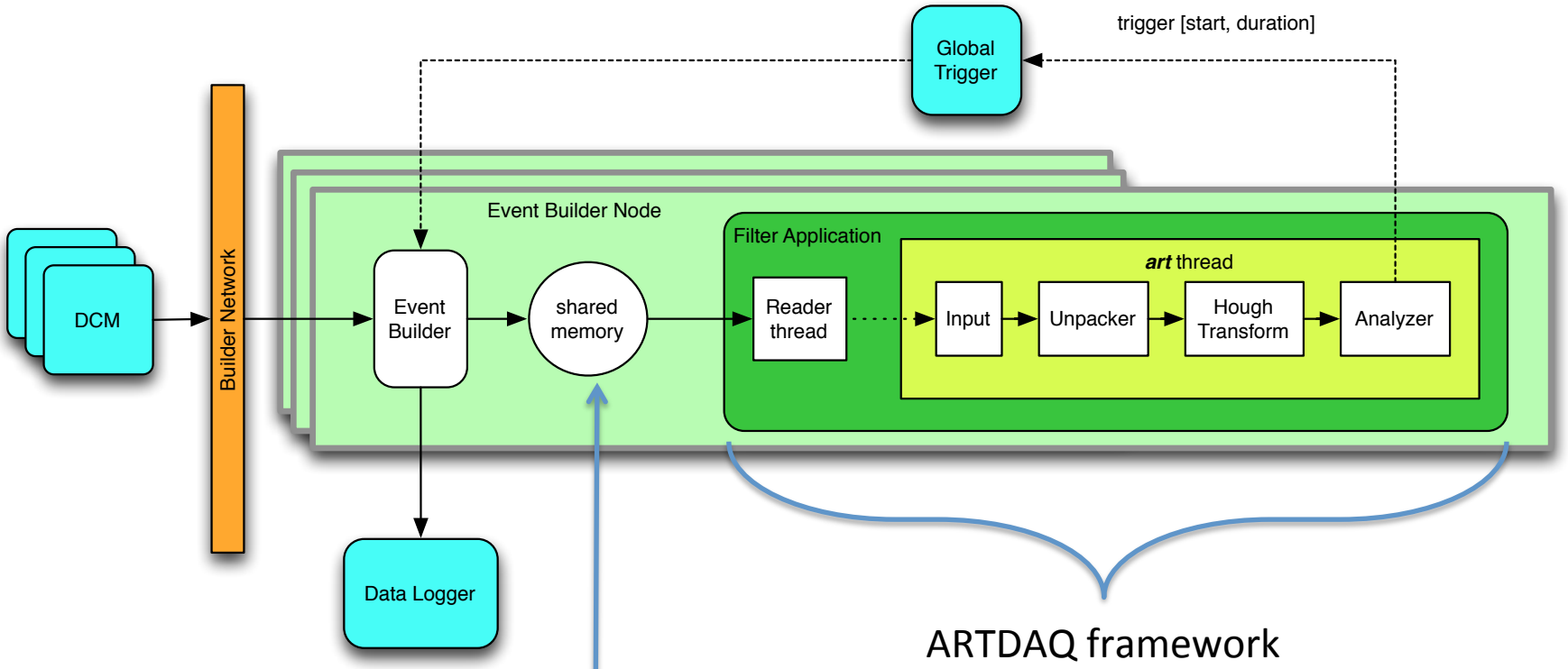
NOvA Trigger

- Realtime processing of “live” detector data
- L3 style computing farm (3200 cores)
- Stores continuous streaming zero-bias readout stream from detector
- Acts as a “buffer” to allow accelerator information from FNAL to transit to Far Detector in Minnesota (810km away)
  - ALL data from the detector goes into this farm
  - Gives an opportunity to examine the raw data for “interesting” topologies (horizontal cosmic, fully contained events, upward going tracks etc…)
- System is designed to be able to tolerate long latencies prior to triggering (i.e. ~20s)
  - But the number of hits that need to be processed per time frame are large (100,000+)

Architecture



Architecture



Shared Memory Interface layer to detector data. *Insulates the Event Builder from the ARTDAQ process.*

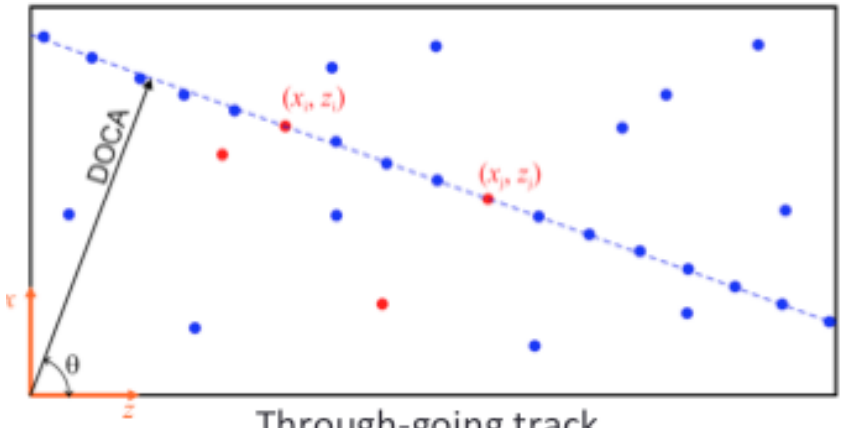
ARTDAQ framework Includes a separate data queue and then the appropriate input/unpacker modules, physics module and decision module

Pattern Recognition

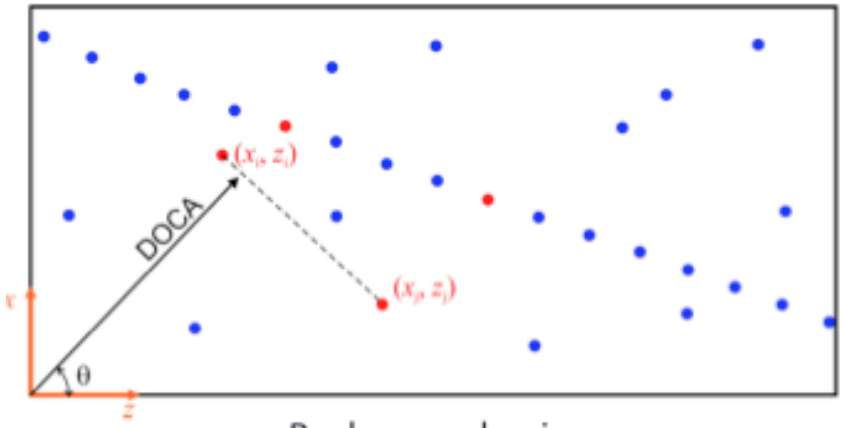
- General class of realtime PatRec algorithms
  - Identify track/cluster candidates
  - Structured with many repetitive, independent computations over the hit field
  - Ideal for parallelization on many-core
- Example: Linear Hough Transform

Linear Hough Transform

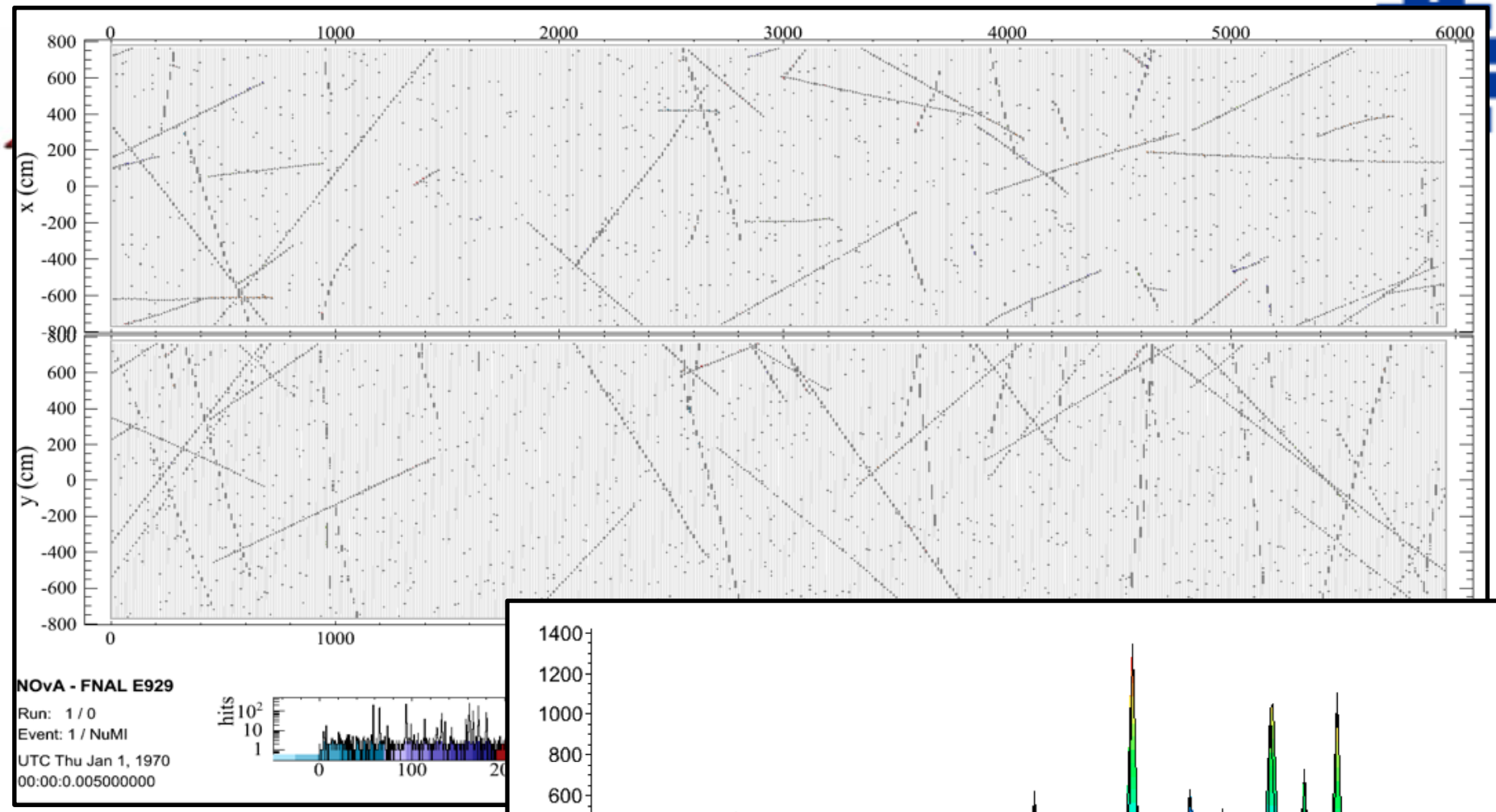
- 2-D Hough uses pair-wise combinations of hits to map from geometric space into an accumulation space
  - Characterizes lines by an angle θ and distance to reference point
  - Changes the problem of track finding to a "peak finding" problem
  - Simple computational kernel
  - Independent N^2 complexity

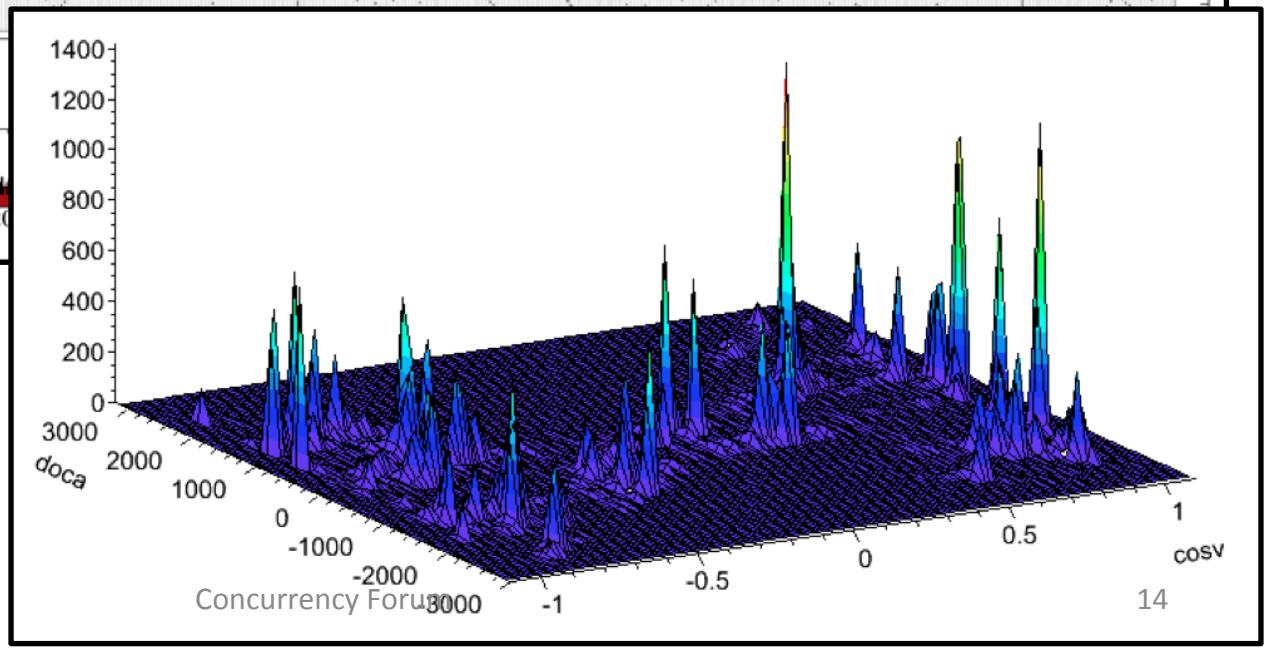


Through-going track.

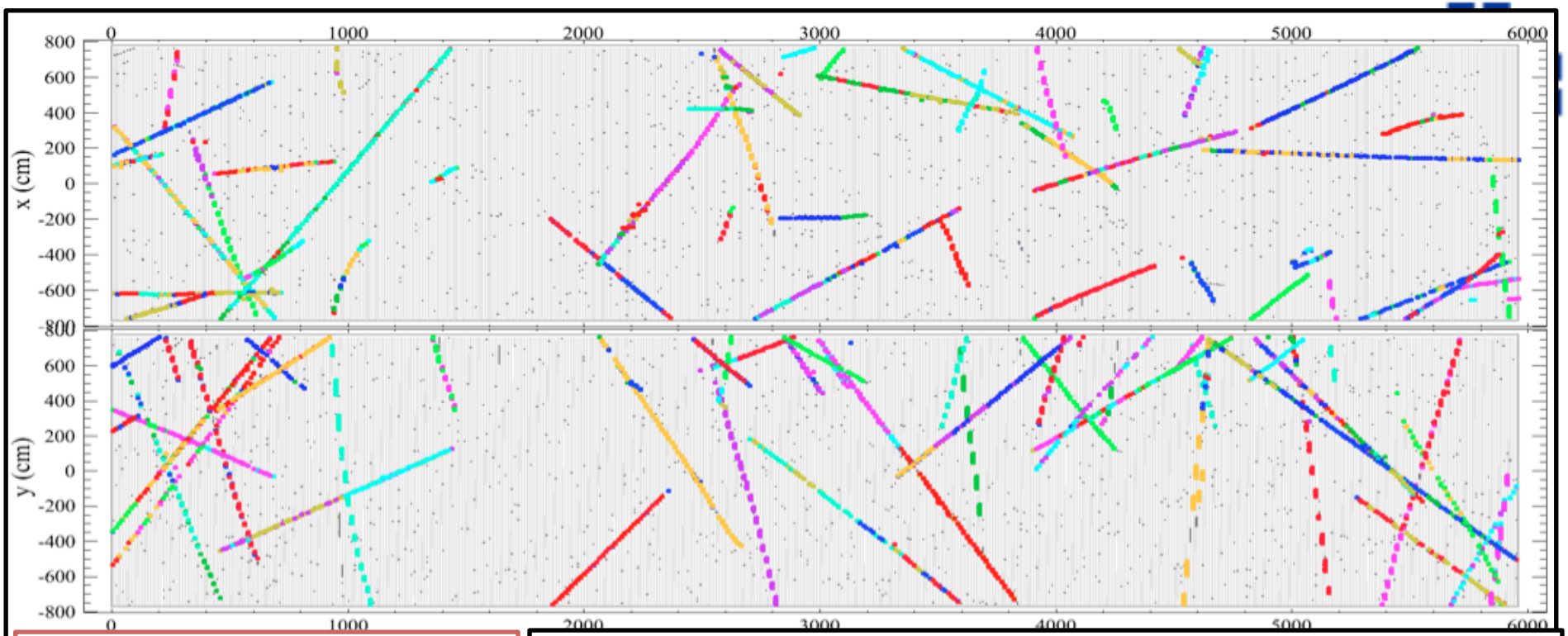


Background pair.

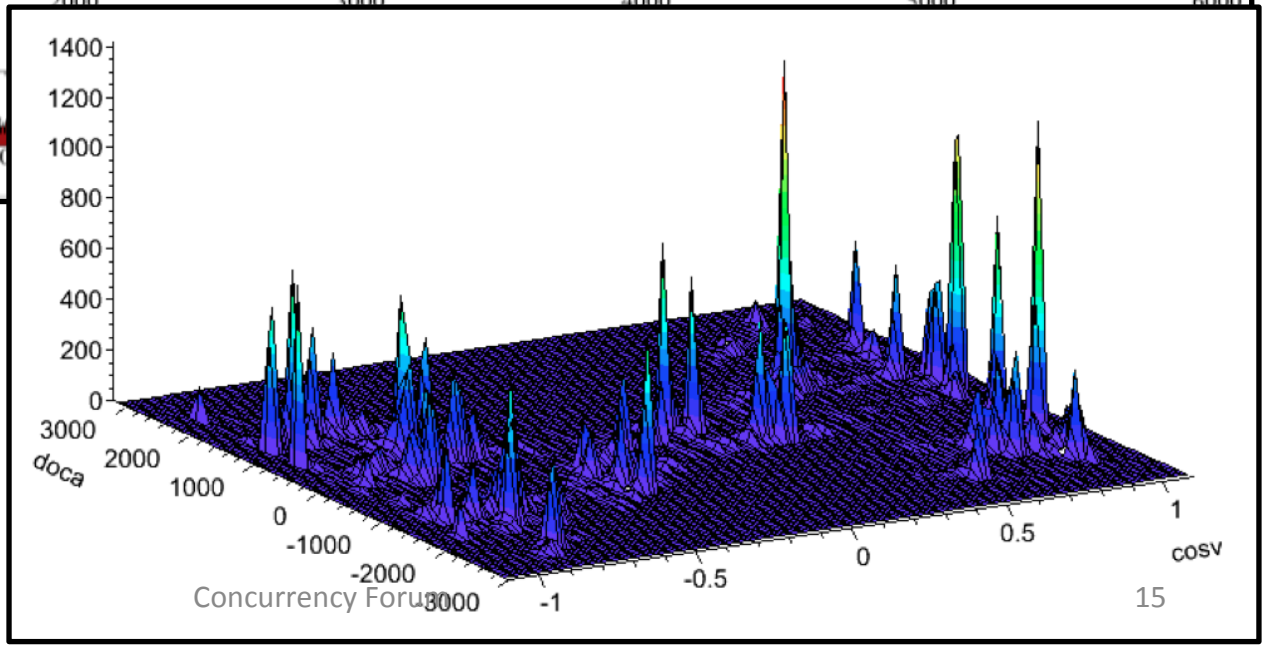




- 2D-Hough map detector hits into peaks in the Hough space



- Peaks identify the cosmic rays in the detector.
- Extensible to 3D and 4D using timing and Pulse Height information (complicated)
- Better isolation of cosmic background at higher dim. (allows for bkg subtraction)

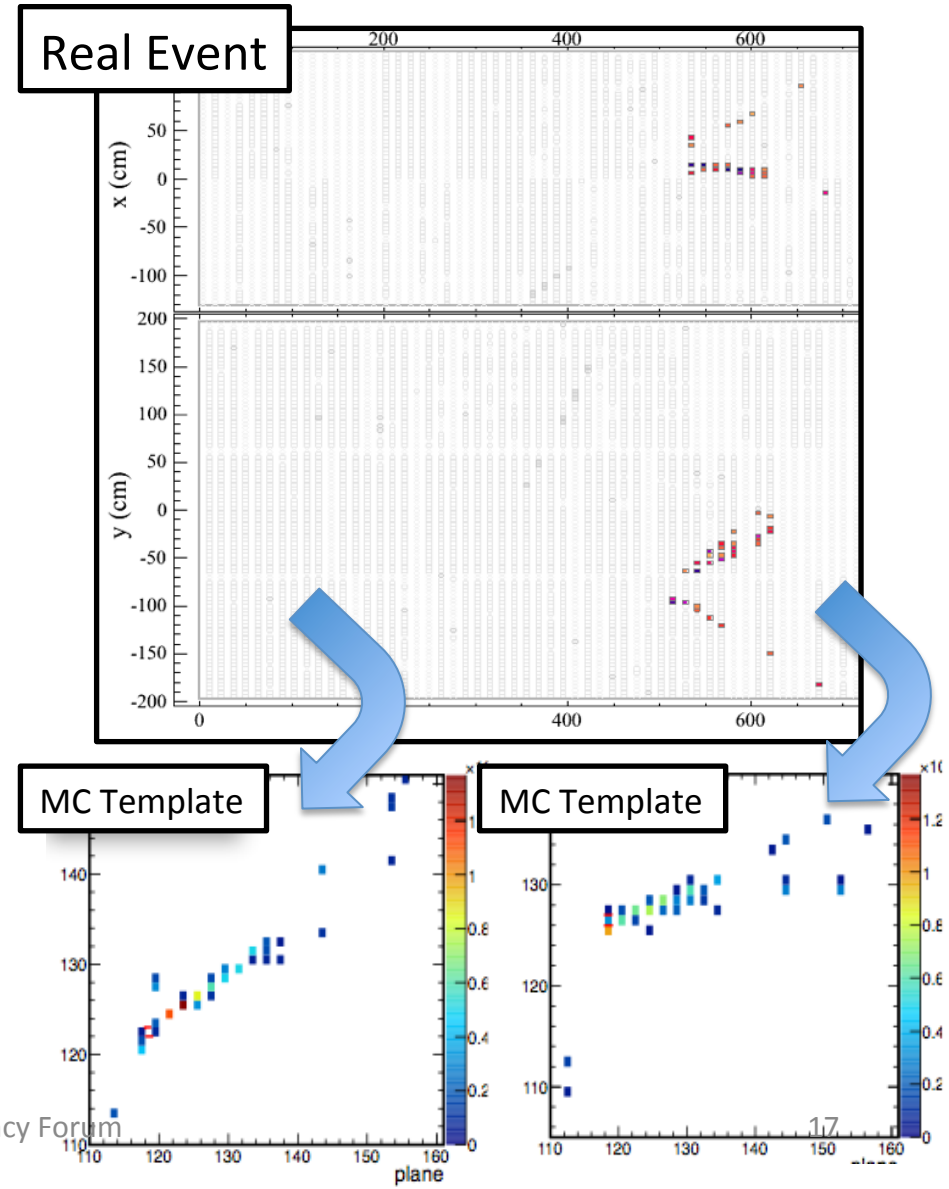


Hough Transform

- Algorithm is easy to parallel
  - Can unroll at hit loop levels
  - Maps onto large thread pools or GPU
  - Performance varies widely depending on how the actual parallelization is done
  - Projects underway within NOvA and Fermilab-SCD to explore performance of these types of algorithms
- See talk from ART framework group

Library Event Matching

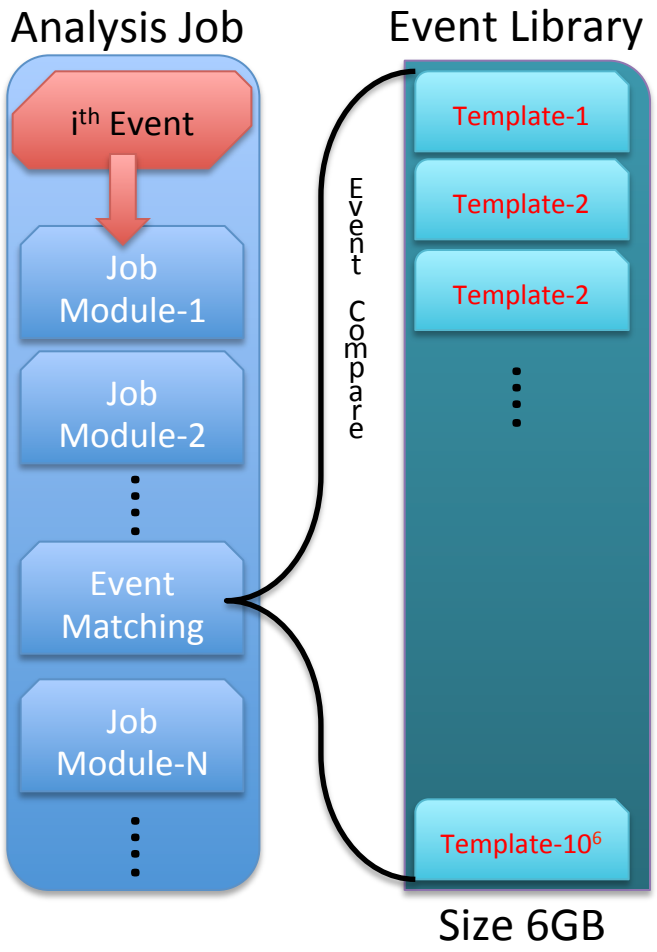
- Offline event classification in NOvA is difficult
  - ν_e-Charged Current events (signal)
  - Almost identical to ν_μ Neutral Current + π^0 (background)
- One approach compares an candidate event to a large library of template MC events
  - Establishes a metric for the degree of matching
  - Extracts event type and critical parameters from best matches



Library Event Matching

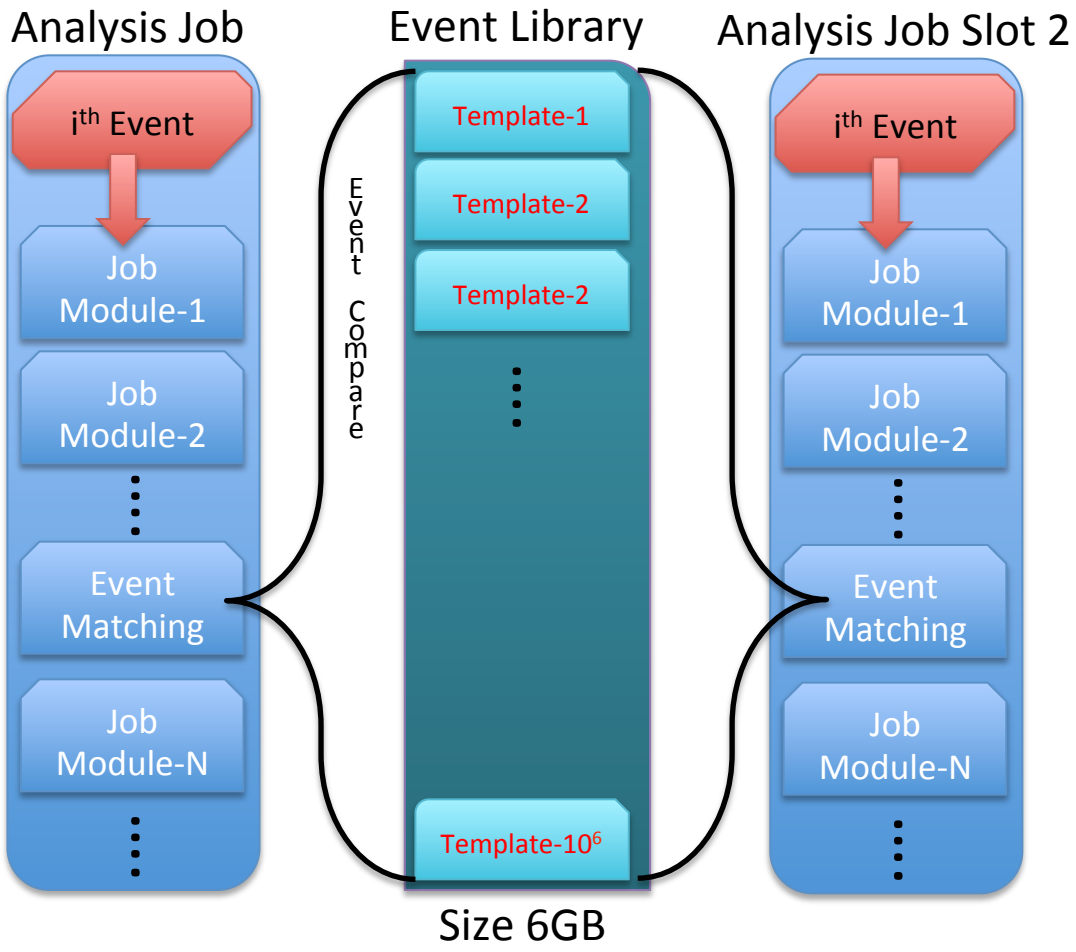
- Library is Large (~10^6 templates)
- Needs to be memory resident and cycled through for EACH event
- Memory footprint ≈6 GB

- Each iteration/comparison is independent



Library Event Matching

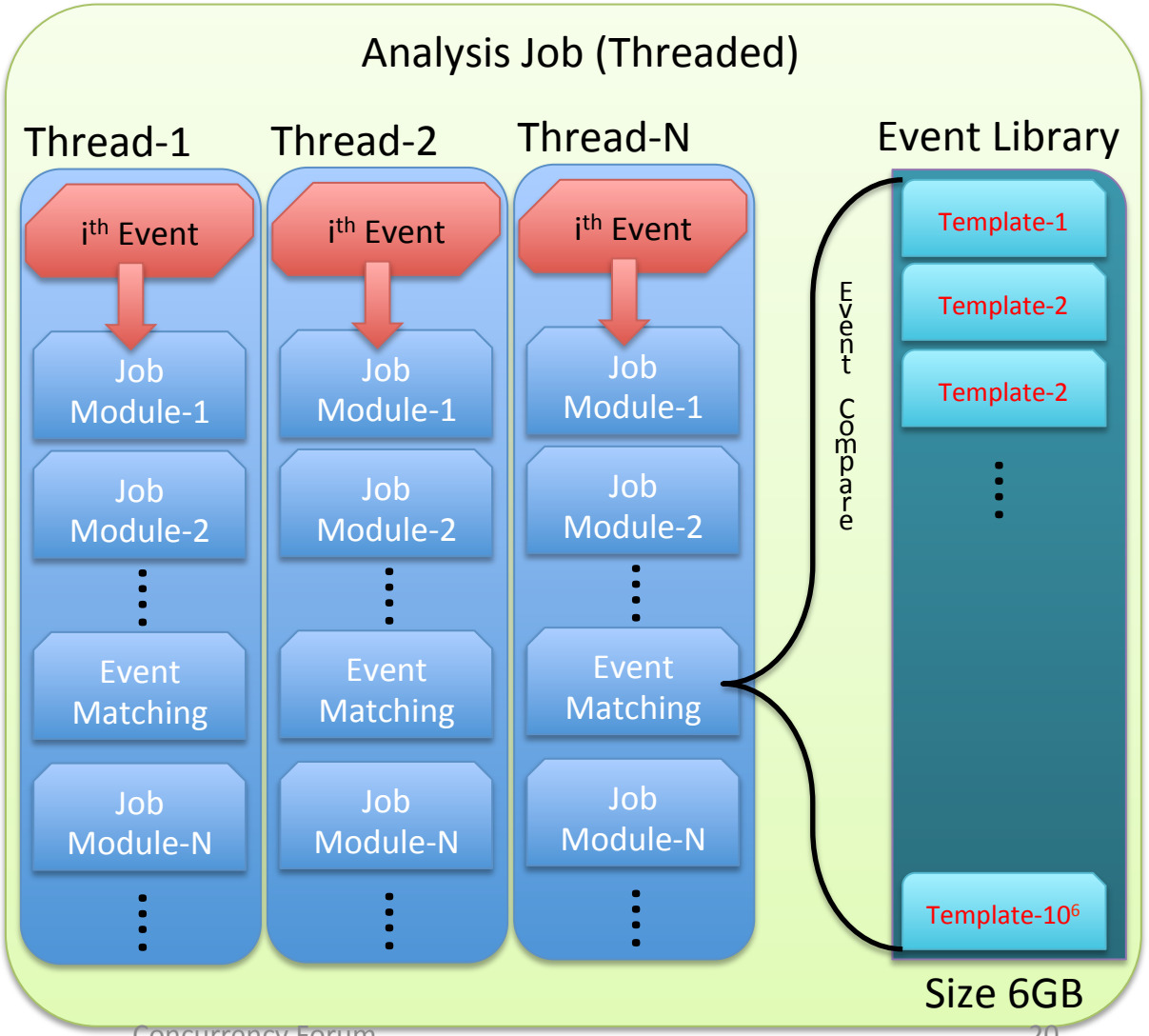
- Represents a static resource that ideally would be shared across a parallel environment
- Would allow for larger libraries
- Need to understand how to match to Grid computing models



Library Event Matching

- Threading represents another option in sharing the libraries
- Not clear how caching helps/hurts
- Need to understand how this can be matched to current Grid computing

Projects within NOvA to understand shared resource management in grid environments



Offline Simulation

- Event Generation
  - Need parallel event generation (neutrino & rock muon generation)
  - Sharing of common resources (Flux files)
- Detector Simulation
  - Need to move towards parallel tracking of particles within Geant4
  - Particularly important for cosmic ray overlays (180kHz muon rate)

Generally desired for g-2 and other intensity frontier experiments

- Goal is to increase performance enough to expand MC/Data ratio from 30:1 to 150:1 within same computing budget (driven by ND cross section analysis)

Summary

- NOvA has many different algorithms which are inherently parallel
  - Some areas have obvious “best” solutions
    - Challenge is using standard tools + hand tweaking to map the solution into the machine (vectorization)
  - Other algorithms have multiple solutions
    - Challenge is understand performance balance between threads, processes, GPU implimentations
  - Different class of problems is how to scale and share resources
    - Applies more generally to offline & Monte Carlo generation w/ grid resources