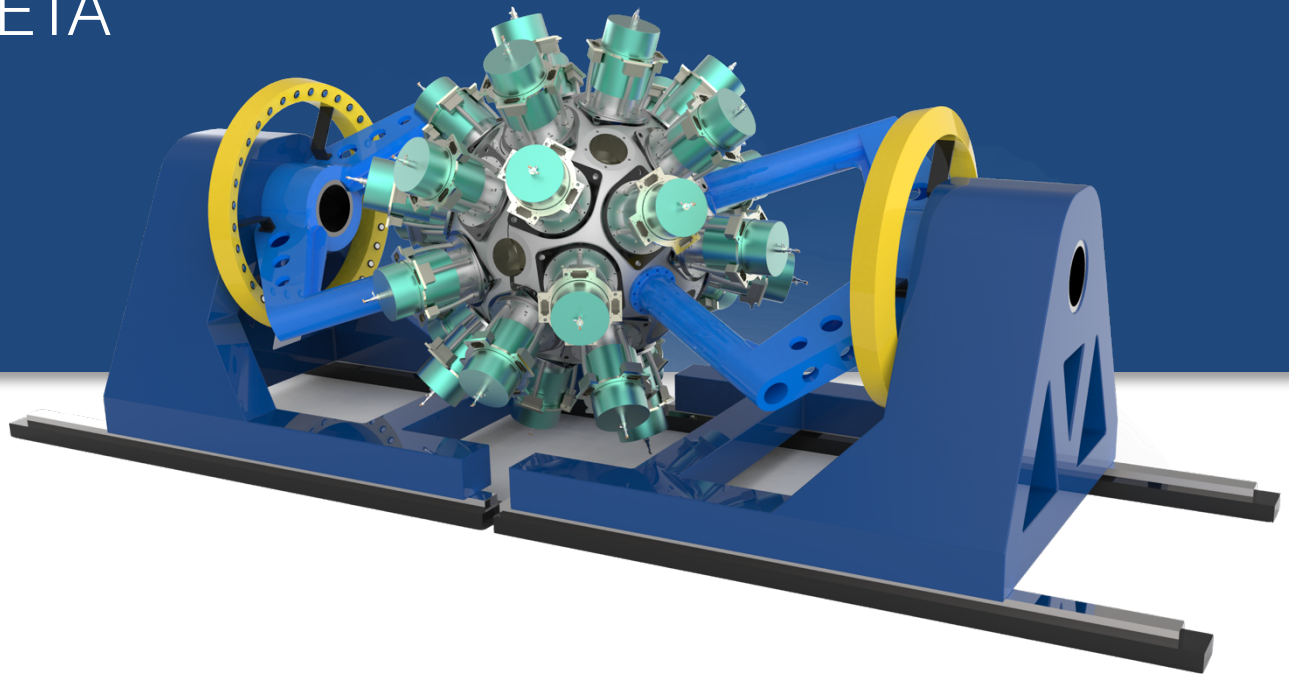
A Hybrid CPU/GPU Computing Solution for GRETA



Mario Cromaz

Computing L2 Manager

Nuclear Science Division

Lawrence Berkeley National Laboratory

U.S. DEPARTMENT OF ENERGY | Office of Science

Going Fast

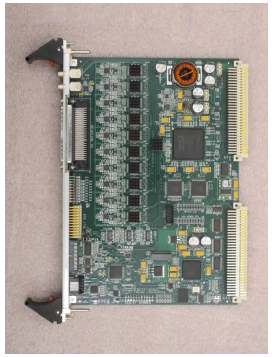
- Need to think about:
  - Implementation of the signal decomposition algorithm
  - The framework in which the algorithm is hosted (GRETA data plane)

People

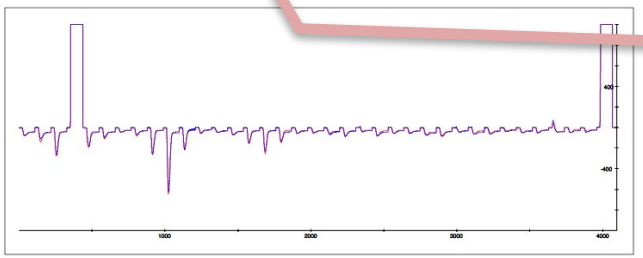
- System Architect
  [Eli Dart - ESNET, LBNL]
- High-performance data plane components (forward buffer, event builder)
  [**Eric Pouyoul** - ESNET, LBNL]
- Scientific programming (signal decomposition)
  [**Gustav Jansen**, David Radford, Robert Varner - NCCS/ Physics, ORNL]
- Cluster deployment, Development environment
  [Tin Ho, Susan James - Scientific IT group, LBNL]
- Control system engineering
  [Tynan Ford, Robert Gunion - Engineering division, LBNL]
- CAM
  [Mario Cromaz - NSD, LBNL]

High-level Data Processing in Tracking Arrays

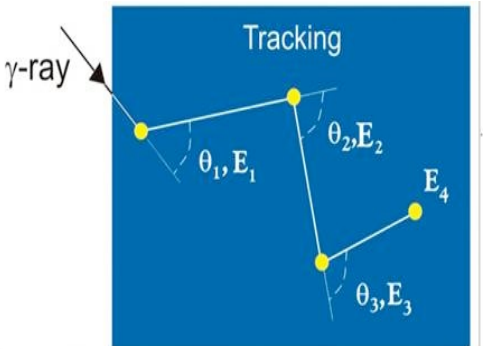
digitize segments, cc (36 + 1)



derive energy from trace



locate interaction points by fitting to crystal simulation

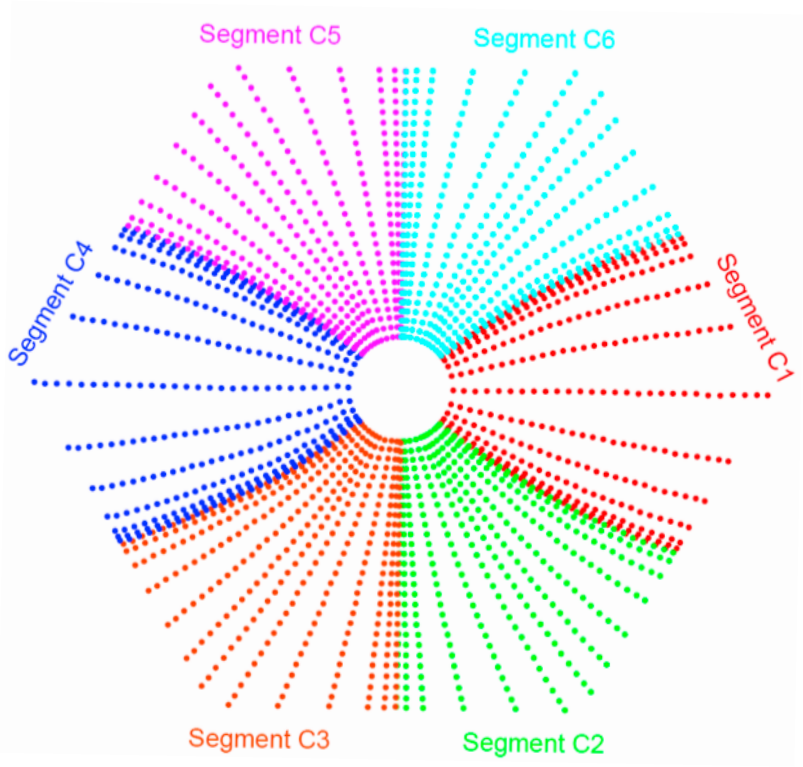


group/order interaction points by fitting to Compton scattering formula

digitizer module → filter boards → computing cluster → data analysis

GRETA Signal Decomposition Algorithm

- GRETA's real-time subsystem for locating interaction points employs GRETINA's signal decomposition algorithm (D.C. Radford)
- **Grid Search:**
  - 2-interactions in a segment; multiple segments by iteration
  - phase 1: course grid search
  - phase 2: fine grid search of neighboring points
- **Non-linear Least Square:**
  - allows fit to:
    - go off grid points
    - fit 3 interaction points / segment where appropriate
    - fit time offsets



Cross-section of points on equi-sensitivity grid in x-y plane for a GRETINA basis

GRETA Rate Requirements

6 Computing Systems Primary Requirements

6.1 Event Processing

Event processing, in terms of the computing systems, refers to the implementation of signal decomposition (pulse shape analysis) to extract interaction points within a given GRETA detector crystal based on the digitized waveforms from all segments and the central contact.

6.1.1 Signal Decomposition Algorithm

GRETA will support the existing GRETINA signal decomposition algorithm or one that exceeds its performance (based on the metric of position resolution) if it becomes available.

6.1.2 Real Time Processing Rate and Rate Asymmetry

The signal decomposition algorithm for GRETA must be able to run in real time, and be sustained at a maximal rate of 480k decompositions/s across the array. The maximal sustained rate for any given crystal should be ≥ 15k decompositions/s. The mechanism for scheduling of decomposition tasks should support a maximal rate asymmetry between detectors of 7:1 at maximal rate.

6.1.3 Bandwidth for Signal Decomposition

The input bandwidth to the signal decomposition processes must be sufficient to accept waveforms with 100 MHz sampling over 2 us (200 samples / ch), while the output should be sufficient to write out interaction points at maximal rate (480k events/s across the array).

Rate Implications on Cluster Size

- Signal decomposition is an online process - interaction point data required to monitor/debug experiments

- Initial estimate of cores required to meet GRETA specs:
  - Number of crystals - **120**
  - Rate (/ crystal, post trigger) - **4 kHz**
  - Time / crystal event - **10 ms / core** (2016 Xeon)
  - Implies **4800 core cluster** (40 cores / crystal)

GRETA Signal Decomposition Refactoring Effort

- **Functionality:**
  - To fit algorithm into the GRETA dataflow pipeline.
  - Decouple signal decomposition from I/O, slow control, and monitor sub systems.

- **Performance:**
  - Optimize for modern processors with high core counts.
  - Investigate the use of libraries in the minimization algorithm.
  - Investigate the use of GPU's in the signal decomposition cluster

Signal Decomposition Platform Evaluation

- Options:
  - ***CPU***
  - ***GPU***
  - ***Hybrid architecture***

- Important step before code refactoring
- Choice has cost/schedule impacts
- Need prototype/toy implementations for evaluations

Four Alternatives Considered for GRETA

- **CPU only** - Signal decomposition algorithm is implemented on a standard CPU. (GRETINA implementation)
- **Gradient descent on GPU** - Non-linear least squares fit part of the algorithm implemented on the GPU.
- **Matrix offload to GPU** - Matrix inversion operations in the signal decomposition algorithm are offloaded to a GPU.
- **Grid-search on GPU** - The grid search part of the signal decomposition algorithm is implemented on the GPU.

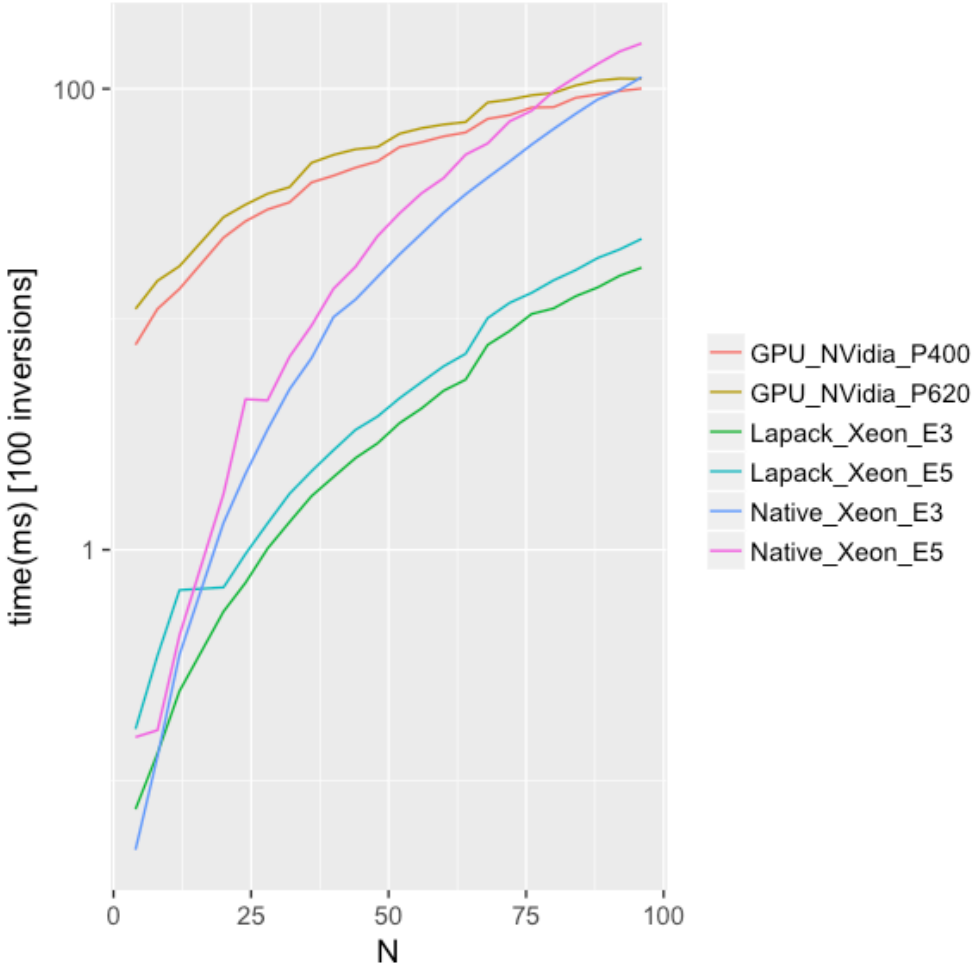
Gradient Descent on GPU

- Large impact .. gradient descent uses 88% of runtime in current test simulation

- ***but ..***
  - Limited parallelism in algorithm.
  - Will require massive rewrite of code
  - Unclear it is feasible to run multiple minimizations simultaneously?
  - Difficult to maintain as low-level code is necessary for optimal performance.

Matrix Inversion Offload to GPU

- Minimal development effort
- ***but..***
  - Matrices are too small for efficient GPU inversion (even when grouped)
  - Modern processors/ libraries push crossover to N=100
  - Current GRETINA matrix inversion works best for typical matrix size
- Holds promise though for larger scale fitting applications



Single precision matrix inversion times (blocks of 100) for Native (GRETINA), Lapack, and GPU implementations - *lower is better*

Grid Search on GPU

- Minimal development effort
- Prototype shows significant speedup compared to CPU-based implementation
- Single precision → consumer-grade hardware ($, not $$$)
- Will need to run full node tests to evaluate cluster performance with GPU's as shared resources

- ***Fine grid search runs in less time on GPU than course grid in CPU***

Why speed up grid search?

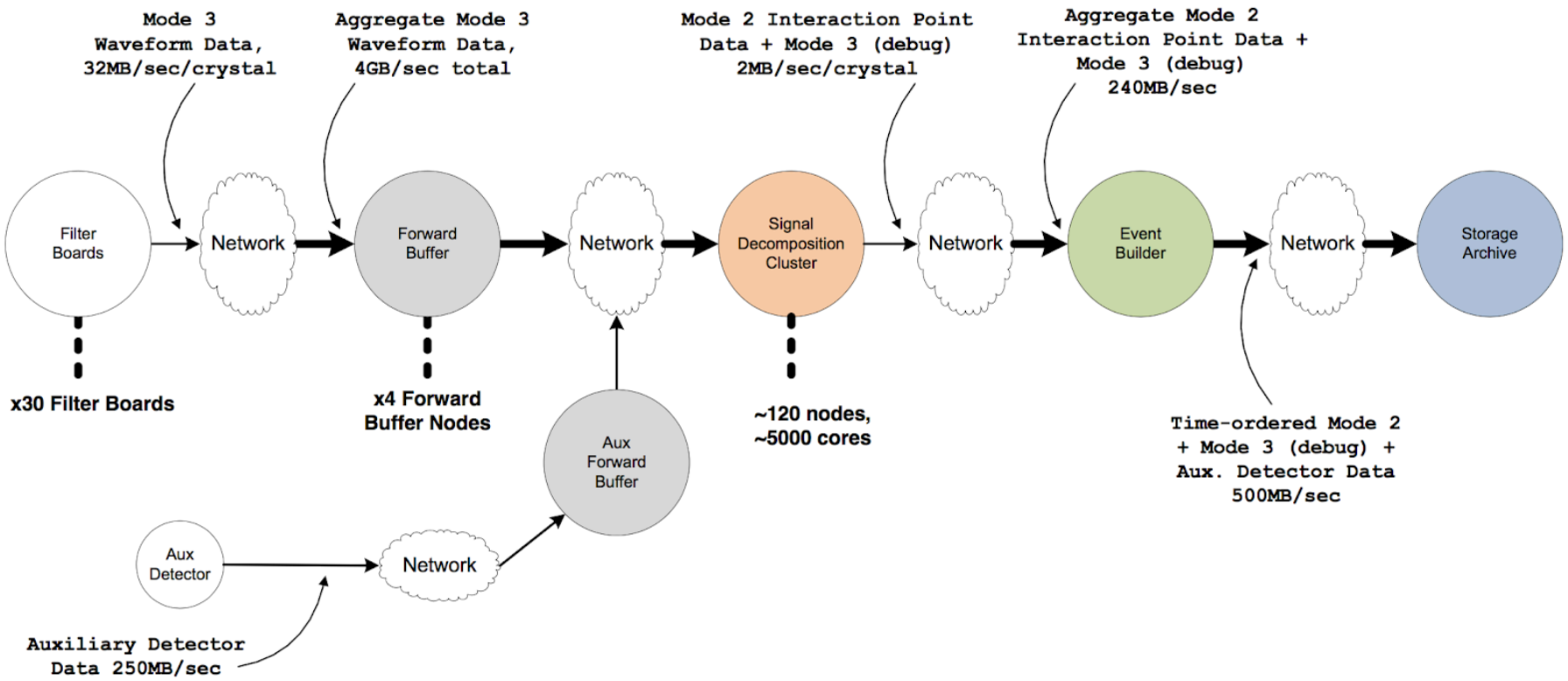
- Only 11% of execution time spent in grid search
- Can provide a better starting point for least squares fitter meaning fewer iterations

- More costly/complex grid search now possible:
  - exhaustive search on fine grid (1mm)
  - grid search + t0 offsets
  - 2 + 1 .. 2 interaction search in one segment, 1 interaction search in a neighbor

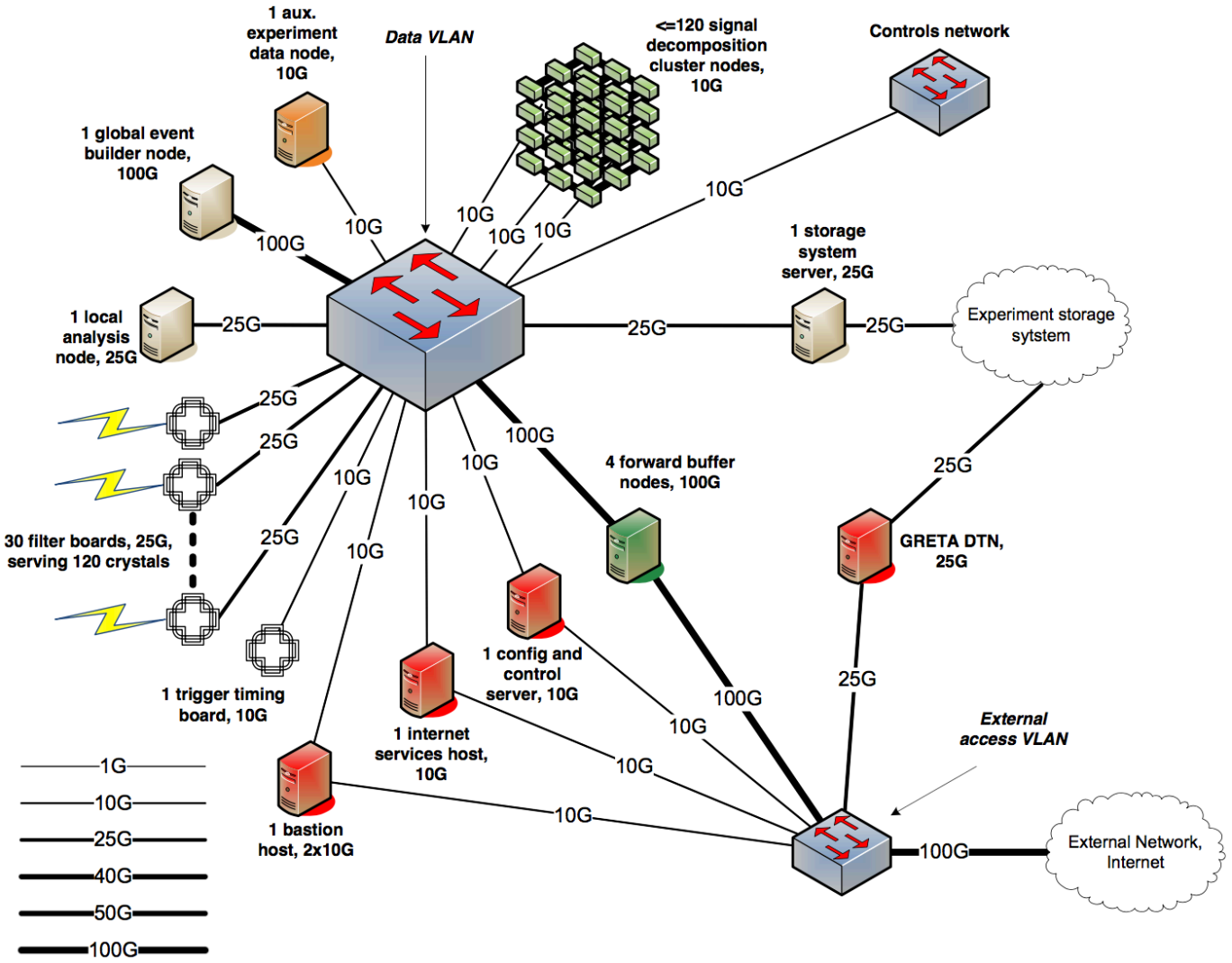
Work to be done

- Time-slicing GPU resources:
  - 1 or 2 GPUs will likely be shared on a single node (2 processors each with ≥ 20 cores)
  - How to schedule work? Bottlenecks?
- 2-interaction + t0 search
- Benchmarking - determine optimal mix of cores, GPU resources

The GRETA Data Plane

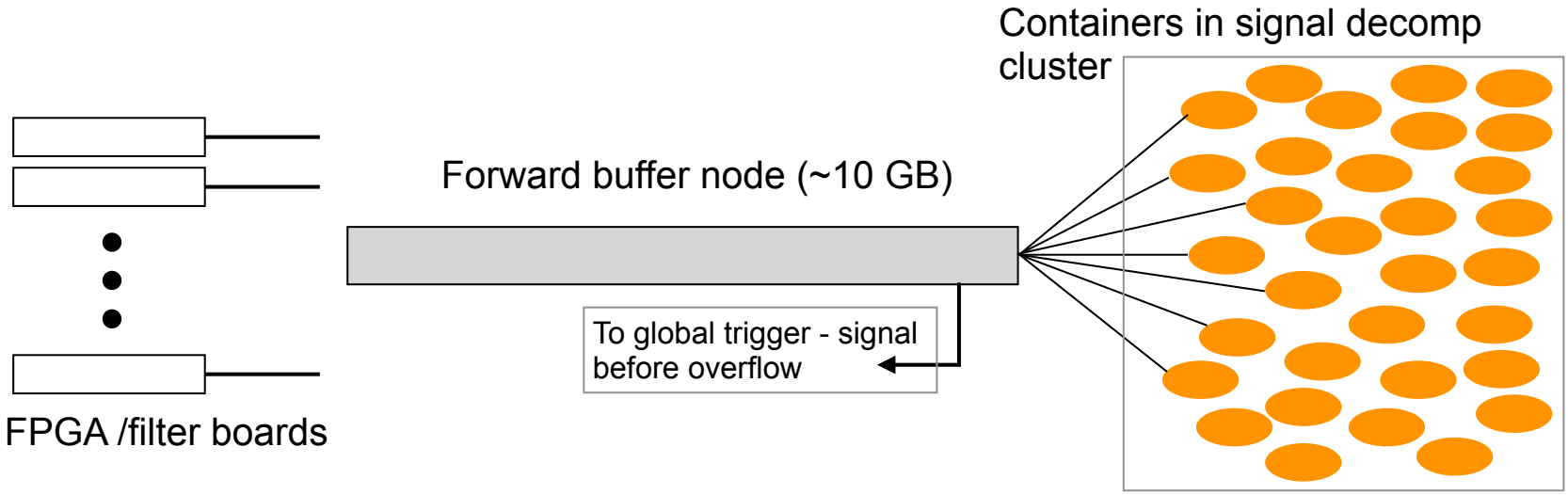


GRETA network diagram



Event Building - Forward Buffer

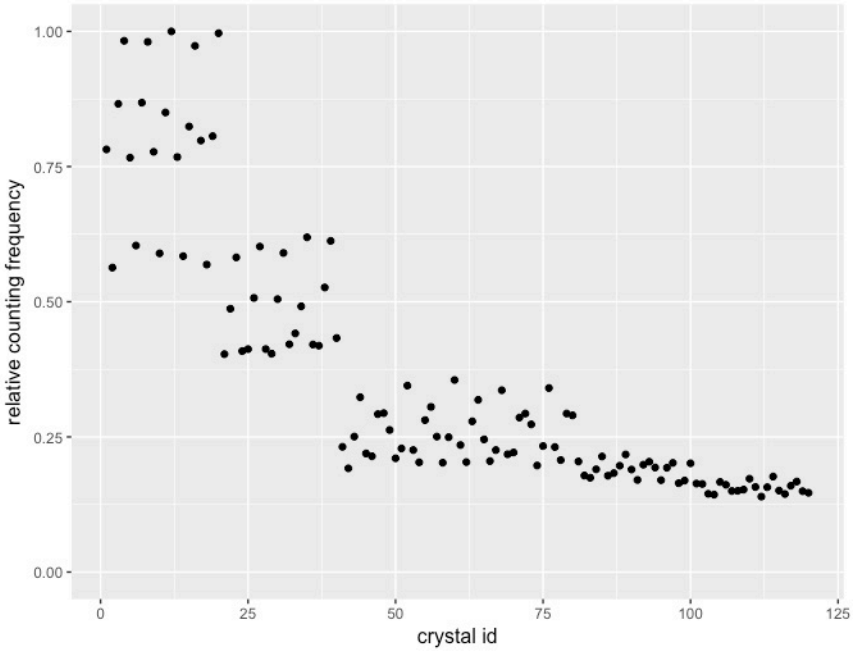
- Forward buffer is central to GRETA's computing architecture
- Aggregates data from electronics, serves data to decomposition cluster:
  - Allows electronics to have a simple 'push' interface using UDP
  - Signal decomposition containers can implement a 'pull' interface and self-schedule
  - Fill state of forward buffer queues allows implementation of global flow control



Forward Buffer Prototype Complete

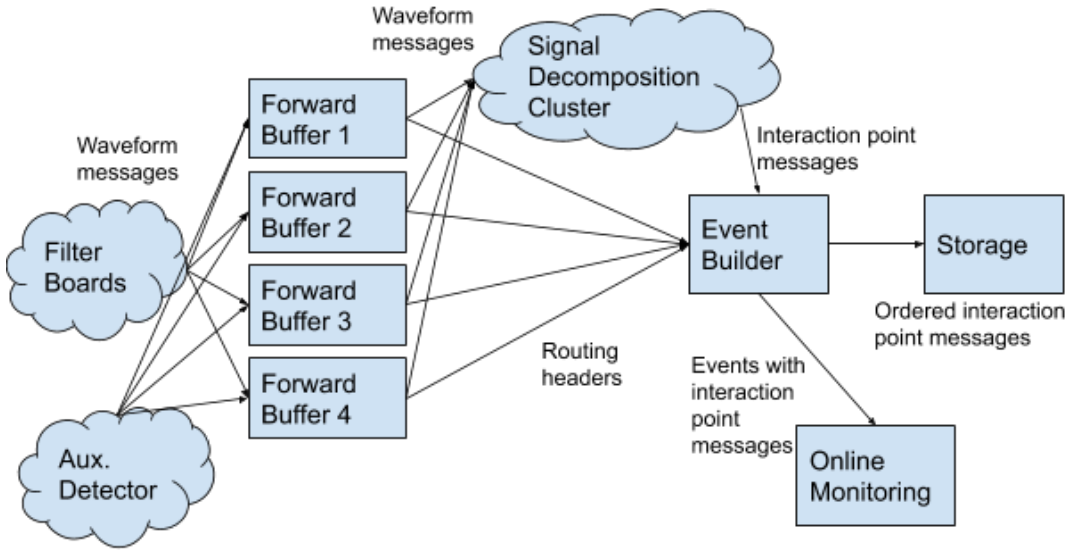
- Coded in *go*, employs nanomsg for outbound communications library
- Tested against filter board simulator, null ‘pullers’
- 2 × Intel Xeon 6146 @ 3.2 GHz, 384 GB RAM, Mellanox ConnectX 5 NIC (100 Gb)
- Prototype meets rate, asymmetry requirements
- Moving to newly installed GRETA development environment

- **Spec**:
  - 480k msg/s over 4 forward buffers
  - data rate: 1 GB/s/forward buffer
  - Support 7:1 rate asymmetry



Event Builder Prototype (in progress)

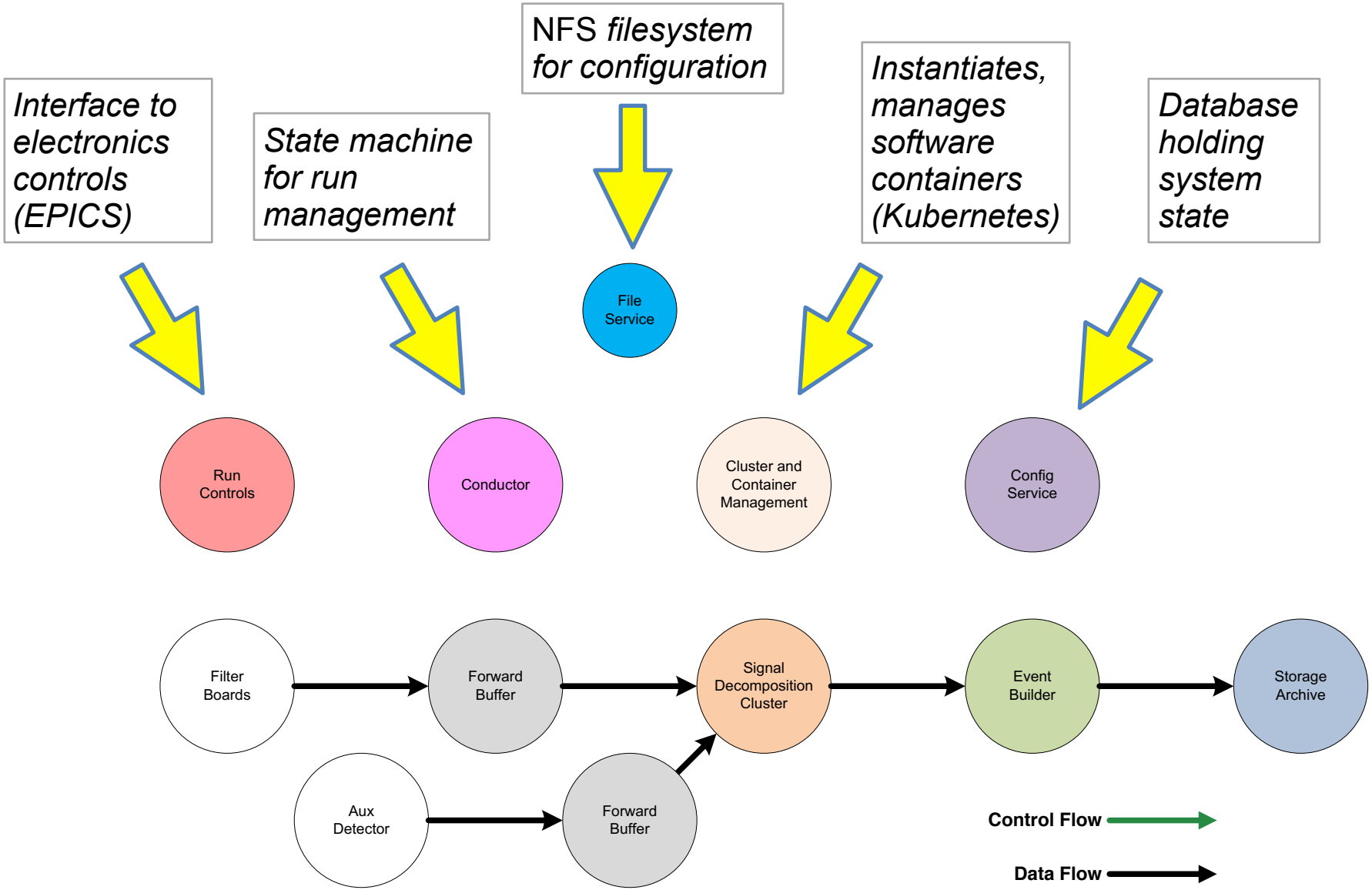
- Time order messages on its input queues and present it as a single stream to archiving
- Partition this time-ordered stream of messages into events and forward a subset on these messages to the online monitoring system.



- 'Deterministic' - not timeout driven (for gamma):
  - forward buffer sends routing headers to event builder on receipt
  - event builder allocates slots for data expected from signal decomposition cluster

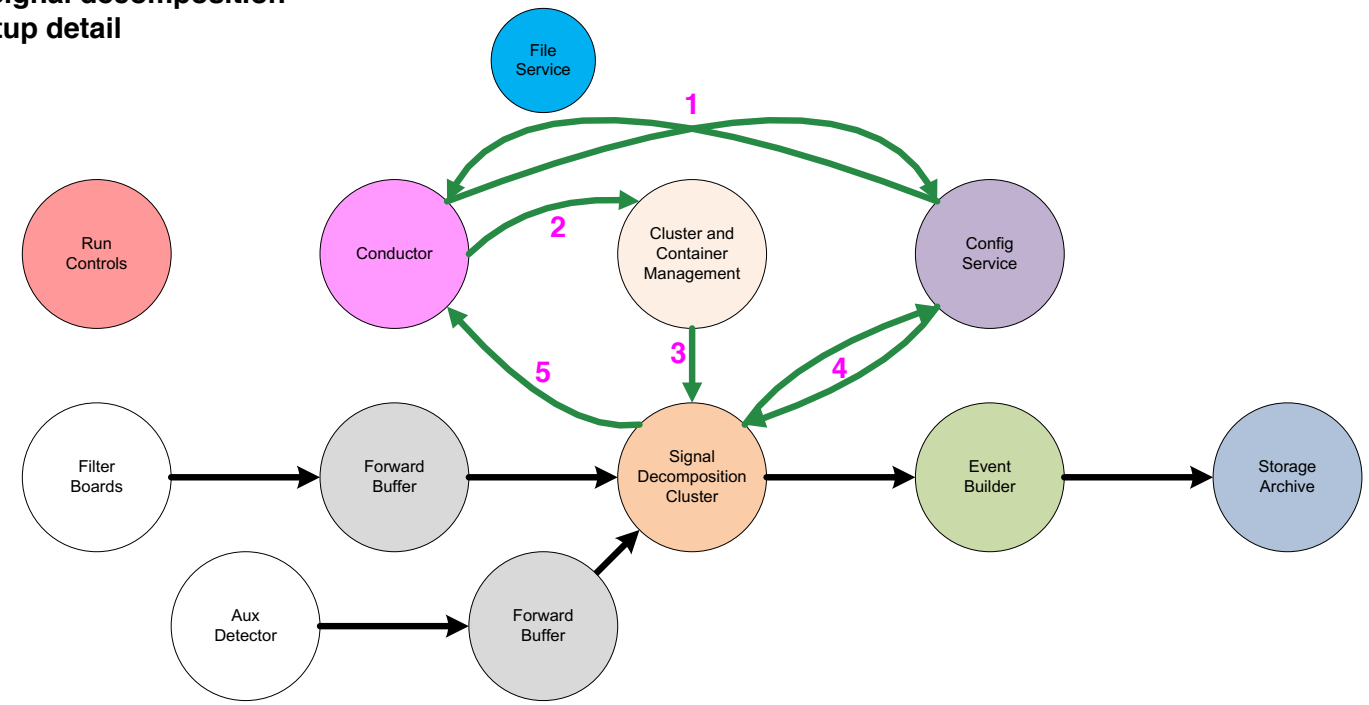
Rate requirement: 500 MB/s

Control Plane Architecture Reduces Development



Example of Control Plane Interactions

Workflow: Signal decomposition cluster startup detail



Start: Ready to start up the signal decomposition cluster containers (see run startup diagram, step 4b)

1) Conductor gets experiment-dependent container set info from configuration service
2) Conductor tells cluster management to launch signal decomposition containers
3) Cluster management launches signal decomposition containers on appropriate hardware and passes each a configuration URL
4) Signal decomposition containers start up and fetch configurations, then start running
5) Signal decomposition containers inform conductor that they are up and running

Finish: Signal decomposition containers are up and running

Control Flow →

Data Flow →

Development Network

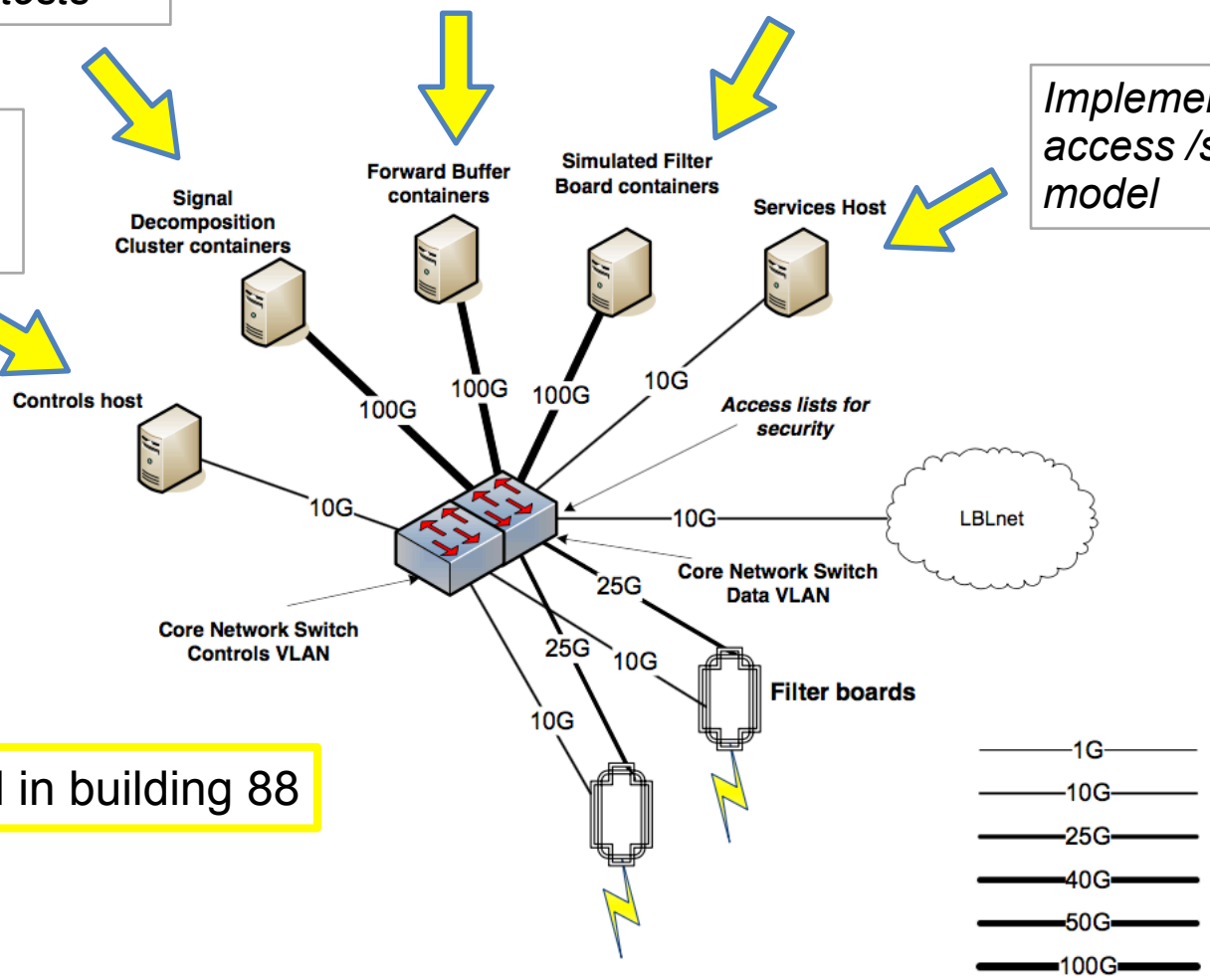
Test platform for signal decomp, scaling tests

Implements 1 of 4 forward buffers

Simulates traffic from electronics subsystem at full design rate

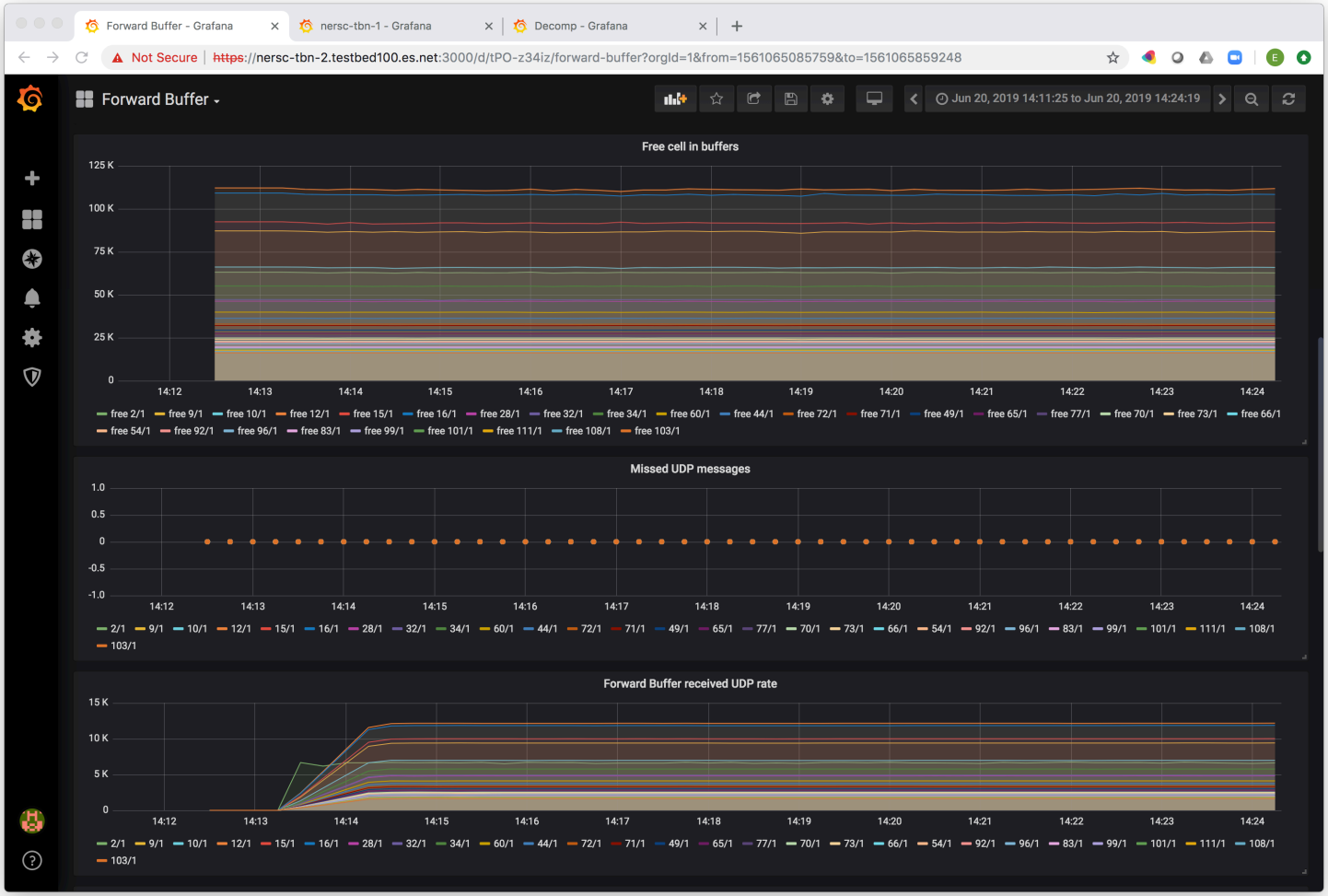
Test platform for EPICS controls

Implements GRETA access /security model



Deployed in building 88

Cluster Monitoring / Analytics



- *GRETA utilizes Prometheus for analytics/time-series database*
- *Supports both application and node level API for monitoring*
- *Graphana provides the UI for op consoles*

Conclusion

- Examined multiple platform/implementation options for GRETA signal decomposition:
  - **CPU only** ✅
  - **Grid-search on GPU** ✅ ✅
- We have a prototype implementation of grid-search algorithm for signal decomposition
- Speed-up will enable more comprehensive searches (finer grid, more parameters) do be done online
- GRETA data plane is network based and enables very high speed data transfers using a forward buffers
- We have a prototype forward buffer in hand which meets specification, prototype event builder under construction