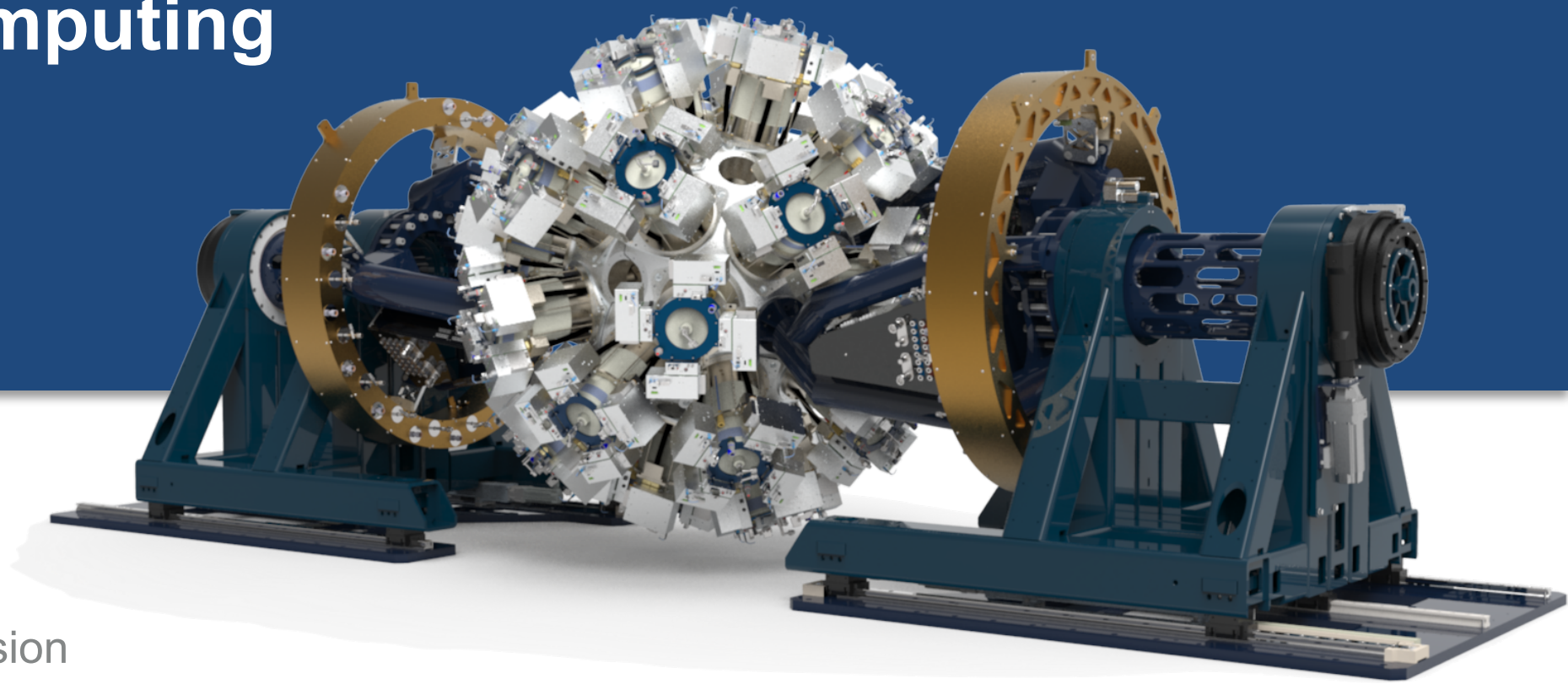
GRETA Computing Model



Mario Cromaz
Nuclear Science Division
Lawrence Berkeley National Laboratory

BERKELEY LAB
Bringing Science Solutions to the World

4th AGATA-GRETINA/GRETA Collaboration Meeting, Nov. 20, 2024

U.S. DEPARTMENT OF ENERGY | Office of Science

GRETA Production Computing Environment

- High-speed readout of the GRETA filter boards
- From digitizer waveforms infer the position of gamma-ray interaction points in real time (seconds)
- Time correlate data from all detector elements and auxiliary detectors to construct global events
- Local file store and data transfer capabilities
- Provide an interface to monitor and optimize experiments
- Controls and monitors for electronics, HV/LV, interface to LN-fill system

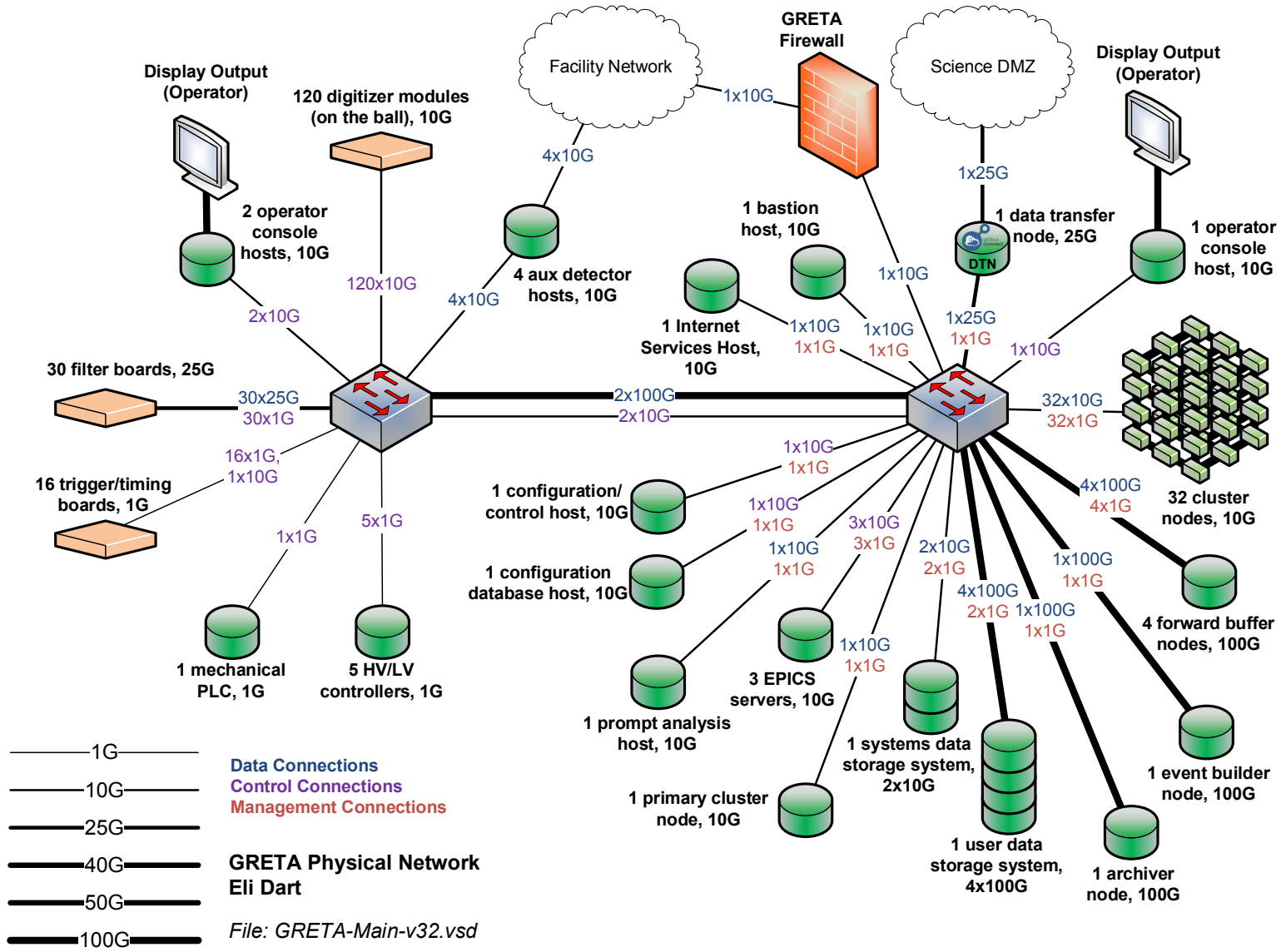
Cluster Racks in 50B-1275 Data Center

Computing Requirements for GRETA

- Computing scale set by the **online** signal processing requirement:
  - 480 k events / s **(GRETA objective KPP)**
  - Implies 4 GB / s aggregate post FPGA processing - 8 kB / event (when compressed)
  - Base calculation requires ~5 ms / signal decomposition - requires large, detector-specific, in-memory simulation (~ 2 GB / detector, 120 detectors)

- Each experiment can load the cluster differently** - up to 7:1 rate asymmetry in detector rates

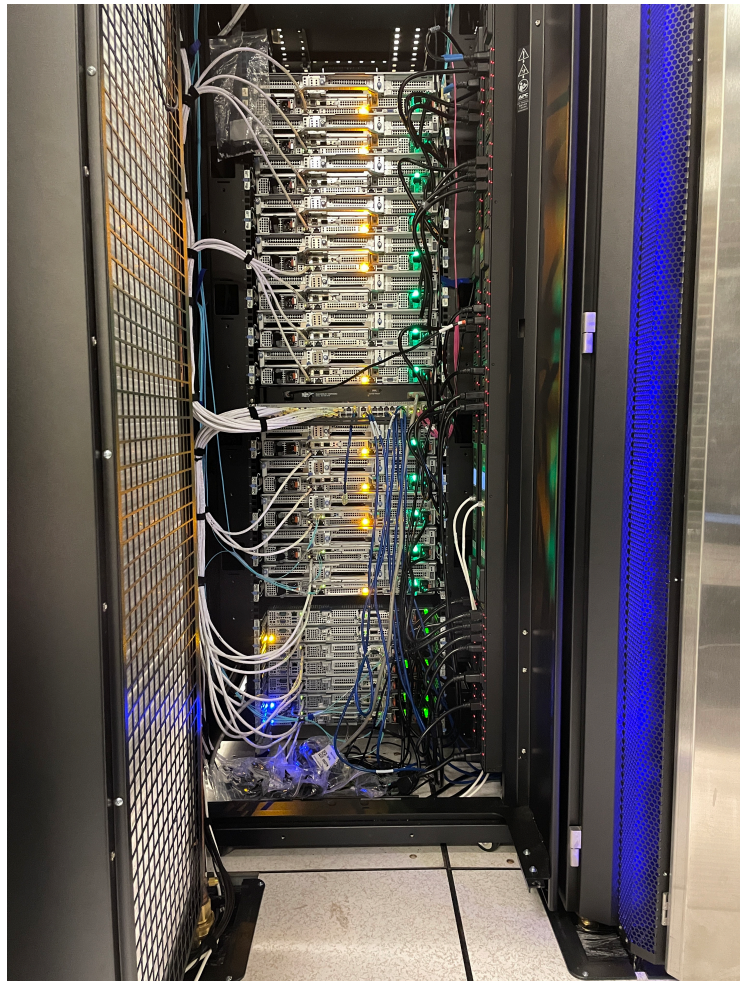
Network Diagram



GRETA Production Computing Cluster

- GRETA cluster hardware is deployed and operational in 50B-1275 data center
- 3 wide-format racks
- 3 basic node types (compute/GPU, network, service) to increase maintainability
- Power: 50 kW
- Racks will move to FRIB fully populated (limited interconnects between racks)

RU#	Rack A	Rack C	Rack B
42	**7280 100G sw "core" Q28**	7020 10G sw	7020 10G sw
41		empty	Juniper Firewall
40		2U gpu node c18	*empty*
39			
38		2U gpu node c17	*empty*
37			
36		2U gpu node c16	2U gpu node c31
35			
34		2U gpu node c15	2U gpu node c30
33			
32		2U gpu node c14	2U gpu node c29
31			
30		2U gpu node c13	2U gpu node c28
29			
28		2U gpu node c12	2U gpu node c27
27	empty		
26		2U gpu node c11	2U gpu node c26
25			
24		2U gpu node c10	2U gpu node c25
23			
22		2U gpu node c09	2U gpu node c24
21	Vast FS 4 for prod, 6RU for final p		
20	empty	empty	1U console
19	Arista 7010T 1G "A" Switch	Arista 7010T 1G "C" Switch	Arista 7010T "B" switch
18	sysbase netapp (2U)	2U gpu node c08	2U gpu node c23
17			
16	sysbase netapp expansion (2U)	2U gpu node c07	2U gpu node c22
15			
14	empty	2U gpu node c06	2U gpu node c21
13	empty		
12	empty	2U gpu node c05	2U gpu node c20
11	n04: fwd buffer 100G 1U		
10	n03: fwd buffer 100G 1U	2U gpu node c04	2U gpu node c19
9	n02: fwd buffer 100G 1U		
8	n01: fwd buffer 100G 1U	2U gpu node c03	empty
7	n00: event builder 100G 1U		s02 - service 10G 1U
6	x02 100G intel cpu	2U gpu node c02	s01 - service 10G 1U
5	x01 100G intel cpu		s00: Service 10G 1U
4	x00 100G intel cpu	2U gpu node c01	s06:: Service 10G 1U
3	t00: Dell 1U		s05:: Service 10G 1U
2	d01: Service+ 100G 1U	2U gpu node c00	infra1 1U
1	**d00: Service+ 100G 1U**		infra0 1U

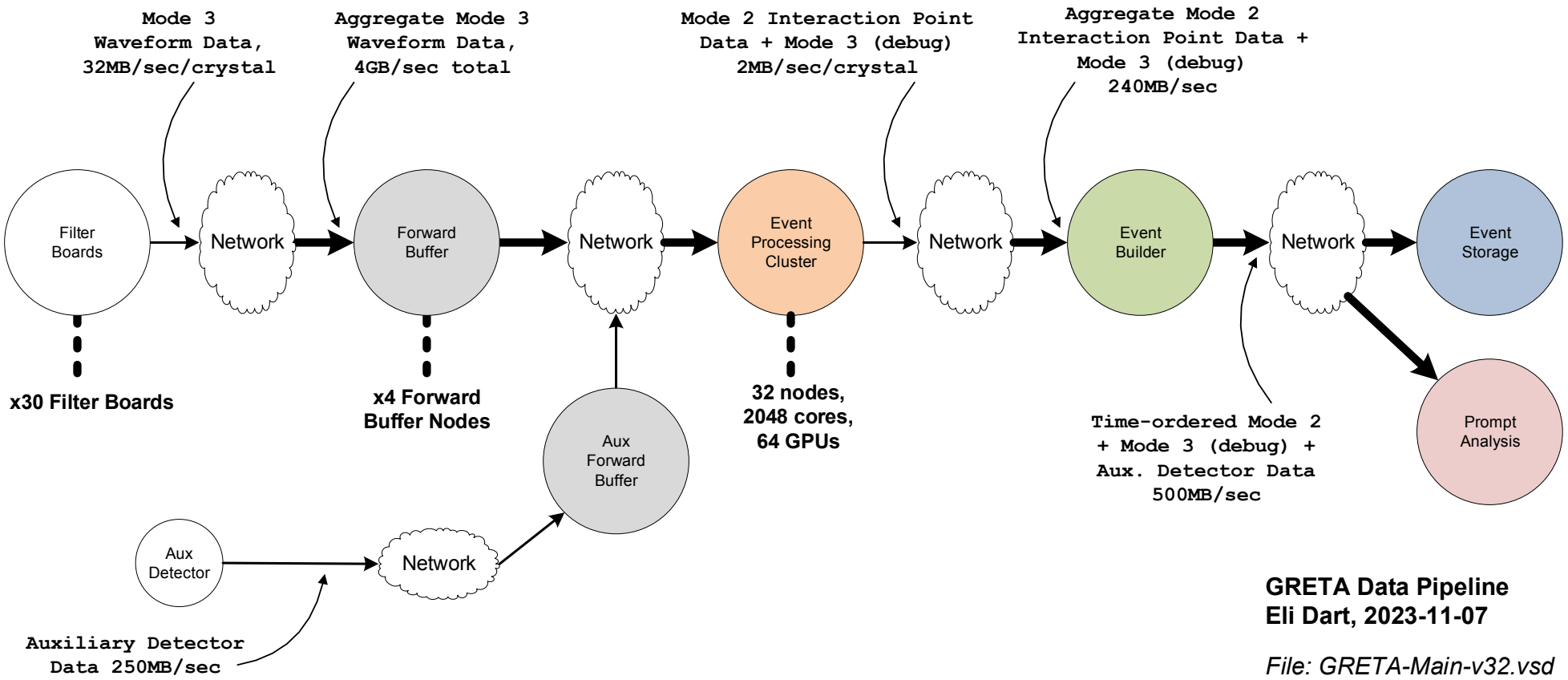


Rack C rear view

GRETA Cluster has 3 Node Types

- **`c` nodes**: compute / signal decomp
  - Dell 7525, 2 x AMD EPYC 7513 (32 core, 2.6 GHz), 2 x Nvidia A10 GPU, 256 GB Ram, 10 Gb Intel X710 NIC
- **`n` nodes**: forward buffers, event builder
  - Dell 6525, 2 x AMD EPYC 7343 (16 core, 3.2 GHz), 512 GB Ram, 100 Gb Mellanox ConnectX-6 NIC
- **`s`, `s+` nodes**: controls, system function
  - Dell 6515, AMD EPYC 7313P (16 core, 3 GHz), 128 GB Ram, 10 Gb NIC / 100 Gb NIC (+)

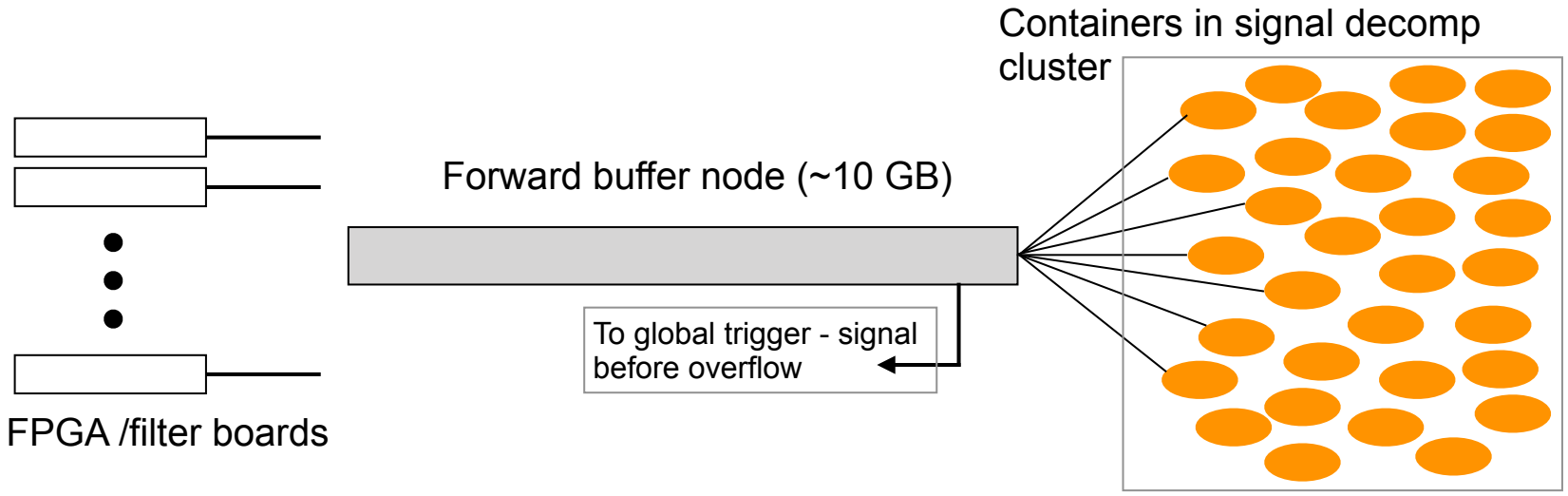
Data Pipeline



See: *Simple and Scalable Streaming: The GRETA Data Pipeline, EPJ Web of Conferences 251, 04018 (2021)*

Forward Buffers

- Forward buffer is central to GRETA's computing architecture
- Aggregates data from detector electronics, serves data for event processing:
  - Allows electronics to be free running - have a simple 'push' interface using UDP
  - Event processing containers can implement a 'pull' interface and self-schedule
  - Fill state of forward buffer queues allows implementation of global flow control



This architecture greatly simplifies the FPGA readout implementation; allows use of standard networking/computing hardware and software tools

Key Pipeline Components

- **Forward Buffer:**
  - Mediates communication between the electronics and computing cluster
  - Allows electronics to be free running and cluster nodes to self-schedule
- **Event Processing:**
  - Locate interaction points from waveforms (signal decomposition)
  - Each compute node presents a number of job slots which are assigned to a specific detector
  - Hybrid CPU/GPU implementation
- **Event Building:**
  - Aggregates events according to timestamps into global events
  - Will provide a tap for online data monitoring

- Pipeline components transfer data via GAP (Greta Application Protocol) layer
- Based on nanomsg - zero copy, low latency, excellent stability (https://nanomsg.org)
- GAP implements standard communication design patterns (fan-in, fan-out)

Pipeline Control Plane

- Configuration Service
  - DB + API for configuration of pipeline
- Conductor
  - Generates component specific config for pipeline, orchestrates containers
- UI / Web app
  - User interface to config service, conductor, and run control
- Monitoring Service
  - Records status info from pipeline elements into time series DB (Prometheus)
  - Displays summaries to operator (Grafana)

- Supports scripting/ web app symmetrically

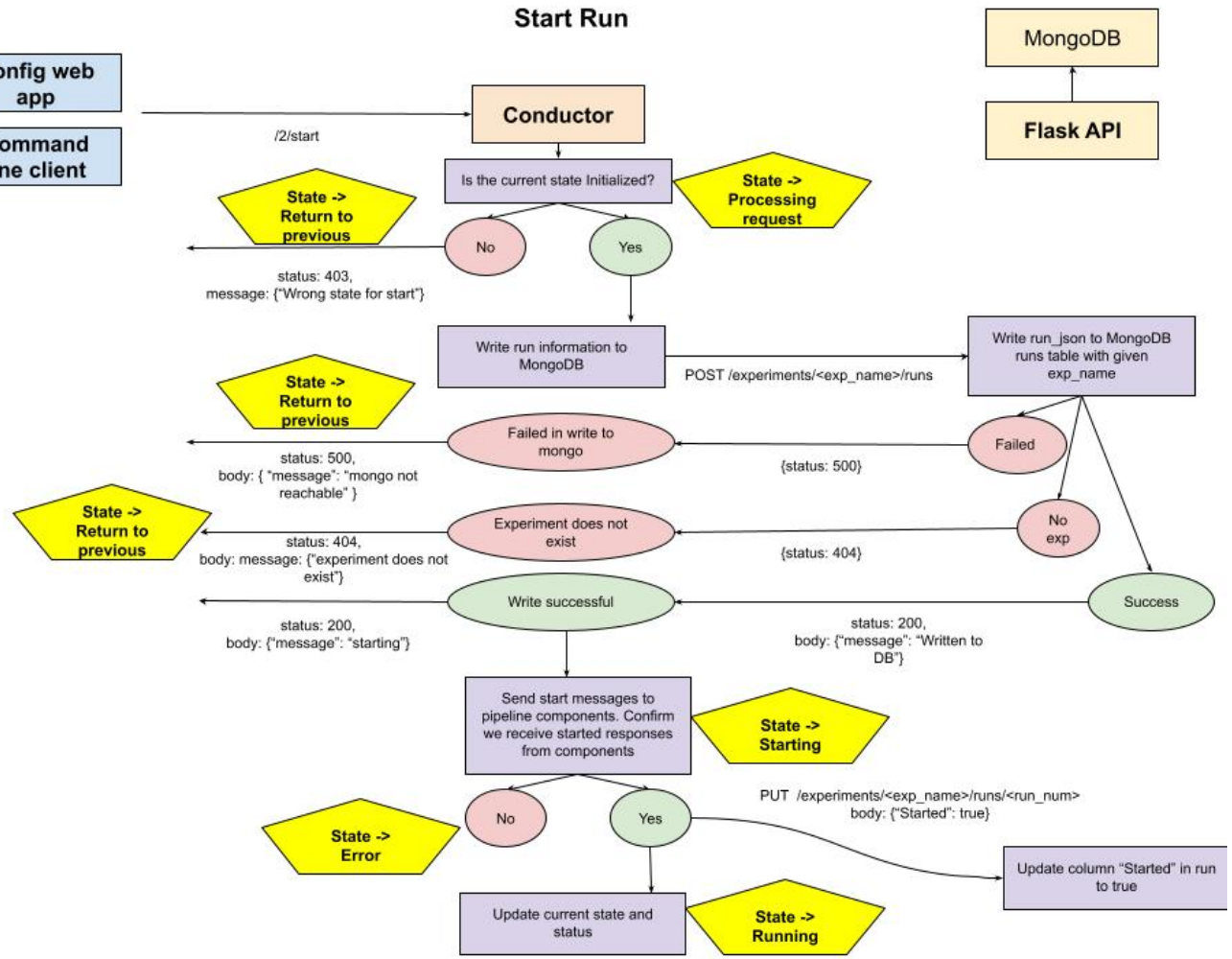
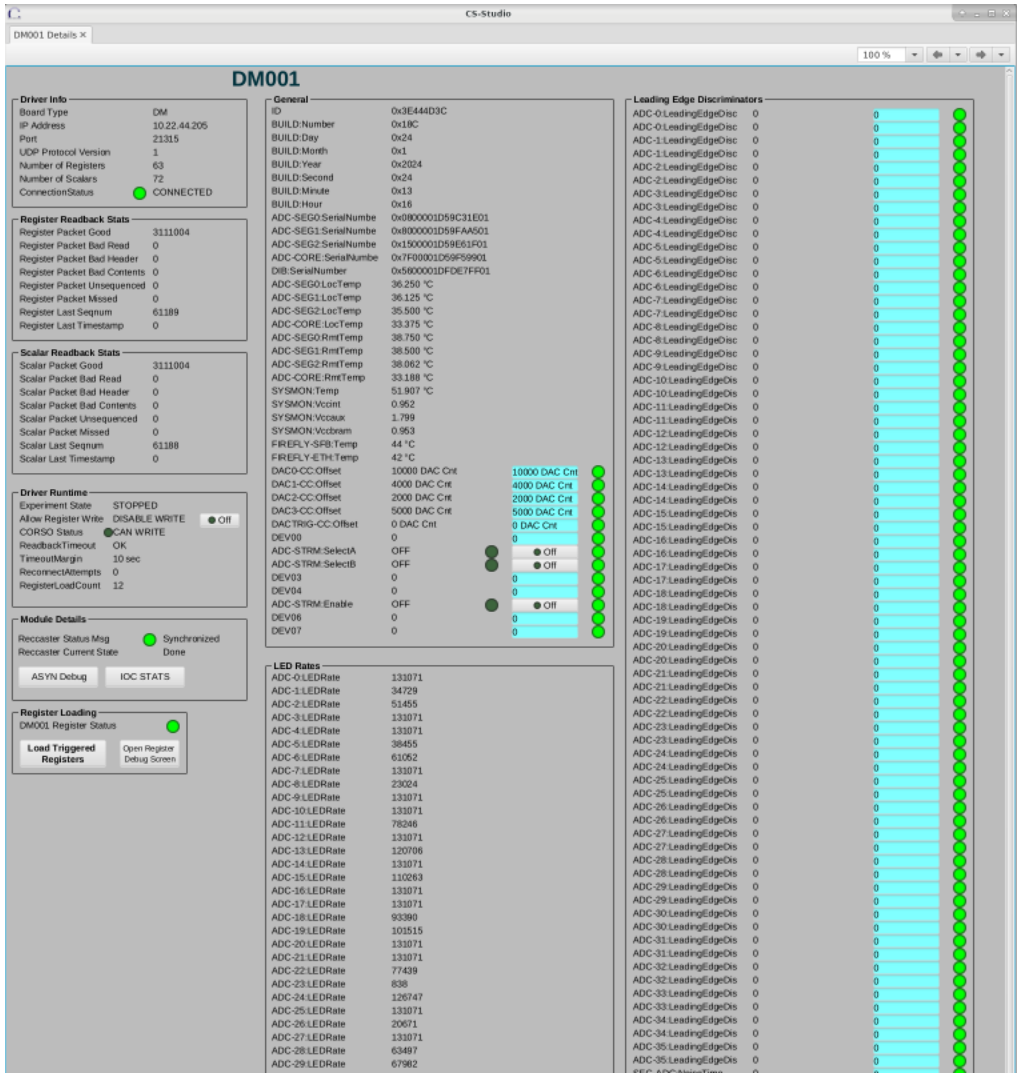


Diagram showing State Transitions / API calls for Run Start

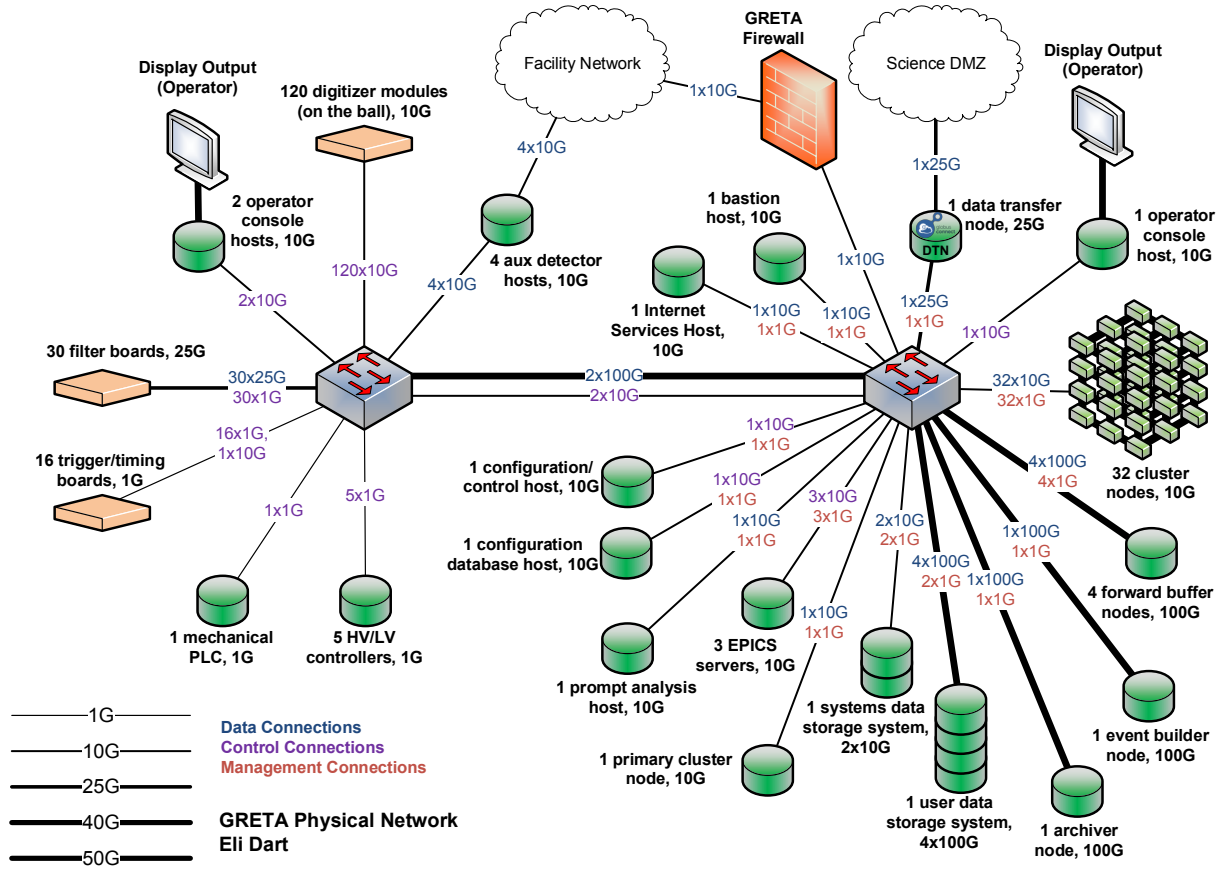
EPICS-based Electronics Controls

- Electronics controls are implemented using EPICS soft-IOCs running on Linux host(s) in Docker containers for flexibility
- Developed specialized driver to send UDP packets to electronics boards for configuration
- Idea is to reduce latency for operators given GRETA's large number of control points
- Driver implements a protocol by which parameters can be loaded synchronously (batch) or asynchronously (immediate)
- Protocol validated in testing and applied to digitizers and filter boards so far



Digitizer Module Engineering Control Screen

Deploying Streaming Today - Co-located Computing



Service	Production environment internal name	Deployment environment	Host service runs on
physical host for VMs	infra0.greta.local	physical	infra0
physical host for VMs	infra1.greta.local	physical	infra1
warewulf provision server	wwulf.greta.local	VM	VM on infra1
vpn	vpn.greta.local	VM	VM on infra0
ssh	ssh.greta.local	VM	VM on infra0
radius server (FreeIPA)	auth.greta.local	VM	VM
docker registry server	registry.greta.local	docker	container on s06
grafana	grafana.greta.local	docker	container on s05
prometheus db	prometheus.greta.local	docker	container on s05
smtp	smtp.greta.local, mail.greta.local	VM shared	service. (VM on infra1)
ntp	ntp.greta.local	VM shared	service (VM on infra1)
syslog	service.greta.local	VM	service(VM on infra1)
CI*	n/a	VM	greyhound.lbl.gov (VM on beagle.lbl.gov)

Is there a better way?

Geographically Distributed Pipelines

- Changes in landscape:
  - Advances in optical networks -100Gb local switched networks common-place, >400Gb wide area networks (ESnet)
  - HPC facilities provide unprecedented compute capability

- The Idea - Distribute pipeline components on multiple experimental and HPC facilities
  - Avoid the need for complex hardware and software deployments at experiments - make it cheaper, faster
  - Send events (not files) with seconds of total latency
  - Make pipelines ‘composable’ across facilities so pipeline components can run where resources are appropriate (compute, storage)
  - Enables the application truly large computing resources to real time steering of experiments

GRETA/Deleria Project

- Goal:
  - *Demonstrate high-bandwidth, event-by-event readout with significant inline processing at remote HPC facilities (OLCF/ORNL) on interactive timescales.*

- Basic idea is to `externalize` GRETA's forward buffers - serve as edge devices to national wide area networks

- Multi-area LDRD (Physical Sciences and Computing Sciences at LBNL) in collaboration with staff at OLCF/ORNL

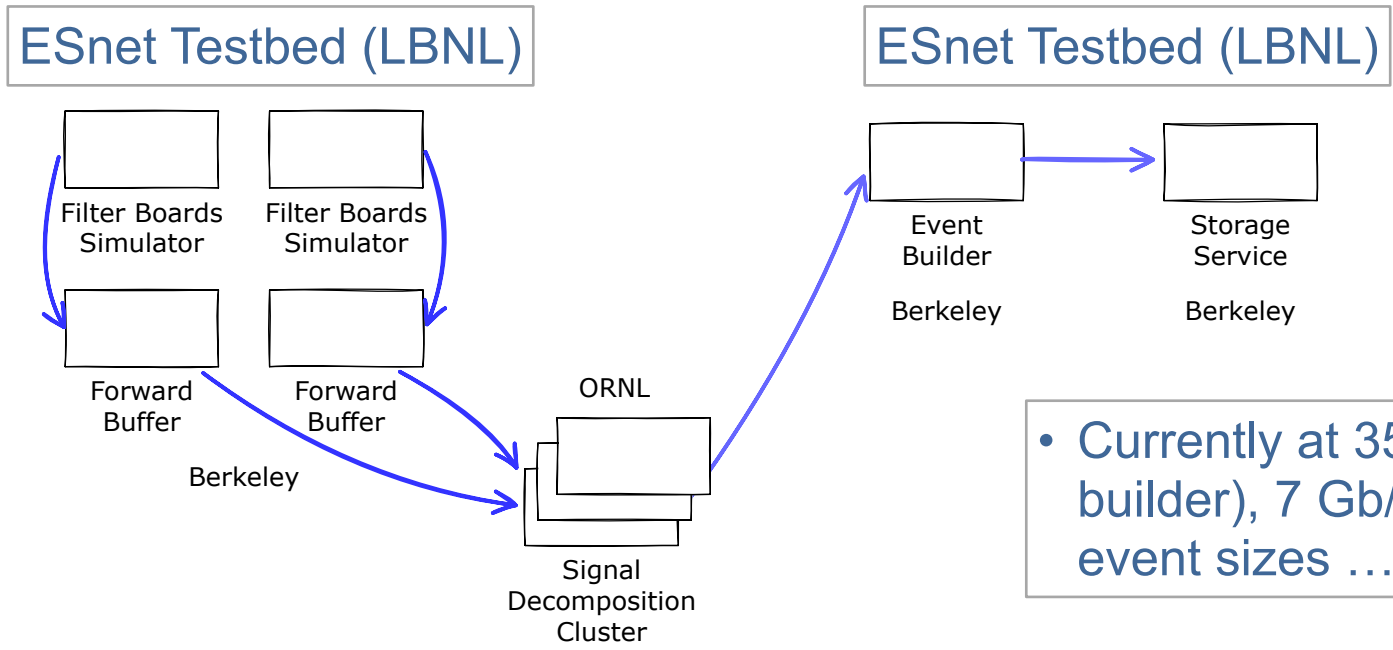
Deleria - Distributed event-level experimental readout & inline analysis

Utilizing DOE ASCR Facilities

- ACE IRI testbed (ORNL)
  - Use the Defiant cluster (Frontier demonstrator)
  - Use Slurm (via JANUS/DTNaaS) for container start, Greta/Deleria conductor for pipeline config
- IRI ESnet testbed (LBNL)
  - Nodes to run non-computational intensive pipeline components + high-performance storage
  - Uses JANIS/DTNaaS for container management
- Established layer 2 connection between the two testbed systems (currently 40 Gb/s, can be easily upgraded to 100 Gb/s)
- Geographical separation allows us to test the impact of high-latency links (RTT = 110ms)

Testing/Optimization

- Utilize GRETA filter board simulators to replay experiment events into forward buffers
- Use Deleria-decomp container to test thus far - currently installing ORNL developed GRETA decomp component which was recently qualified.



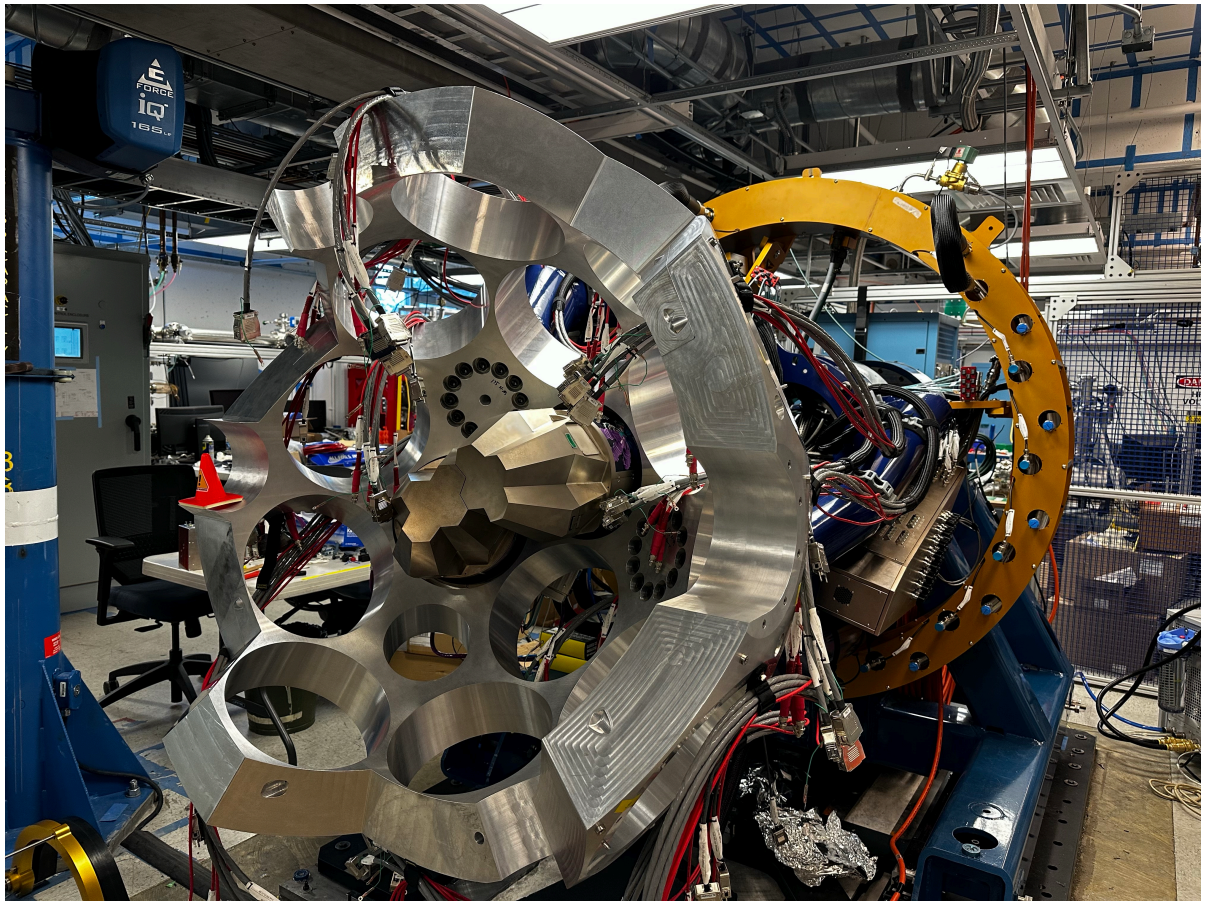
RTT = 110 ms

~50k events in-flight at GRETA design rate 🙂

- Currently at 35 Gb/s (to event builder), 7 Gb/s (to storage), 8 kB event sizes … currently tuning

Near Term Plans for Deleria

- Continue to scale testbed experiment to ORNL (> 40 Gb/s)
- Live detector testing to ORNL following GRETA System assembly at LBNL
- TCP ingest at forward buffer for large events
- Proxies!! Bridge across HPC security boundaries. Working with SciStream group at ANL.
- 'T', 'Labeling/Formatting' pipeline components (ML applications)



GRETA Assembly in 88 K-area

The Team (GRETA + Deleria)

- Eric Pouyoul, Eli Dart, Kiran Vasu, Ezra Kissel, Seyoung Yu - ESnet, LBNL
- Gustav Jansen - NCCS / Physics, ORNL
- Tin Ho, John White - Science IT group, LBNL
- Tynan Ford - Engineering division, LBNL
- Ross Miller - OLCF, ORNL
- Mario Cromaz - Nuclear Science Division, LBNL