Data Lakes vs Data Warehouses Applications for OOI Cyberinfrastructure

Rob Bochenek
Axiom Data Science
NSF DDCI, October 2020

Axiom DATA SCIENCE

Today’s presentation

- Data Warehouses vs Data Lakes
  - Theory
  - Data Warehouse & Data Lakes in OOI and Axiom
- Motivations for moving OOI towards Data Lake pattern
  - Cassandra
  - Improved Data Access for all parties - humans and computers
- Explore Data Lake Concept for OOI with Low Risk- Low Cost Experiment

Data Warehouse

A data warehouse is a blend of technologies and components which allows the strategic use of data. It is a technique for collecting and managing data from varied sources to provide meaningful business insights.

It is electronic storage of a large amount of information by a business which is designed for query and analysis instead of transaction processing. It is a process of transforming data into information.

- Designed for a very specific purpose, powerful but inflexible.
- Data is Persisted (Stored) in software and data appliances and accessible via standard and custom APIs
- High Structured Data Model (Strong Schema) that is Defined at Write
- Storage and Compute are Coupled
- Extract Transform Load (ETL)

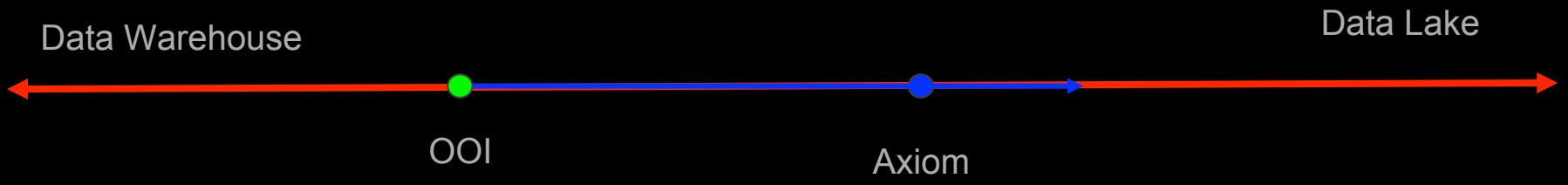
Data Lake

A Data Lake is a storage repository that can store large amount of structured, semi-structured, and unstructured data. It is a place to store every type of data in its native format with no fixed limits on account size or file. It offers high data quantity to increase analytic performance and native integration.

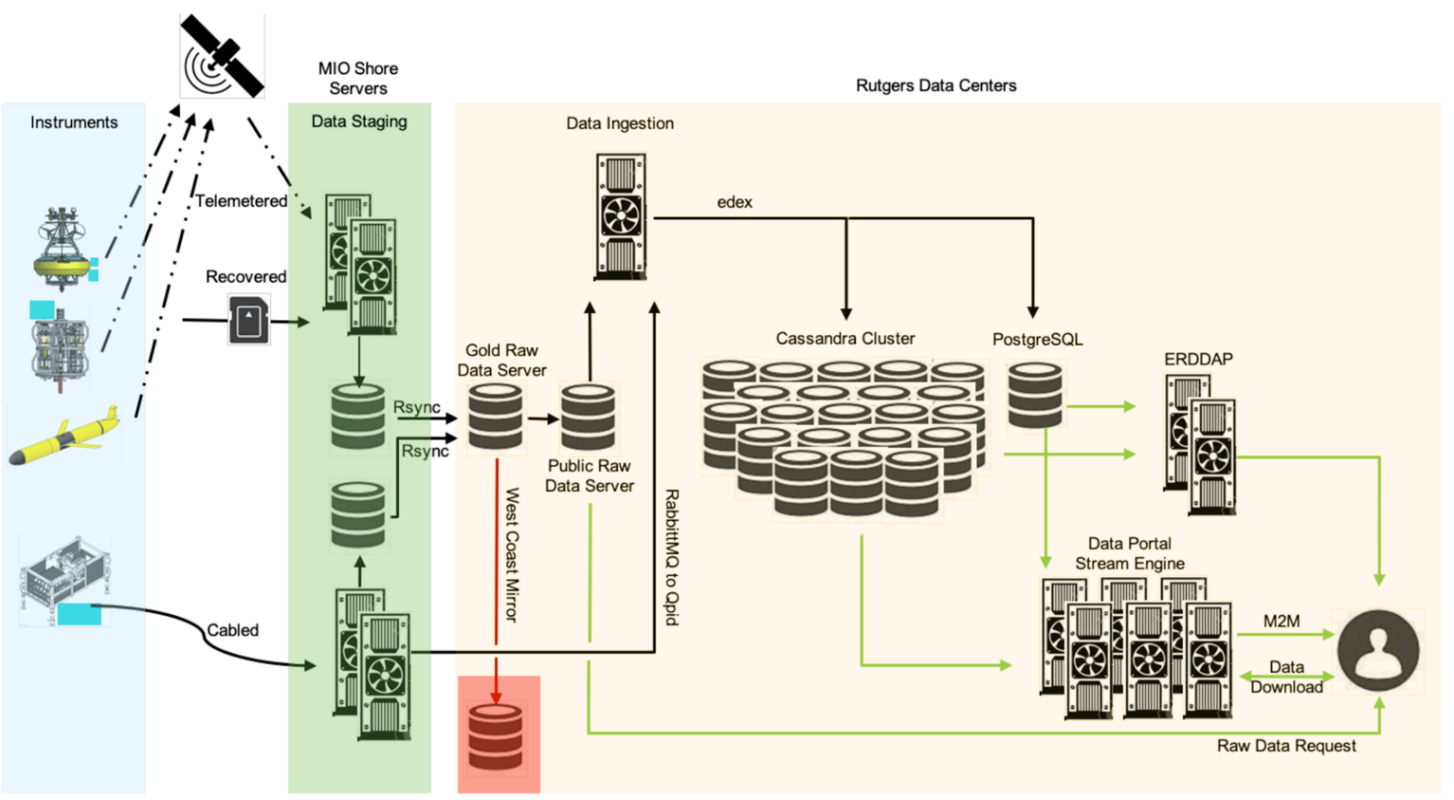
Data Lake is like a large container which is very similar to real lake and rivers. Just like in a lake you have multiple tributaries coming in, a data lake has structured data, unstructured data, machine to machine, logs flowing through in real-time.

- Data is Persisted (Stored) as Files in Various States (L0, L1, L2,...)
- Lowest common access method designed for any generic purpose and openly accessible to software, computers and humans.
- NO Structured Data Model - Schema Defined at Read
- Storage and Compute are De-Coupled - Improving Scalability and Data Pipeline Harmonization
- Extract LOAD Transform (ELT)

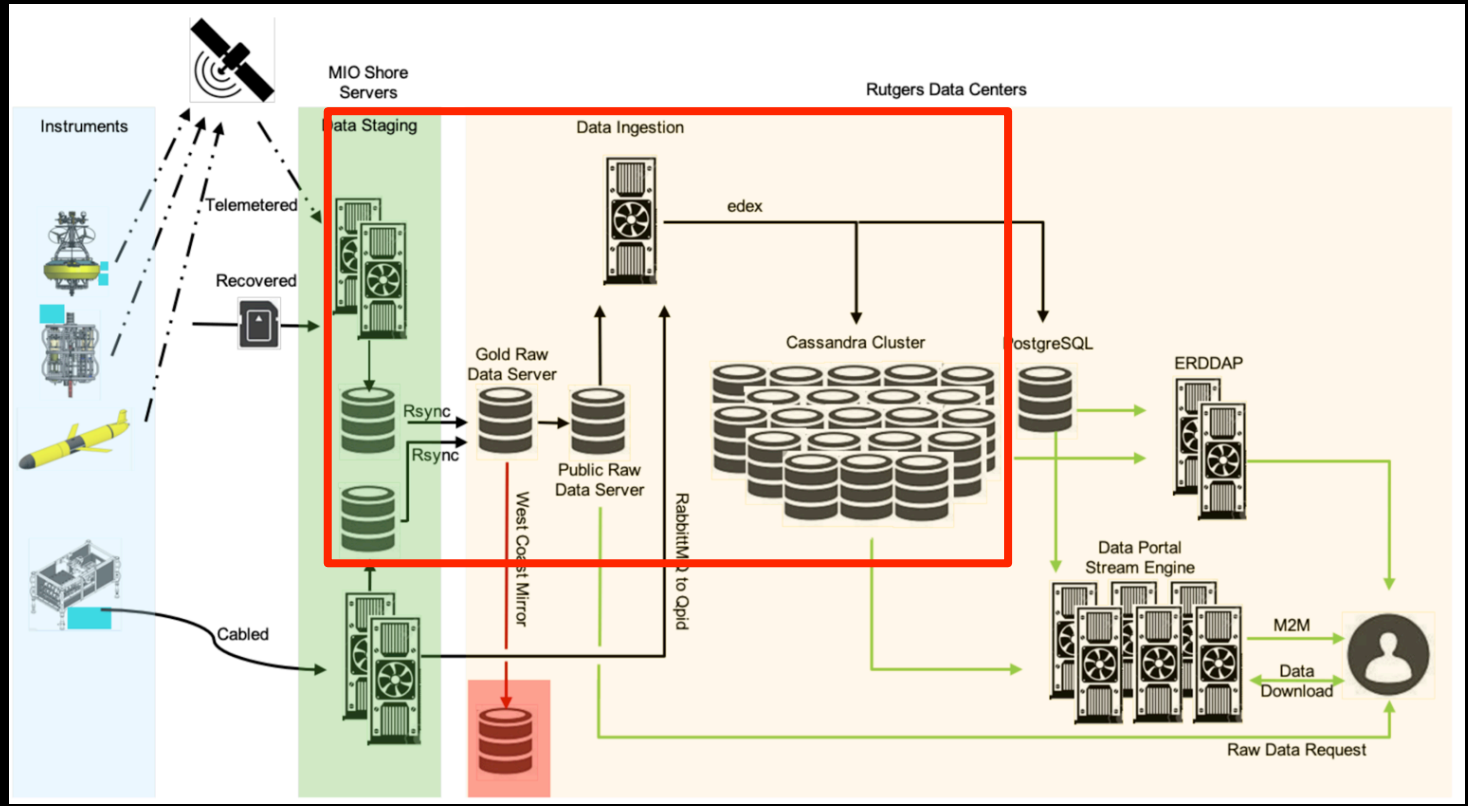
On the Spectrum



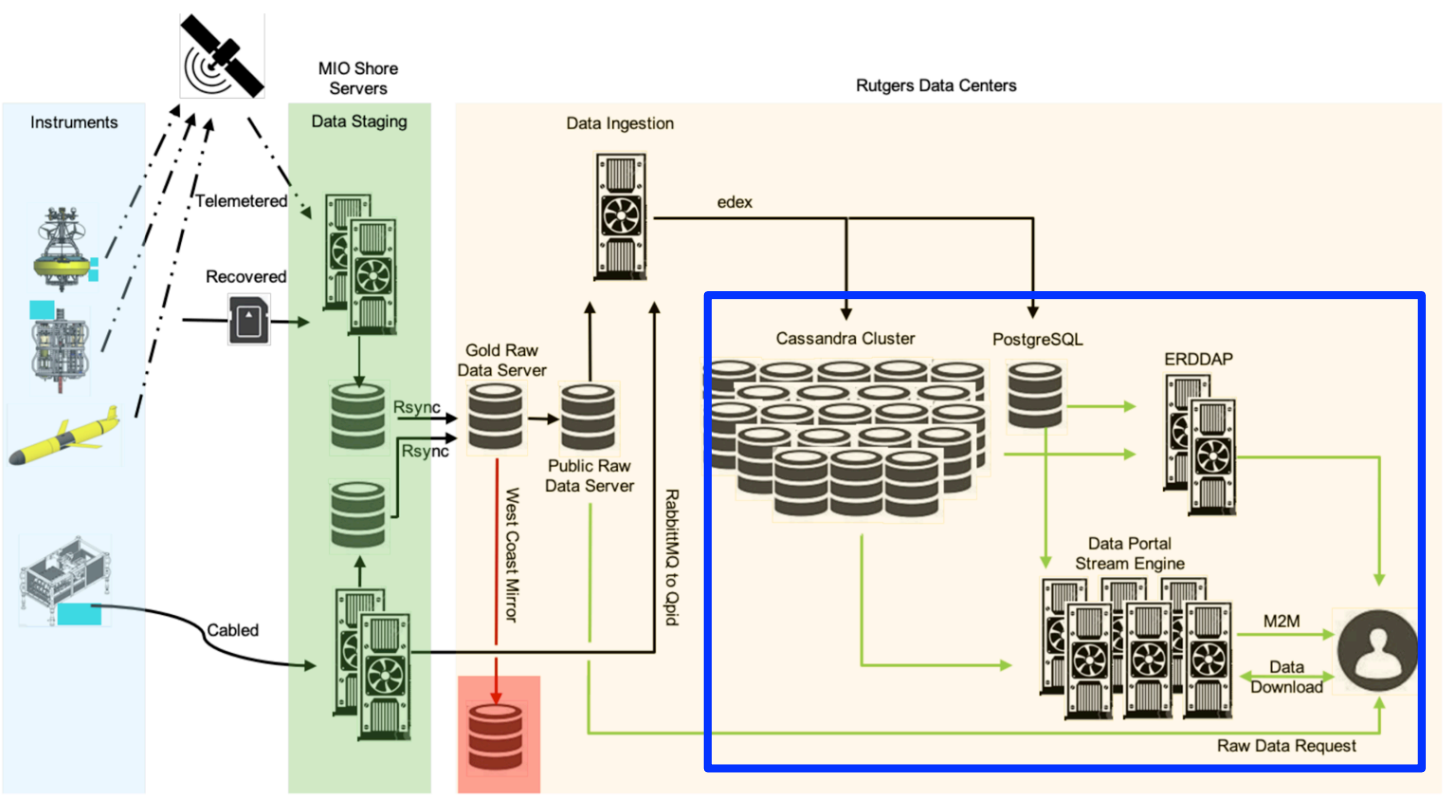
OOI Backend


MIO Shore Servers
Rutgers Data Centers
Instruments
Data Staging
Data Ingestion
Telemetered
Recovered
edex
Cassandra Cluster
PostgreSQL
ERDDAP
Gold Raw Data Server
Rsync
Rsync
Public Raw Data Server
West Coast Mirror
RabbitMQ to Qpid
Data Portal Stream Engine
M2M
Data Download
Cabled
Raw Data Request

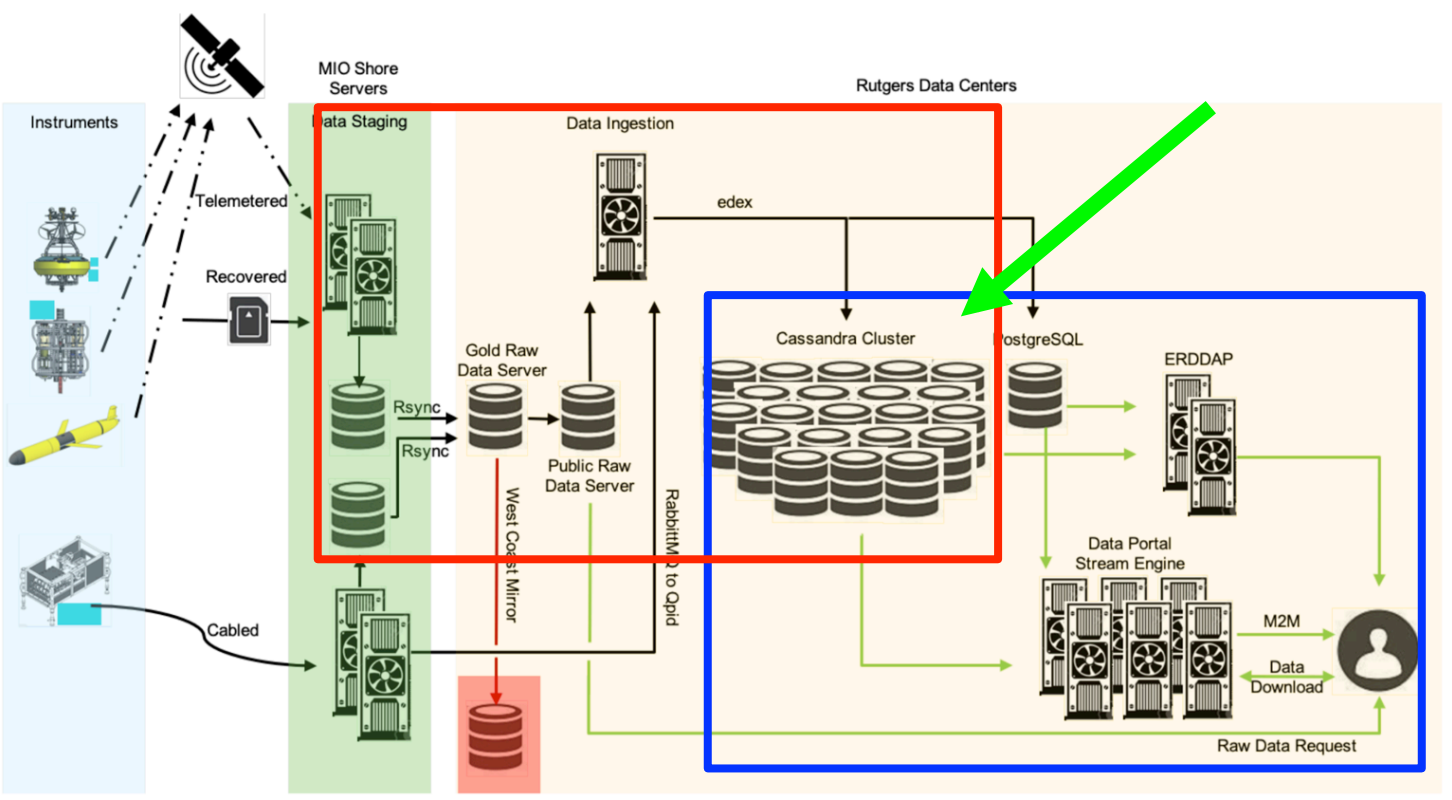
OOI Backend - Data Ingestion (Persistence)



OOI Backend - Data Access



OOI Backend - Cassandra!

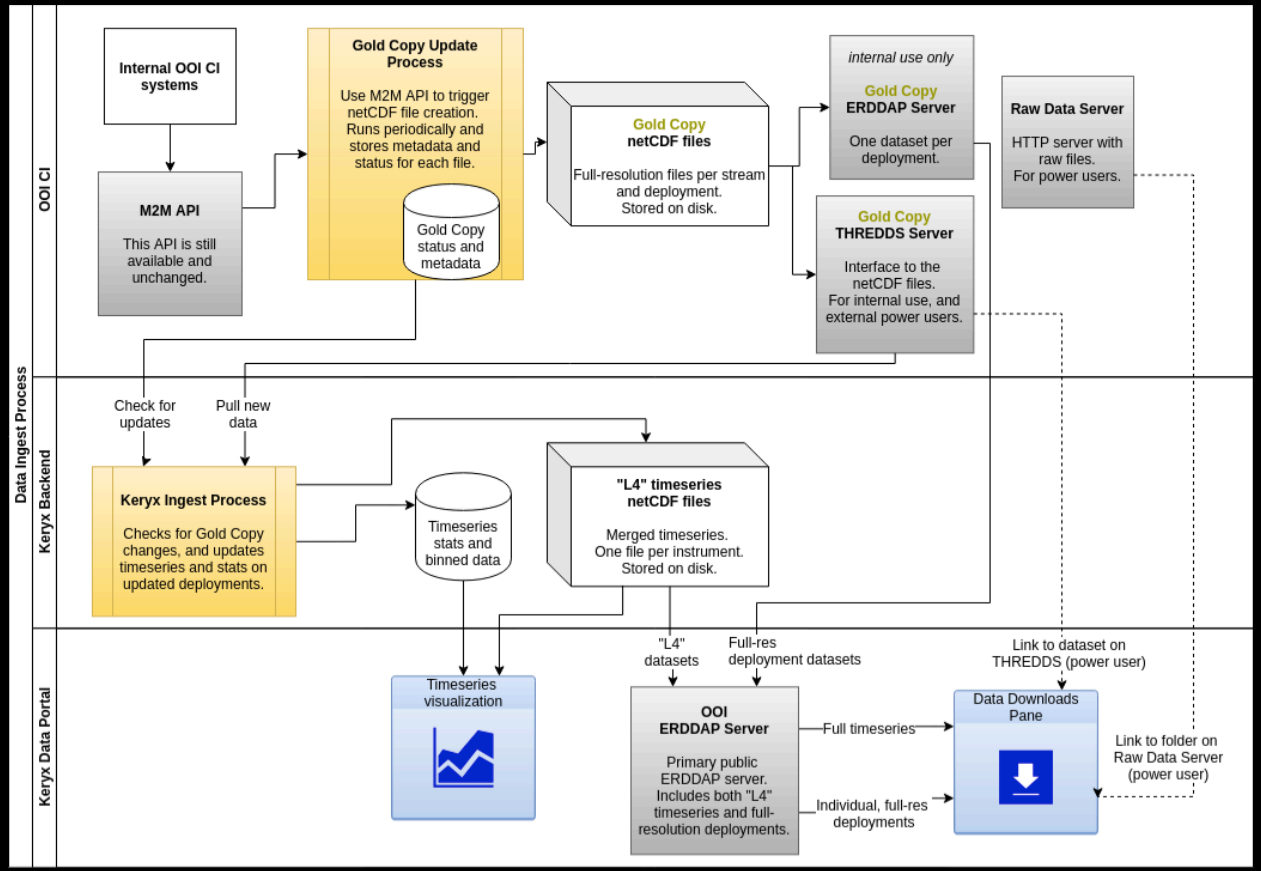


Axiom DATA SCIENCE

Cassandra Conundrum

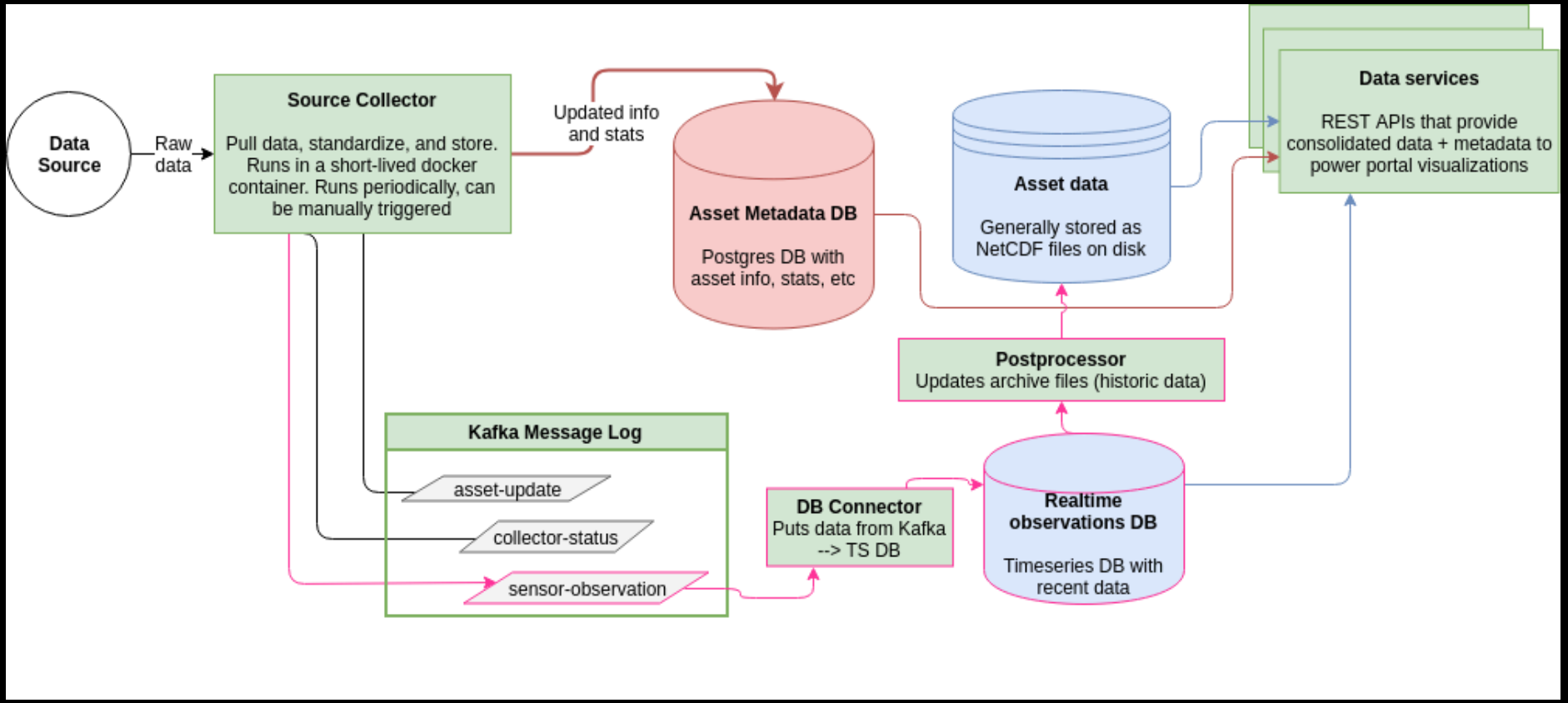
- Current Rutgers Build 21 Nodes
- Axiom estimated 42 nodes to support the system for the next 5 years
- 100+ nodes at end of the OOI project
- 1 FTE to support - someone's full time job keeping this thing running
- Do we need to store all these engineering units is a huge data warehouse?
- Design Flaw? Not critical but surely expensive, complex and leading to heartburn

Axiom Sensor Data Pipeline Bolted on to OOI

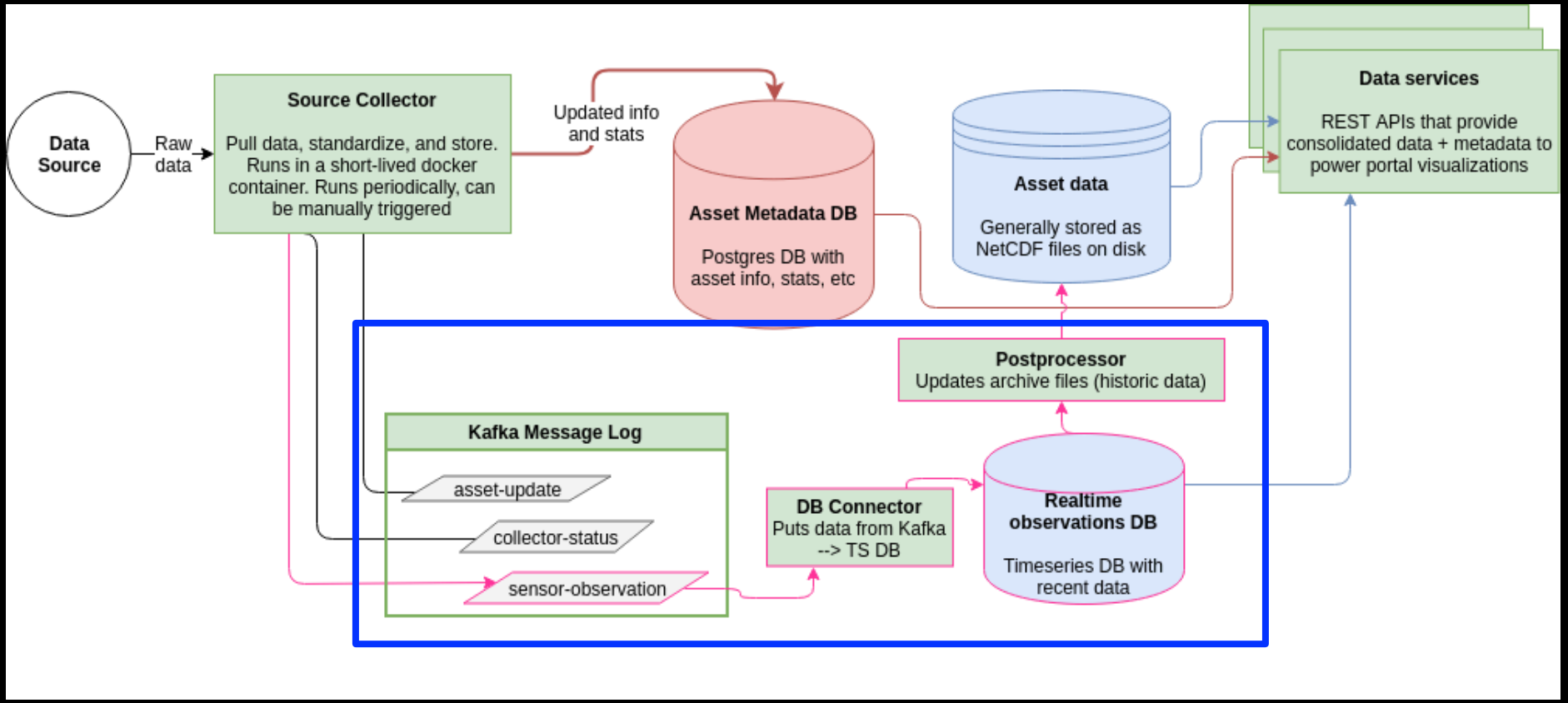


Axiom DATA SCIENCE

Environmental Sensor Data Pipeline

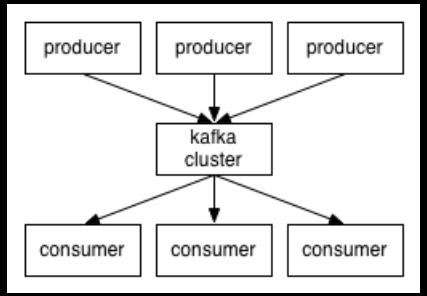


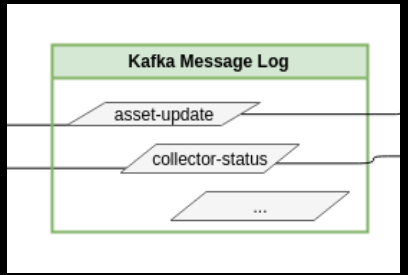
Firehose - DW used as cache, data persisted as files



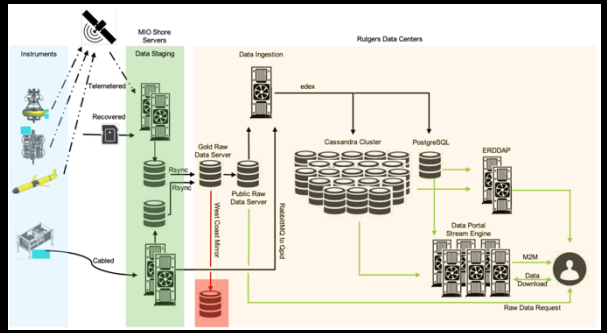
Apache Kafka in Data Pipelines

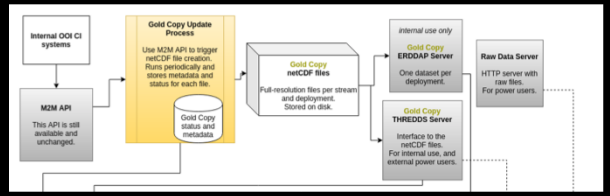
- Kafka is a distributed, publish-and-subscribe messaging system
  - All messages in Kafka are stored on a **topic**
  - Processes that publish messages to topics are called **producers**
  - Processes that subscribe to topics and listen to messages are called **consumers**
  - Each topic has a message **schema** that defines the message structure
  - Consumer pull model; can produce/consume in batches for quick I/O
  - Benefits:
    - Easily decouple processes
      - Producers/consumers don't talk directly
      - Topic is generic, so can push data from anywhere
      - Can scale producers or consumers independently
    - Topic log is history of events (great for debugging)
    - Can handle ridiculous number of messages
  - Downsides:
    - Steep learning curve, complex ecosystem, still in flux
- We use Kafka topics to link together components of our pipelines, and refresh caches that power portal visualizations

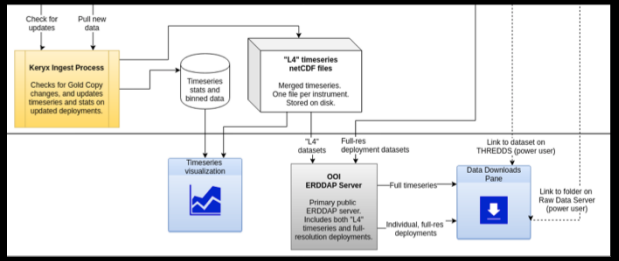




Status Quo



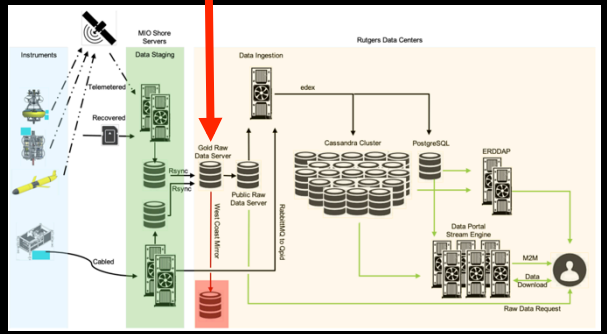


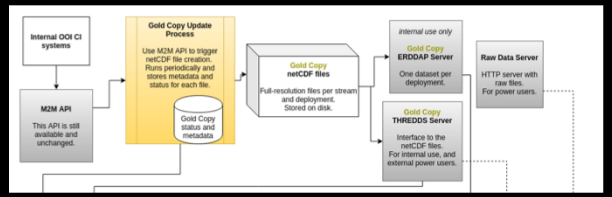


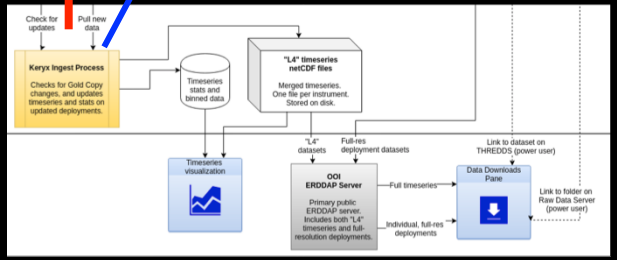
- 4 Unique Software Frameworks for Data Processing
- 4 Different API's For Data Access to Levels of Data (0-4)
- Lots of hardware, partitioned across stovepiped heterogeneous processing engines

Experimental Design

L0/L1, L2, L3, L4 all stored as files







- 1 Software Frameworks for Data Processing
- 1 Different API's For Data Access to Levels of Data (0-4)
- Processing Engine Convergence and Data/Processing Decoupling will reduce complexity, costs and hardware footprint