STREAM: A Scalable Federated HPC Telemetry Platform

Ryan Adamson, Tim Osborne, Rachel Palumbo, Corwin Lester

National Center for Computational Sciences (NCCS)

Oak Ridge National Laboratory (ORNL)

CUG 2023

Oak Ridge National Laboratory

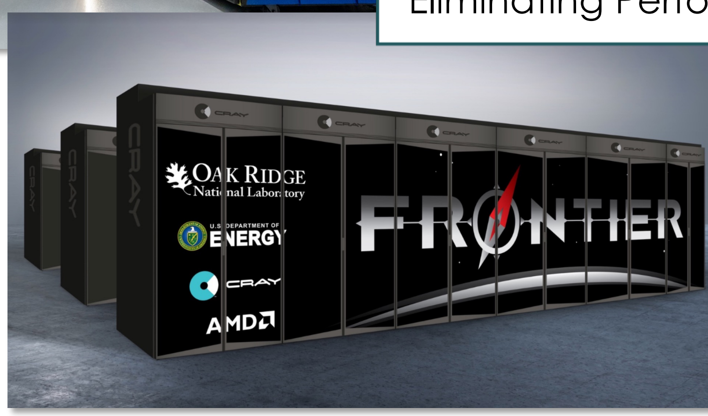
Roadmap

1. **Observations and Motivation**
2. Design Requirements
3. Architecture
4. Lessons Learned
5. Future Work

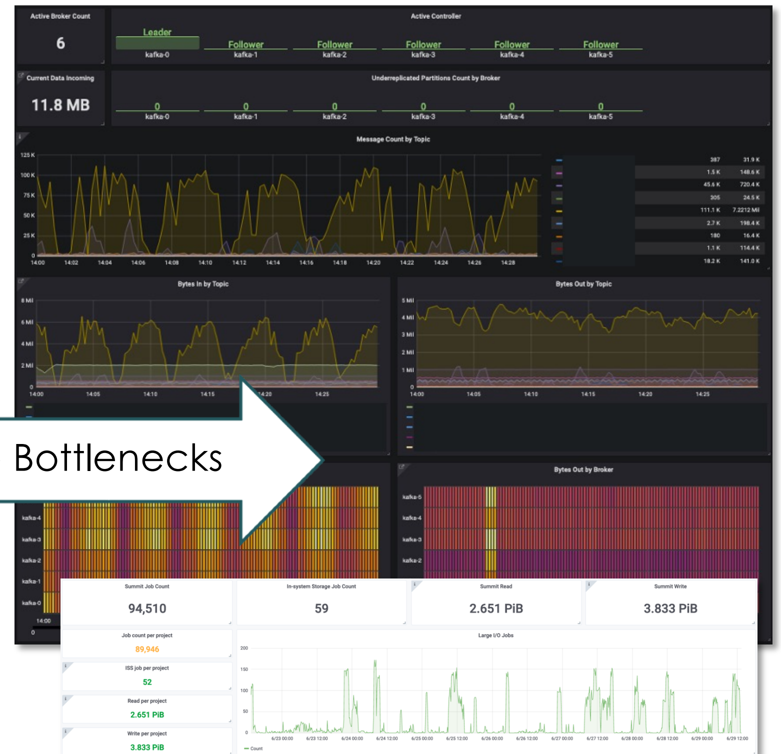
Observations and Some Motivation

1. ***Everyone*** wants to collect and access your data!
2. Collecting data can adversely affect performance
3. Providing data to consumers implies an endorsement of support
4. Systems staff are the gatekeepers of system data
5. Best practices don't ***really*** exist
6. AI/ML workloads and tools are becoming more prevalent
7. Systems staff don't have data science skills

Hardware and Application Monitoring

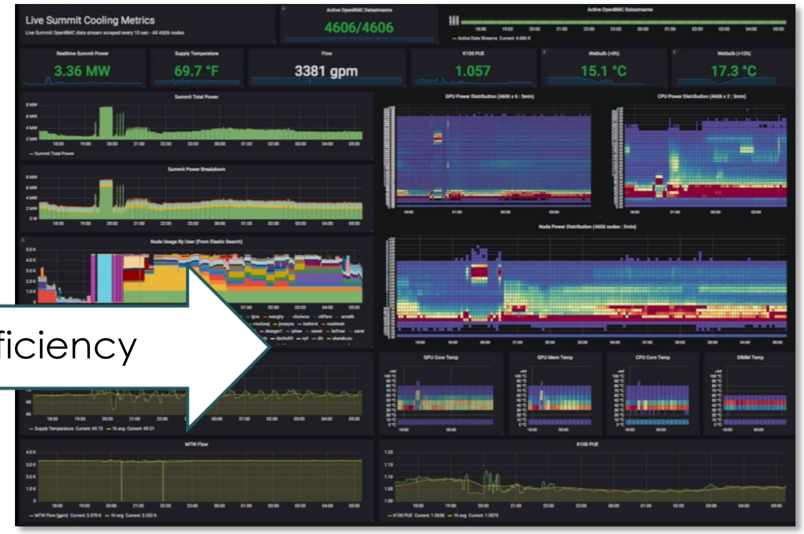


Eliminating Performance Bottlenecks

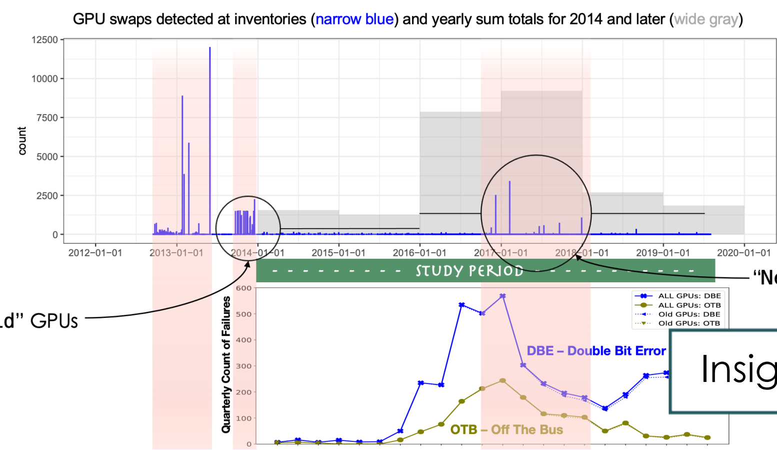


Power, Water, Cooling Infrastructure Analytics

Understanding Infrastructure Efficiency



Event Log Monitoring and Failure Analysis



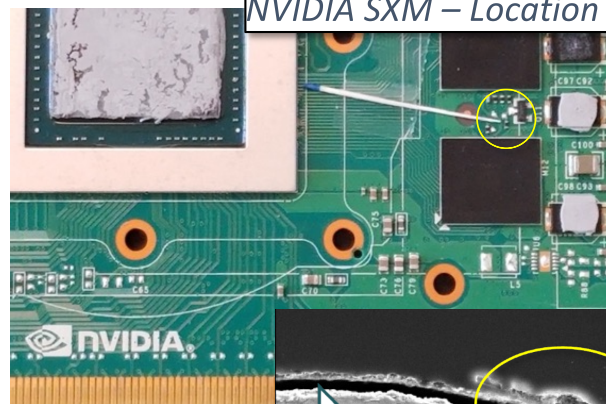
“Old” GPUs

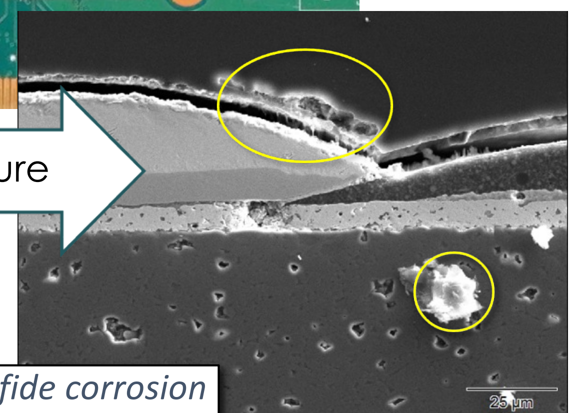
“New” GPUs

Anti-Sulfur Resistors (ASR) were not appropriately used in Titan GPUs

Insights into Component Failure

NVIDIA SXM – Location of a non-ASR





Silver-sulfide corrosion

"Flowers-of-Sulfur"

The Data Engineering ‘Data Transformation Lifecycle’

Data Collection

- Identification of Data Sets
- Data Acquisition
- Monitoring

Exploratory Data Analysis (EDA)

- Treatment of Outliers / Missing Data
- Elimination of ‘Junk’ Fields
- Improvements to Overall Quality

Feature Selection / Engineering

- Augmentation of Data
- Transformation of Fields
- Storage and Maintenance

Useful Data

- Reporting
- Analysis
- Model Training

The Data Engineering 'Data Transformation Lifecycle'

Data Collection

- Identification of Data Sets
- Data Acquisition
- Monitoring

Exploratory Data Analysis (EDA)

- Treatment of Outliers / Missing Data
- Elimination of 'Junk' Fields
- Improvements to Overall Quality

Feature Selection / Engineering

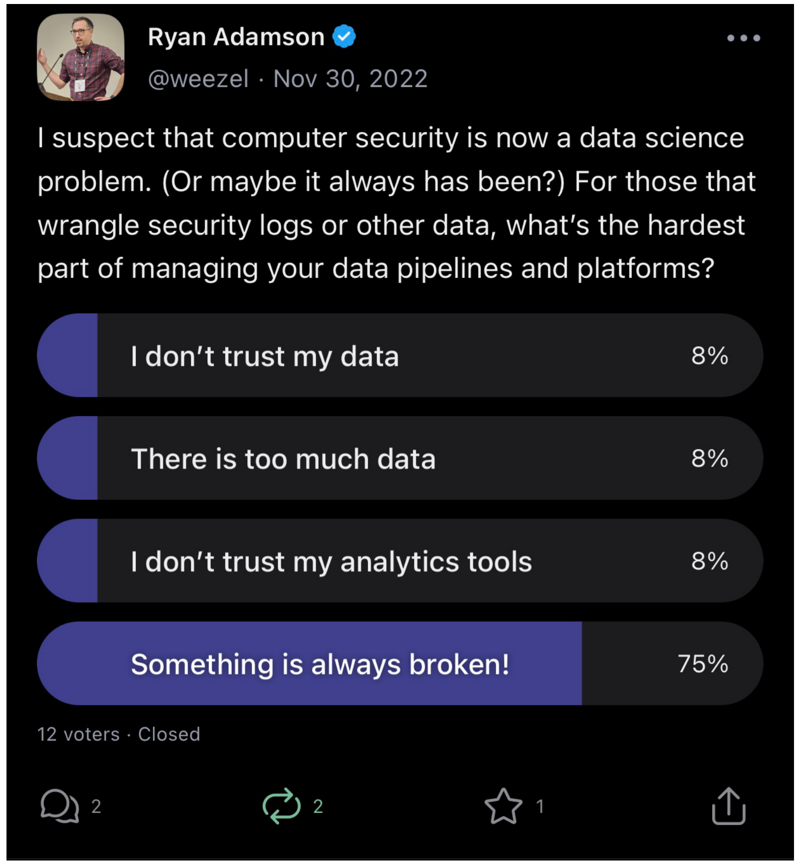
- Augmentation of Data
- Transformation of Fields
- Storage and Maintenance

Useful Data

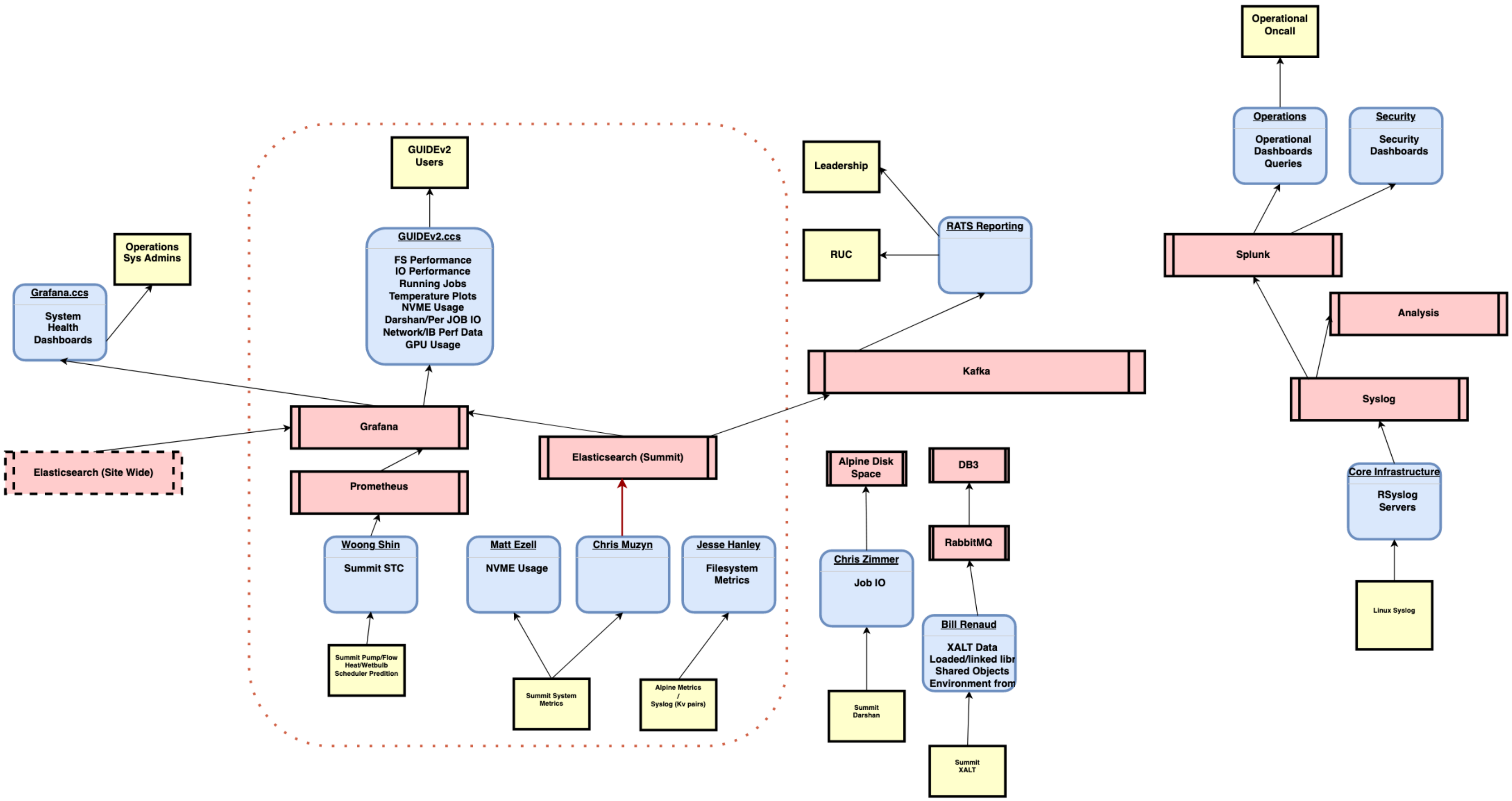
- Reporting
- Analysis
- Model Training

80% of data engineering effort is spent in the 'data wrangling' stages of the lifecycle

The Data Engineering 'Data Transformation Lifecycle'

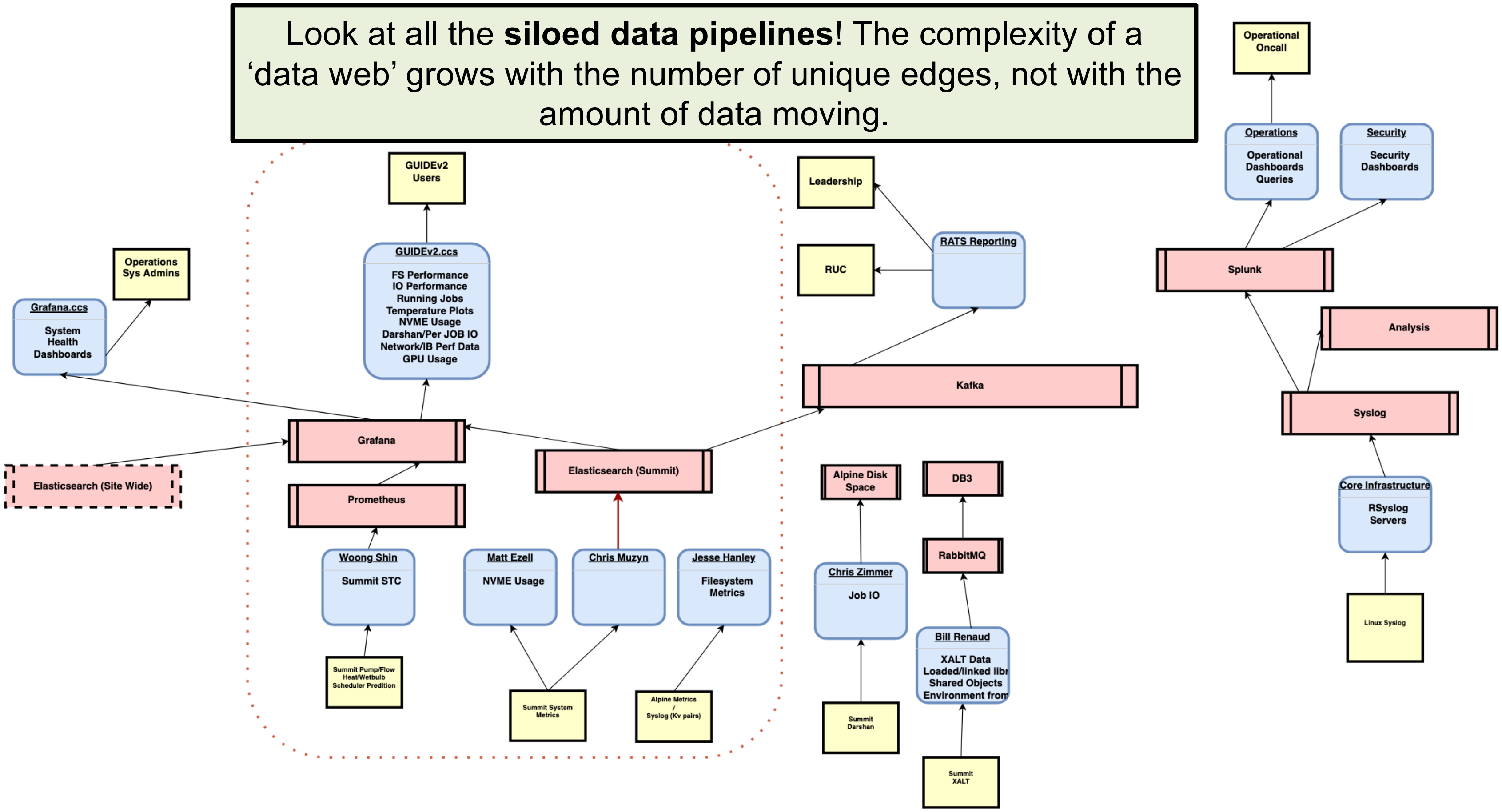


2019: NCCS Analytics and Monitoring ‘Platform’



2019: NCCS Analytics and Monitoring ‘Platform’

Look at all the **siloed data pipelines**! The complexity of a ‘data web’ grows with the number of unique edges, not with the amount of data moving.



OAK RIDGE National Laboratory

Open slide master to edit

2019: Data Platform Observations

'Scaling' was no longer scaling

- Increased system complexity and changes to systems and schemas over time made data analytics incredibly manual
- We needed to replace batch processing and enable stream processing where possible
- Traditional data sinks did not provide flexibility of modern data warehouses for applications users wanted to use
- A central data bus was necessary to decouple opaque data pipeline sources and sinks and provide O(n) scaling

New technologies made this possible

- Several scalable, robust, flexible message busses were becoming mature
- Modern data warehouse designs and search/analytics tools like Elastic were being explored by various teams
- Data analytics tools were maturing and our operations teams were becoming more capable of slicing and dicing telemetry streams
- Platform as a service (PaaS) had just been deployed within NCCS and was reducing administrative burden

STREAM Design Goals

1. Ease the burden of telemetry pipeline management @ OLCF
2. Be able to scale up with the deployment of new systems
3. Provide a centralized point of service for streaming data for all NCCS programs
4. Provide a data-agnostic abstraction layer for data consumers

Roadmap

1. Observations and Motivation
2. **Design Requirements**
3. Architecture
4. Lessons Learned
5. Future Work

Ease the Data Burden

- We needed to assume operational responsibility for pipeline stewardship
  - Develop and enforce best practices and documentation
  - Deploy and monitor pipelines
  - Provide technical expertise for data producers and consumers
- We wanted to perform as much of the ‘80%’ of EDA and data wrangling as is reasonable
  - Additionally, develop data science and data engineering expertise
  - Become data liaisons to broker insight about system behavior

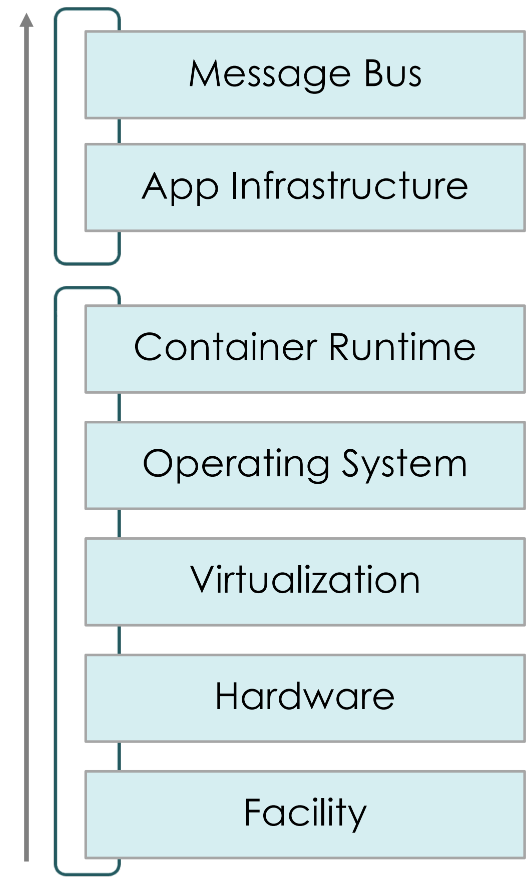
Data Platform Strategy

Assume Operational Responsibility for Data

- Use operational / SRE best practices to provide data assurance
- Reduce the ‘data wrangling’ that scientific end users have to do
- Be advocates for both data producers and consumers
- Inform institutional data policy and help resolve data ownership conflicts

Application Stack Requirements

- Deliberately focus on ‘top of stack’ to support applications, platform users, and data analysis
- Utilize PaaS for supporting layers to minimize complexity
- Cleanly and clearly define data pipeline roles and responsibilities between consumers and producers



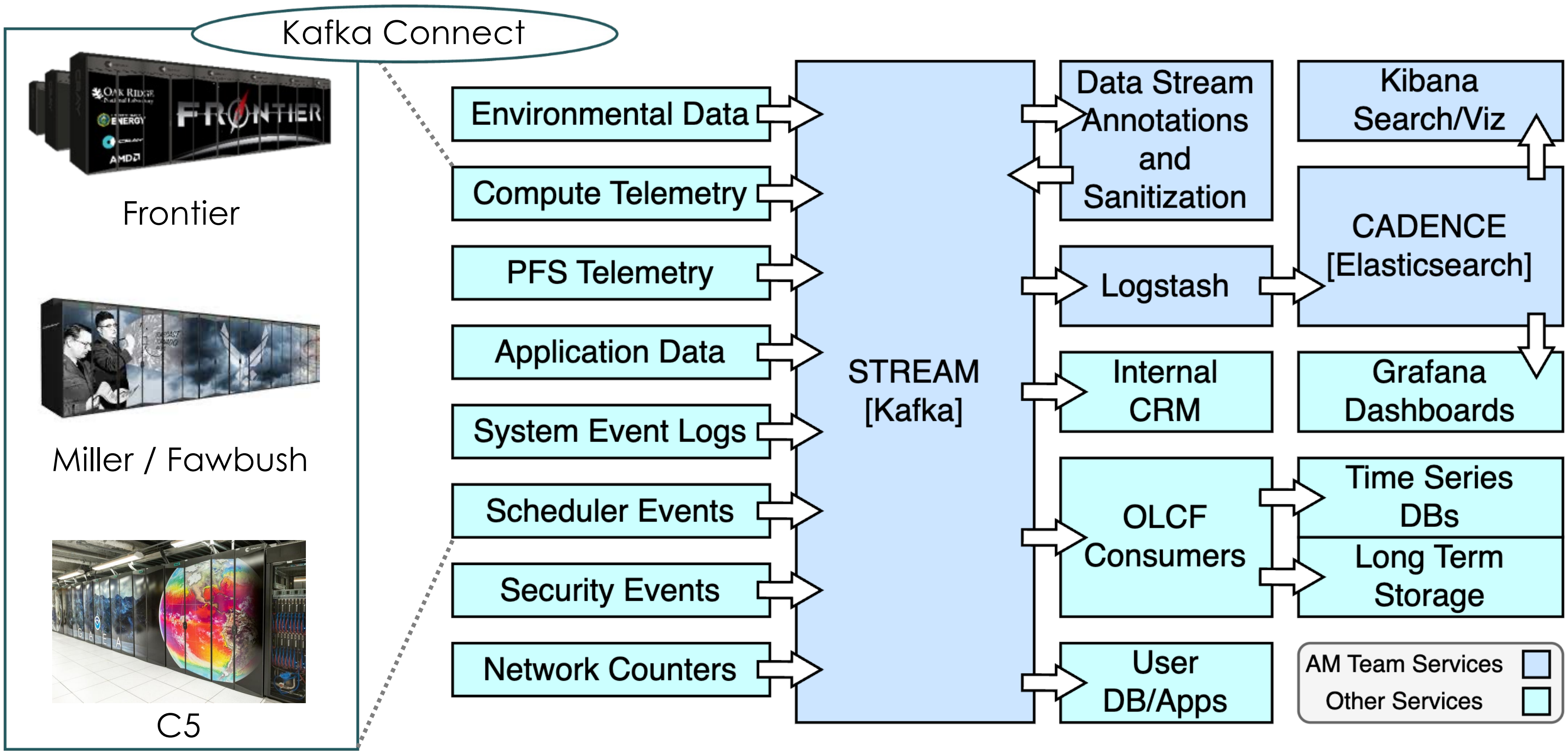
Performance and Sizing

- The bottleneck with streaming telemetry platforms are not what we're typically used to in HPC!
  - 20PB of telemetry data generated over 5 years is ~11TB / day
  - This is *roughly* one gigabit ethernet link's worth of pipe
  - Fast disks and network connections are primarily for failover/recovery of under-replicated partitions
- Consumer behavior drives decision making
  - We expect consumers to usually be 'caught up'. Incoming messages will either be forwarded immediately or will most likely exist in application memory or page cache
  - "Data Lakes" are a *consumer* of STREAM: We don't need big disks here!

Roadmap

1. Observations and Motivation
2. Design Requirements
3. **Architecture**
4. Lessons Learned
5. Future Work

STREAM Architecture in 2023 (Information View)

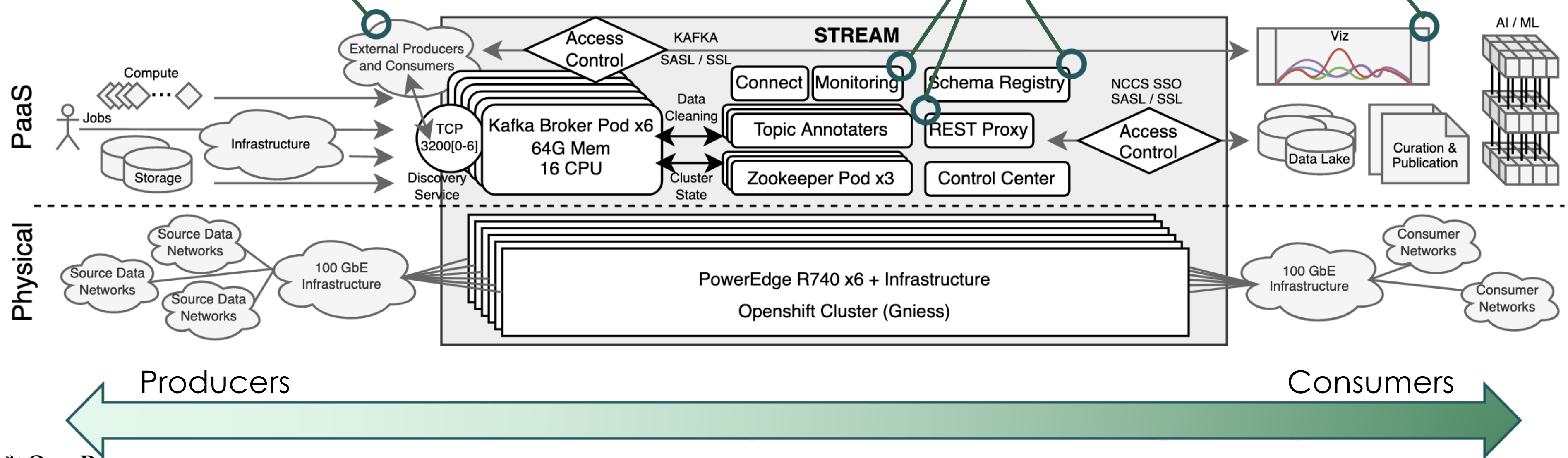


STREAM Architecture in 2023 (System View)

Producers and consumers configure Kafka clients to connect to a discovery service using SASL auth and TLS

Auxiliary services like the REST proxy, schema registry, Kafka connect, and sidecars do not require significant resources

Grafana dashboards monitor Kafka brokers and topics while Confluent Control Center is used to examine topic specifics.



Roadmap

1. Observations and Motivation
2. Design Requirements
3. Architecture
4. **Lessons Learned**
5. Future Work

Data Platform Challenges

Sustainability

- Data sources ***will change*** over time
- Systems will come and go and technology will change
- Technical debt can be difficult to reduce once accrued
- Once automation exists for production and consumption… **good luck**!

Documentation

- Data producers should, ***in theory***, be the best equipped to answer questions about data sources
- Data consumers typically ***don't have enough context*** to understand the information they receive through telemetry pipelines

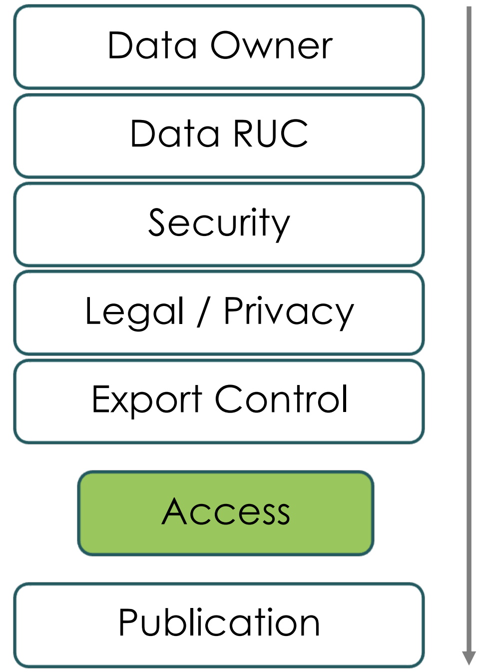
Performance / Robustness

- Controlling types and sizes of data can be challenging – ***data throughput tends to grow over time***
- Monitoring individual topics can be difficult, especially when ***a few key topics dominate*** systems engineer time

Lessons Learned – Access Control

Controlling Access

- We define a data ‘owner’ to be the producer of data, and we give some control over who can access streaming messages.
- Data ‘consumers’ apply for access and are granted individual topic credentials based on need.
- The OLCF has an interest in reviewing potential research outcomes and discoveries
  - Misinterpretation of information is quite common!



Lessons Learned – Topic Naming

Topic Naming

- Changes to topic names as well as changes to client configuration is very difficult to manage
- We developed a sustainable topic naming scheme based on use cases
- NCCS uses a delimited topic name 'tuple' based on data source owner, system name, the subsystem that produces messages, and the specific topic subject the topic is about
- Example:

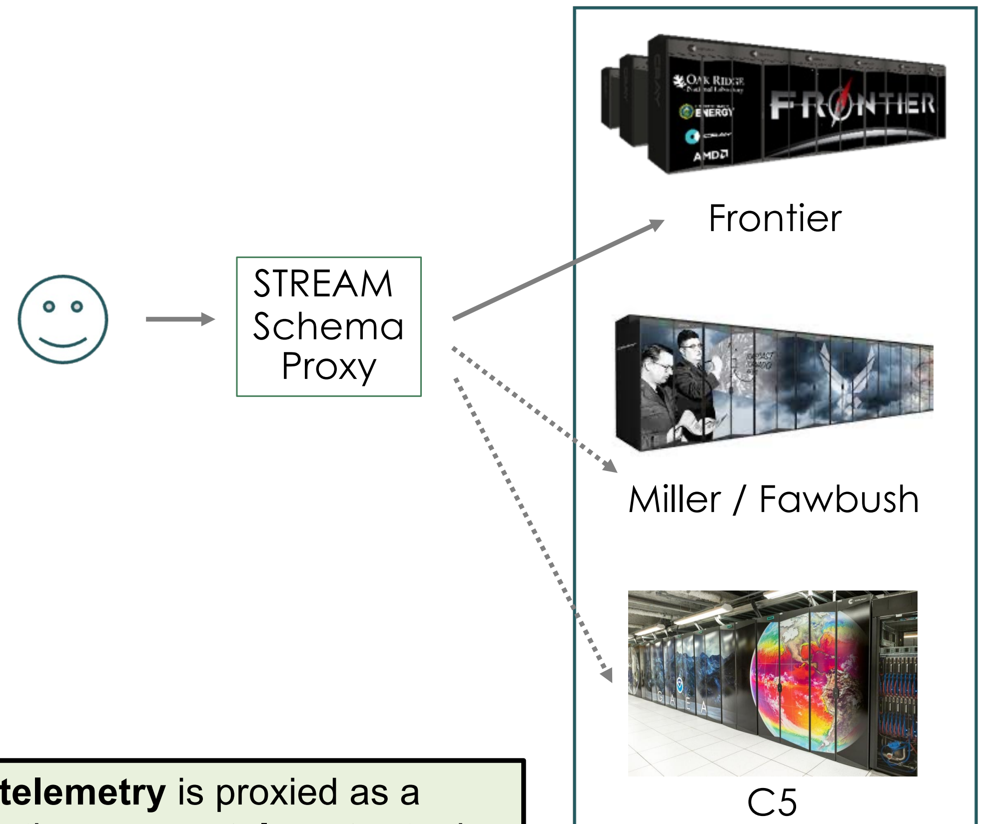
stf002hpc.frontier.hpcm.crayex_telemetry

Source System	Subsystem	Topic Subject
frontier	hpcm	HPCMLOG
c5		SYSLOG
t5		crayex_alerts
miller		crayex_telemetry
fawbush		event_cooldev
ace		hpcm_inventory
		hpcm_inventory_dimm
		log_iml
		powerservice_operations
		powerservice_rawpower
		sensors_node
		slurm_jobs
		…

Lessons Learned – Schema Registries

There are many, many registries!

- HPCM Kafka schema registries on various systems may not be configured in the same way
  - The ordering of topics and versioning of topics over time lead to different schema definitions for the 'same topic' across systems
- OLCF developed a fairly simple flask application to 'proxy' schema registry access
  - On client access to STREAM, schema registry request is modified to connect to HPCM schema registry of the system the topic is produced from



A user request for **stf002.frontier.hpcm.crayex_telemetry** is proxied as a request to the Frontier schema registry service for the **crayex_telemetry** topic.

Roadmap

1. Observations and Motivation
2. Design Requirements
3. Architecture
4. Lessons Learned
5. **Future Work**

Future Work

- Automation of Topic Naming and Access
  - End-user based access
  - Ephemeral topics (Automated creation and deletion)
- Development of training and easy to use examples
- Automating EDA for topics and other data exploration tools
- Developing a common 'schema' or Entity Relationship Diagram for HPC specific information
- Broadening scope to become a streaming I/O platform from external data sources or to external data sinks
- Embracing lossy compression?

STREAM Summarized

Performance and Scalability

- Our **narrow waist** design has been very successful
- Apache Kafka is very scalable and can operate quite naturally in a federated way (with one or two caveats)
- Scalable units of STREAM are topics and topic partitions as well as brokers

Lessons Learned

- Be sure to **have deliberate management strategies** for topic creation and consumer/producer access
- Lifting the data burden from operations staff has helped streamline data access processes
- Documentation is never quite good enough, **personal expertise is required** to understand data streams

Future Work

- **Automation** of some system management functions will help pay down technical debt
- Creation of and standardizing on an ERD within industry partners and labs will help
- Automating some data engineering steps to save time for all users of a data stream

Discussion

Acknowledgements:
This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.