# Coming Online: Enabling Real-Time and AI-Ready Scientific Discovery

Adam Thompson | Product Lead – Computational Instruments | **NVIDIA**

Luigi Cruz | Staff Engineer | 👽 **SETI Institute**

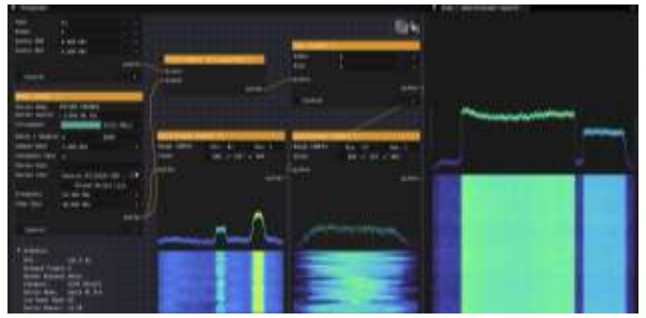
# GPU Accelerated Sensor Processing

Sensor Processing Solutions for the Scientific Python Community

**cuSignal Released as Part of NVIDIA RAPIDS**

2019

**Luigi Cruz Releases CyberEther for Multi-Platform GPU-Accelerated SDR Applications**



2021

**NVIDIA Holoscan Released as Domain and Sensor Agnostic Platform**

2023

2020


2022


2024

**cuSignal Transitions to CuPy v13 thanks to a CZI Essential OSS for Science Grant and Quansight**

# Challenges in Real Time Sensor Processing

**Front end signal processing is typically separated from data processing via storage**

- Many architectures collect snapshots and store to disk, with developers and operators using offline data for analysis, slowing time to scientific insights
- Existing frameworks for online processing are hardware defined and challenging to scale

**Demanding data-rates and real time requirements**

- Next-generation instruments are delivering O(Tbps) scale, particularly as the size of the instrument increases

**Compute intensive algorithms**

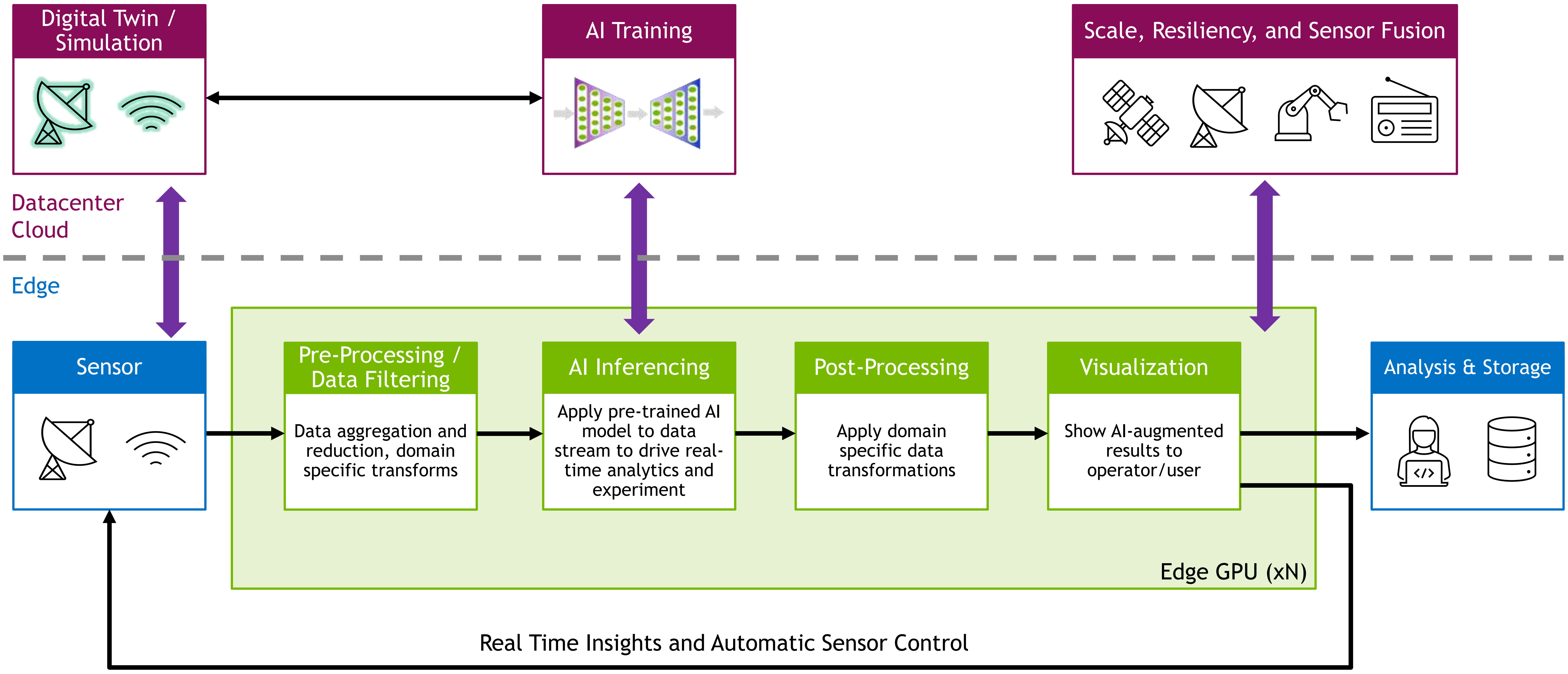
- Correlation (aggregating data from multiple antenna feeds) requires O(N^2) compute

**Massive volumes of processed data, nascent AI research**

- AI techniques for anomaly detection, sensor command and control, signal identification, etc reduce data storage costs and provide developers with faster and better data insights

# Anatomy of a Sensor Processing Workflow

## A Vision of Towards Self Driving Instruments and a Science Programmable Edge



# A Sensor Processing Engineer's Wish List

Andrew Siemion, Bernard M Oliver Chair – SETI Institute

An extensible framework allowing near-theoretical peak-UDP network ingest of sensor data to heterogeneous compute infrastructure that is:

- **Well documented**
- **Extensible**
- **Compatible with upgradable commodity hardware**
- **AI-ready**
- **Scalable to Tbps data rates with 1,000 – 100,000 sensors**
- **Multicast capable**

***Holoscan Developer Day, GTC 2024, Starting ~39-minute Mark***
*https://register.nvidia.com/flow/nvidia/gtcs24/attendeeportal/page/sessioncatalog/session/1700248989187001E4DN*

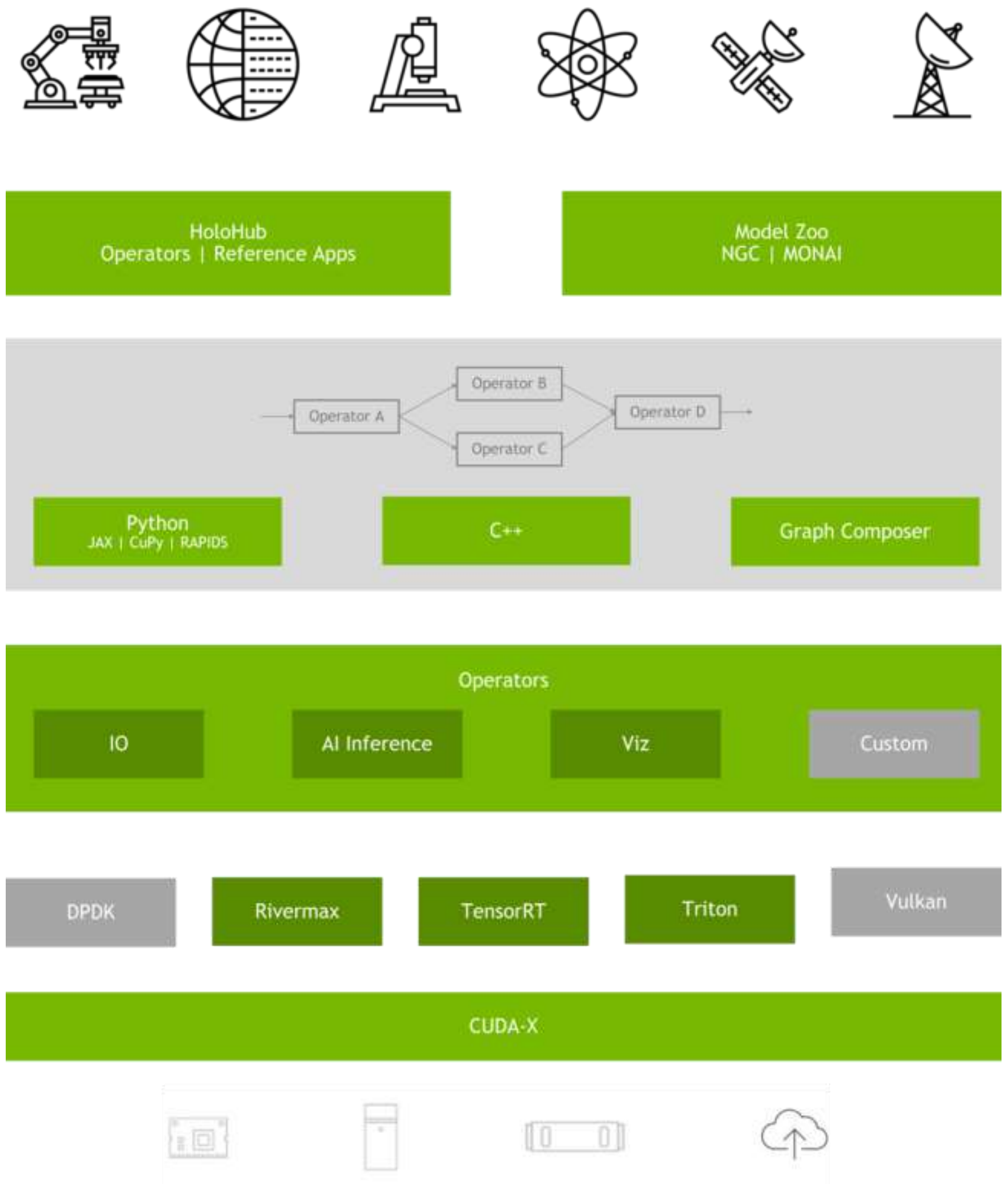
# Holoscan

# AI-Enabled, Real-Time Sensor Processing Platform

# NVIDIA Holoscan

SDK for Building AI-Enabled Sensor Processing Applications

GitHub



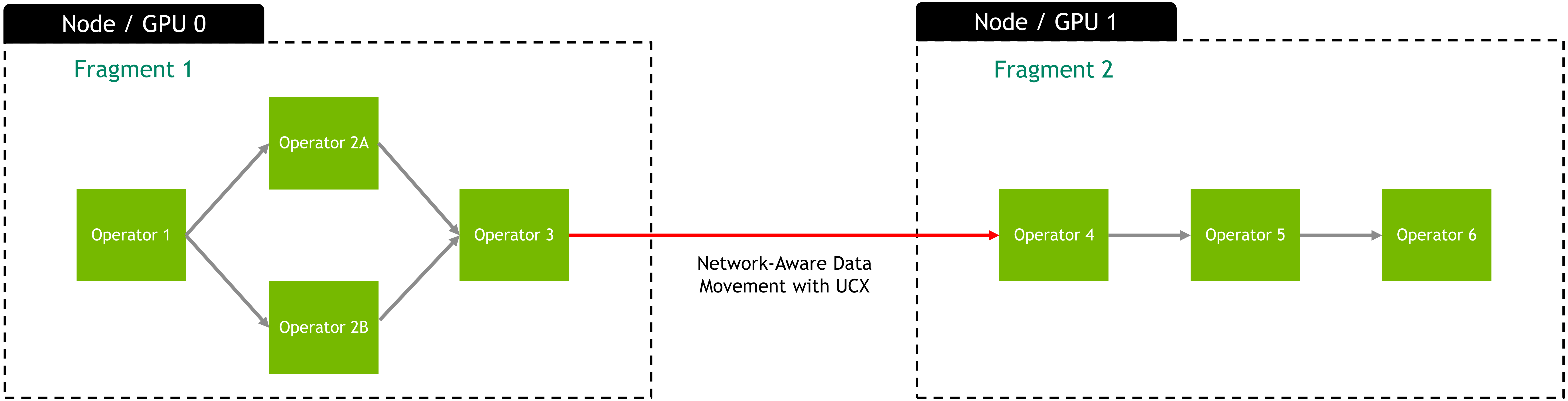

## Features

- C++ and Python APIs for domain agnostic sensor data processing workflows
- Multi-Node and Multi-GPU support with advanced pipeline scheduling options and network-aware data movement
- AI Inference with pluggable backends such as ONNX, Torchscript, and TensorRT
- Scalable from Jetson Orin Nano (ARM + GPU) to DGX (x86 + H100)
- Apache 2 Licensed and Available on GitHub

## Benefits

- Simplifies sensor I/O to GPU
- Simplifies the performant deployment of an AI model in a streaming pipeline
- Provides customizable, reusable, and flexible components to build and deploy GPU-accelerated algorithms
- Scale workloads with Holoscan's distributed computing features
- Deploy to the Cloud with Holoscan App Packager and Kubernetes

# Holoscan Fundamentals




Holoscan Applications are built by forming a graph of either core or custom Holoscan Operators. Operators are the fundamental unit of work in Holoscan and can define I/O, AI inferencing, visualization, and accelerated computing functions

Holoscan Fragments define hardware locality of a given series of connected Operators. Data movement within a Fragment is facilitated via shared GPU pointers

## Schedulers

**Greedy Scheduler**

Uses single CPU thread to launch operators in a pipeline sequentially

**Multi-Threaded Scheduler**

Can pin operators to a specific CPU thread for async execution and pipeline parallelism

**Event Based Scheduler**

Multi-Threaded scheduler that waits for events rather than polling for status

## Profiling Tools

**Data Flow Tracking**

Tracks data latency as a packet/frame moves through a Holoscan application

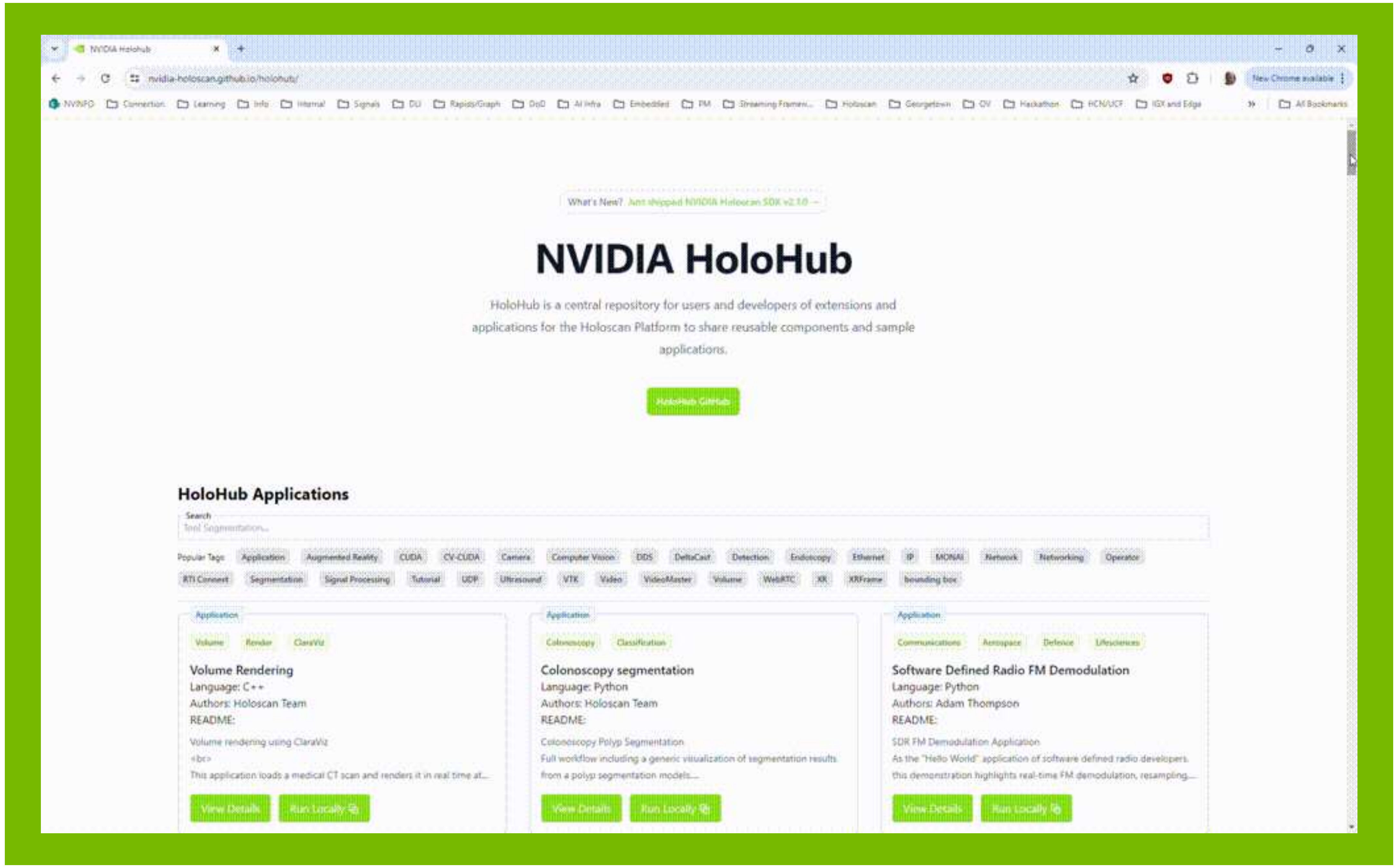
**Nsight Systems**

Observe overall application behavior and performance

# Holohub

Sample Holoscan Applications, Tutorials, and Helpful Operators

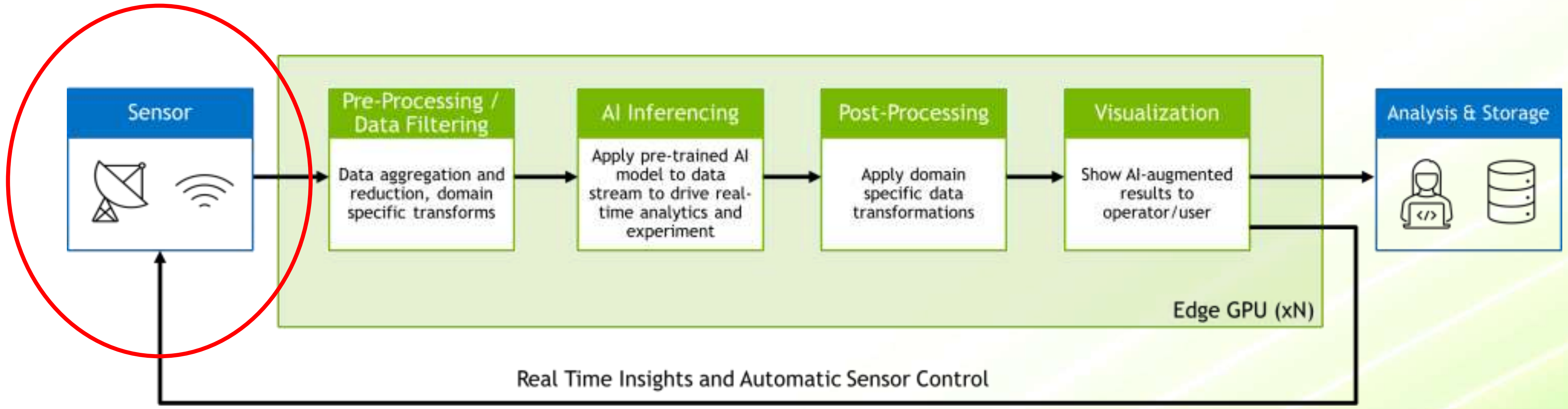
GitHub




**IO:** UDP Ethernet, Lidar, High Speed Cameras, SDR, SAR, 3D Volumes

**AI:** Segmentation, Tracking, AR/VR, LLMs

**Tools/Tutorial:** MATLAB, MONAI, Playground on AWS, Scaling Apps




# Holoscan Networking Operators

# Holoscan Network Operators

Simplification of Moving Data to and From GPU

## Basic Network Operator

Focus on **Simplicity**: TCP/IP to and from GPU at < 10Gbps

Uses Linux Sockets to provide a common interface to send and receive data

Works on all Linux distributions and network cards

Kernel provides protocol stacks, freeing the user from worrying about retransmits, headers, etc

Cannot achieve line rate on modern NICs

**Learn more about the BNO on Holohub**


## Advanced Network Operator

Focus on **Performance**: Any Ethernet Packet (UDP, VITA-49, etc) to and from GPU at Line Rate

Bypasses Linux kernel for access directly to NIC DMA buffers

Requires a Mellanox/NVIDIA Network Interface Card (NIC)

Zero-copy interface from NIC into user buffers or directly to GPU using GPUDirect via standard UDP packets (no RDMA protocol needed!)

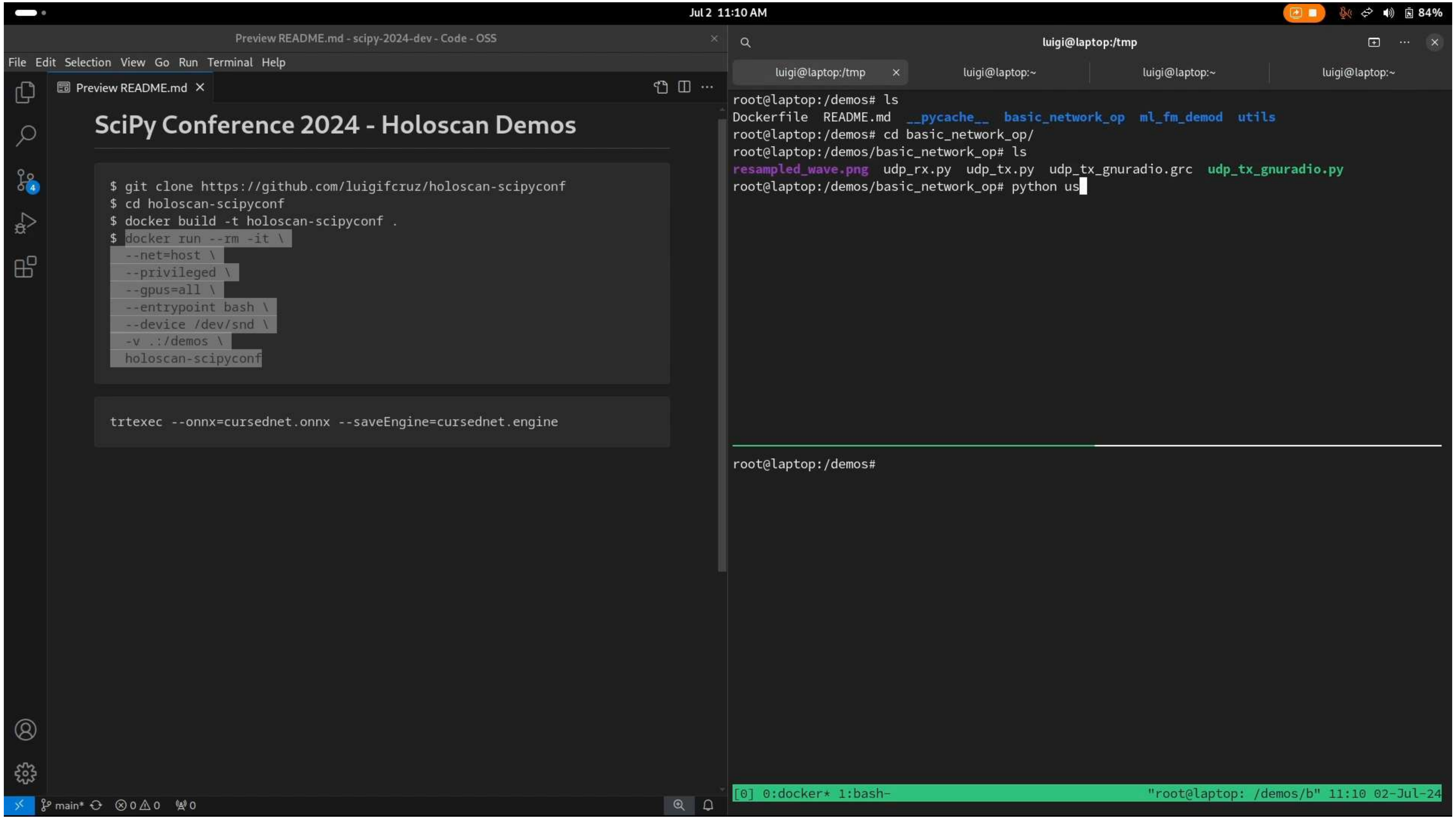
Python bindings are in progress

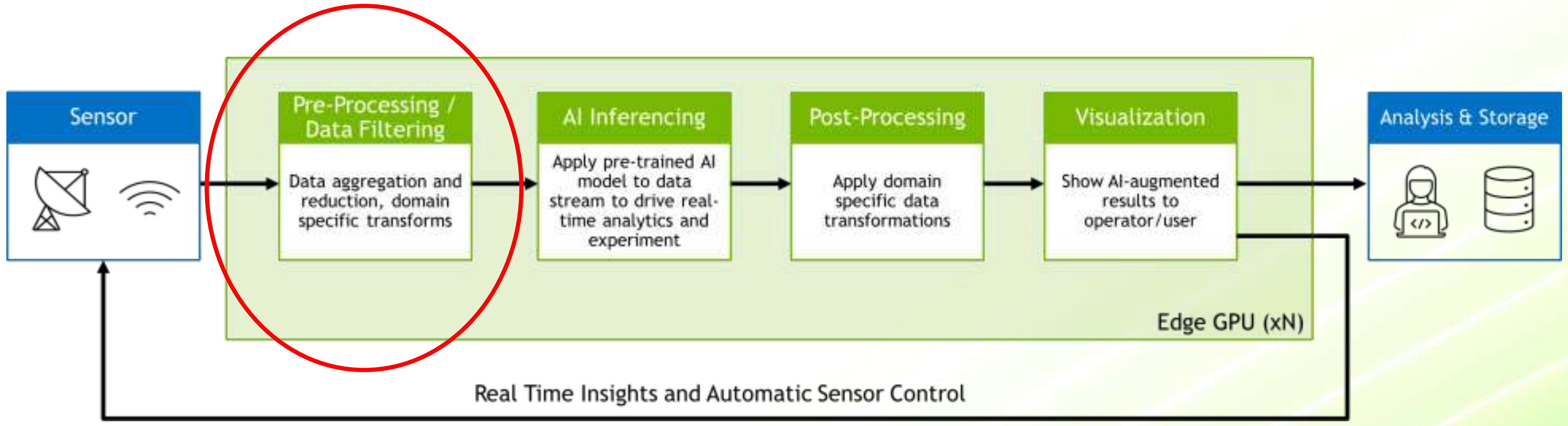
**Learn more about the ANO on Holohub**

# Basic Network Operator

## Receiving Data from External Ethernet Source

GitHub






# Holoscan Compute Operators

# Tap Into A Sensor Stream with GPU Accelerated Python Packages

Connect Domain Specific Algorithms with Real Time Sensors

**Holoscan Operator (Python)**


Custom Operator Class

```
class ResampleOp(Operator):
    def __init__(self, *args, **kwargs):
        up = 2
        down = 3
        # Call base constructor class
        super().__init__(*args, **kwargs)
```

Define Input / Output Ports

```
    def setup(self, spec: OperatorSpec):
        spec.input("in_sig")
        spec.output("out_sig")
```

```
    def compute(self, op_input, op_output, context):
        sig = op_input.receive("in_sig")
        resample_sig = cusignal.resample_poly(sig, up, down)
        op_output.emit(resample_sig, "out_sig")
```

Receive Message from Input Port

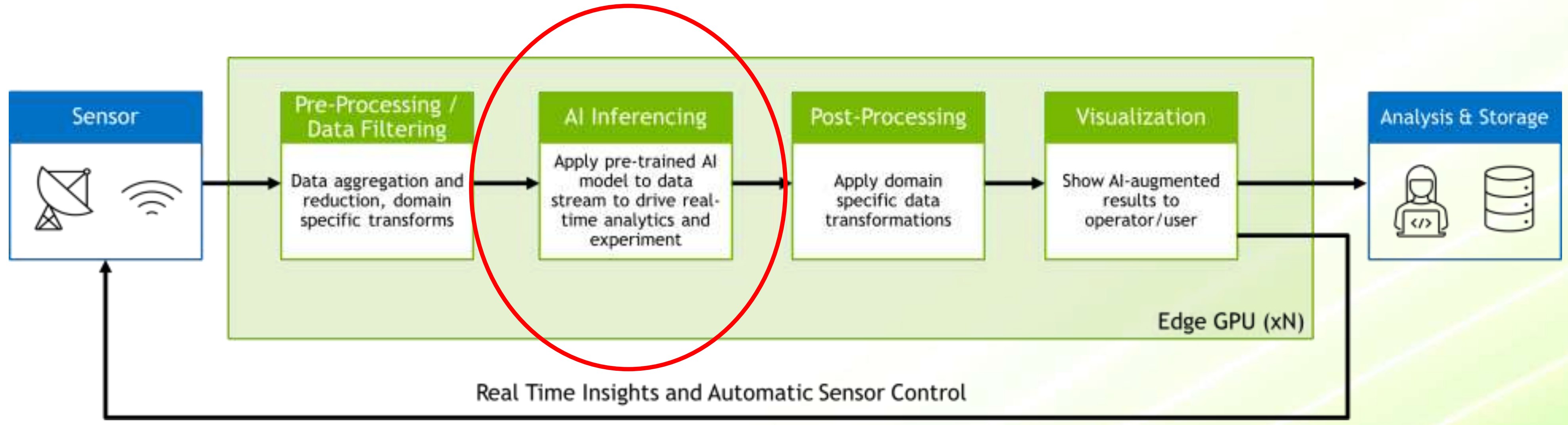
Emit Message from Output Port

**New in v2.2: Decorators for Operator Creation**

```
from holoscan.decorator import create_op

@create_op(inputs='in_sig', outputs='out_sig')
def resample_poly(in_sig, up, down):
    out_sig = cusignal.resample_poly(in_sig, up, down)
    return out_sig
```

NVIDIA




# Holoscan AI Inferencing Operator

# AI-Based FM Demodulation

## Using Holoscan to Transform a Classical Sensor Processing Pipeline with AI

### Classical Holoscan Software Pipeline

Receive I/Q Stream from SoapySDR → (I/Q Samples) → Buffer Radio I/Q Samples in GPU Memory → (Pointer) → Convert Buffer to CuPy Array → (Pointer) → FM Demod (cuSignal) → (Pointer) → Polyphase Resample (cuSignal) → (Move to CPU) → Write to PyAudio Queue (Python)

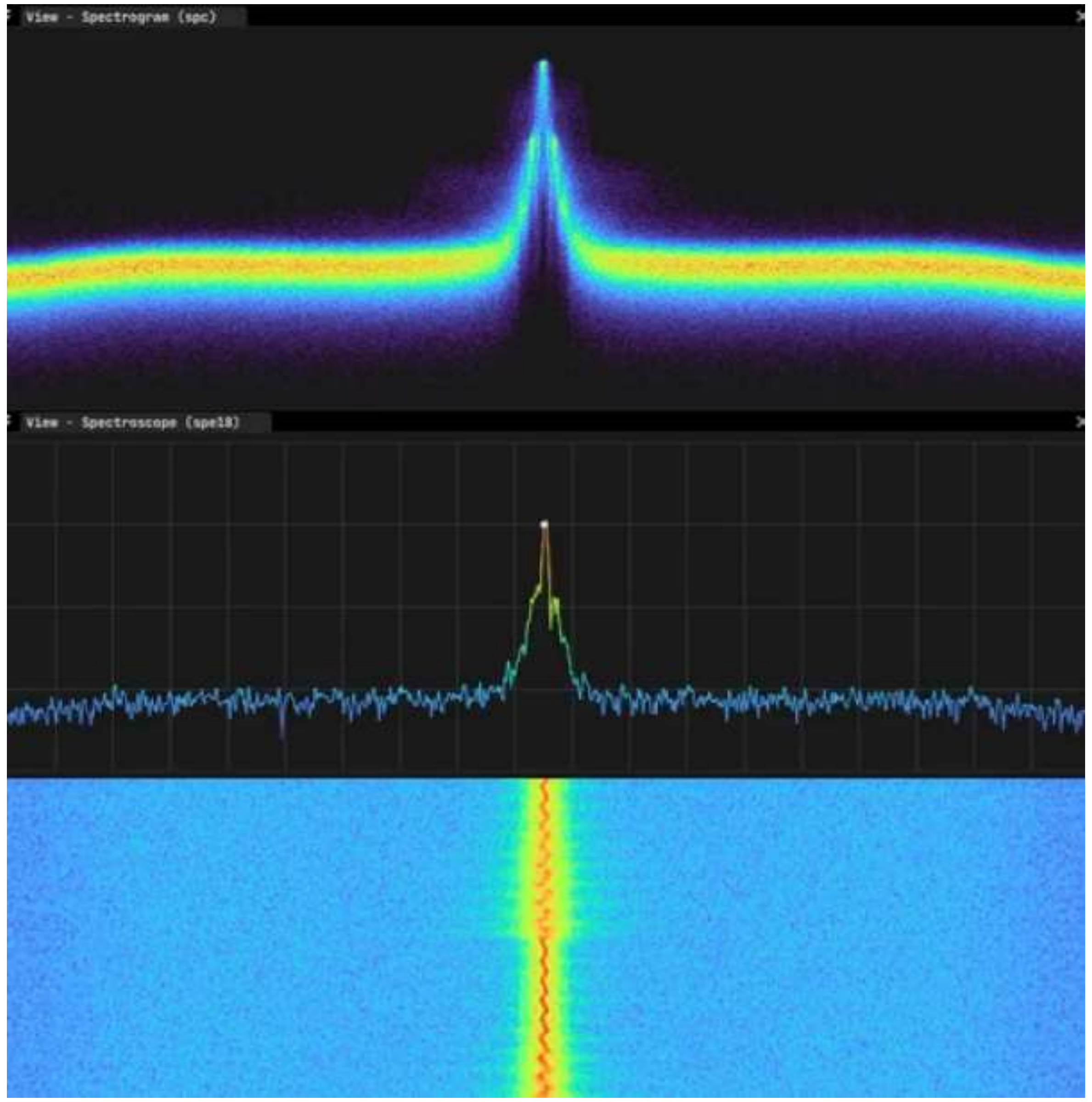
Reduce Computation (no more resampling!)
Data Movement is Unchanged (same overall pipeline)

### AI Enabled Holoscan Software Pipeline

Receive I/Q Stream from SoapySDR → (I/Q Samples) → Buffer Radio I/Q Samples in GPU Memory → (Pointer) → AI Inference Operator → (Move to CPU) → Write to PyAudio Queue (Python)

# AI Inferencing Operator

## Behind the Scenes of FM Demodulation with a Trained Neural Network



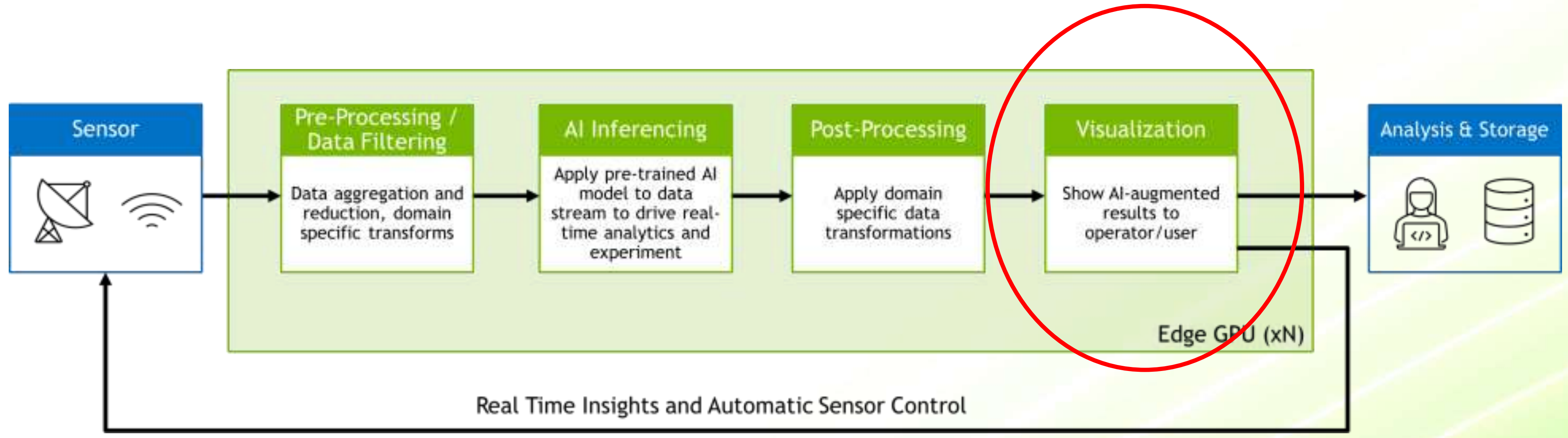

```python
class NeuralFmDemod(Application):
    def __init__(self):
        super().__init__()

    def compose(self):
        pool = UnboundedAllocator(self, name="allocator")
        src = SignalGeneratorOp(self, name="src")
        preprocessor = PreProcessorOp(self, name="preprocessor")
        inference = InferenceOp(self,
            allocator=pool,
            backend="trt",
            input_on_cuda=True,
            output_on_cuda=True,
            transmit_on_cuda=True,
            is_engine_path=True,
            pre_processor_map={
                "demod": ["rx_sig"],
            },
            model_path_map={
                "demod": "./cursednet.engine",
            },
            inference_map={
                "demod": ["rx_sig"],
            },
        )
        postprocessor = PostProcessorOp(self, name="postprocessor")
        sink = SDRSinkOp(self, name="sink")

        self.add_flow(src, preprocessor)
        self.add_flow(preprocessor, inference, {("tensor", "receivers")})
        self.add_flow(inference, postprocessor)
        self.add_flow(postprocessor, sink)
```

GitHub

NVIDIA

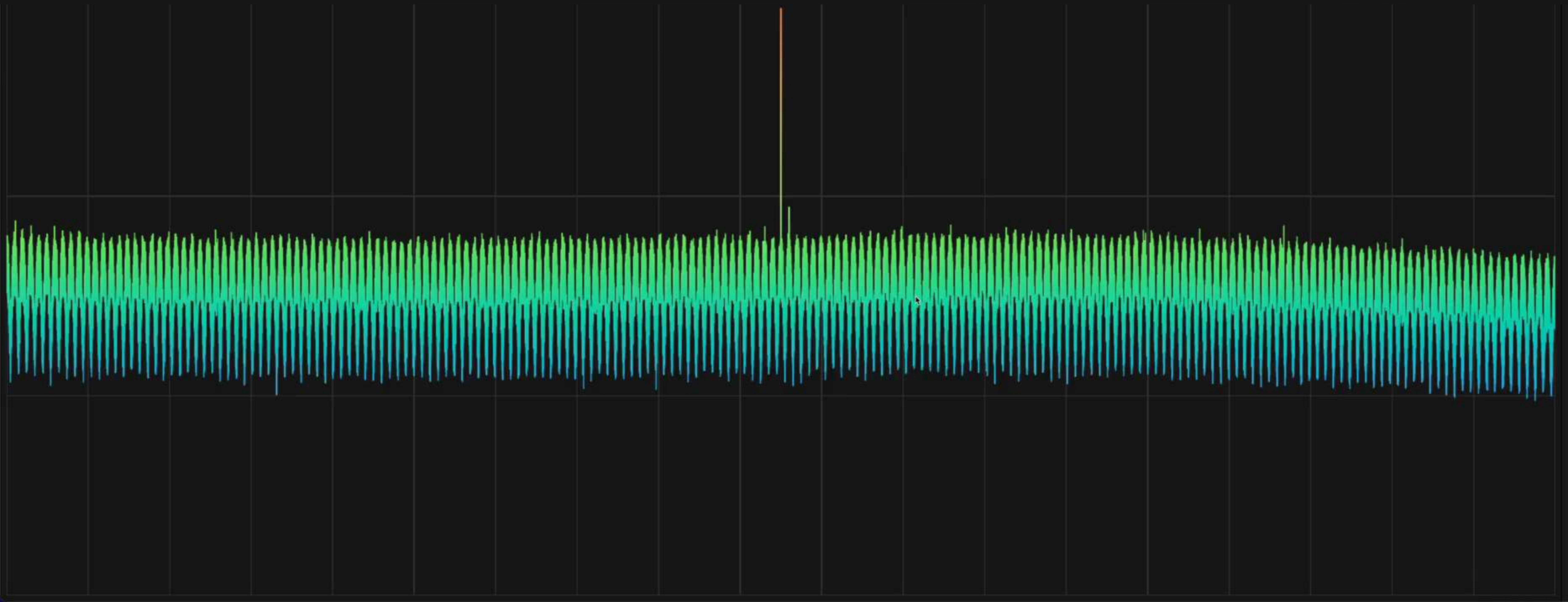



# Holoscan Visualization with Community Tools

# Deep Space Demonstration

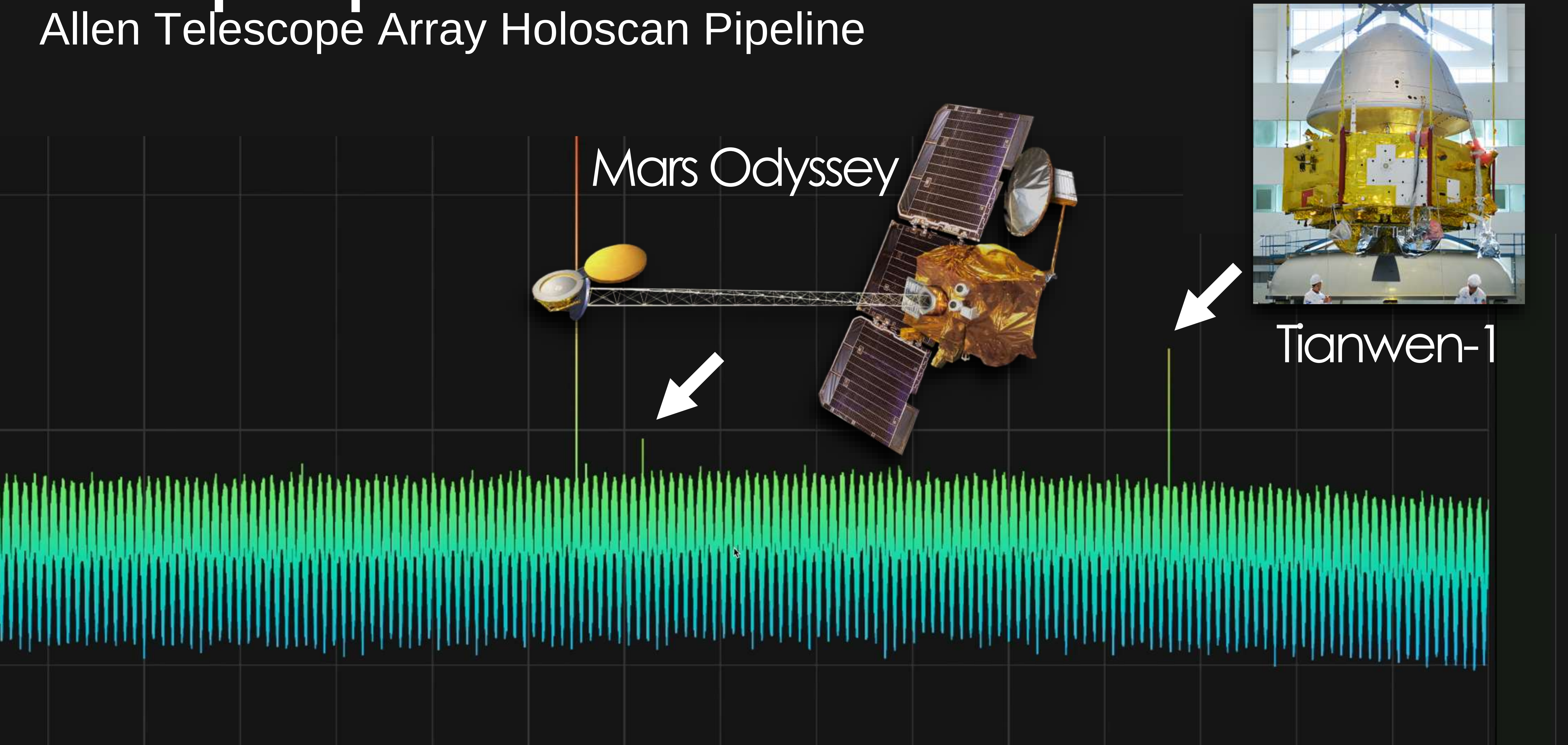
## Allen Telescope Array Holoscan Pipeline with CyberEther

CyberEther
GitHub



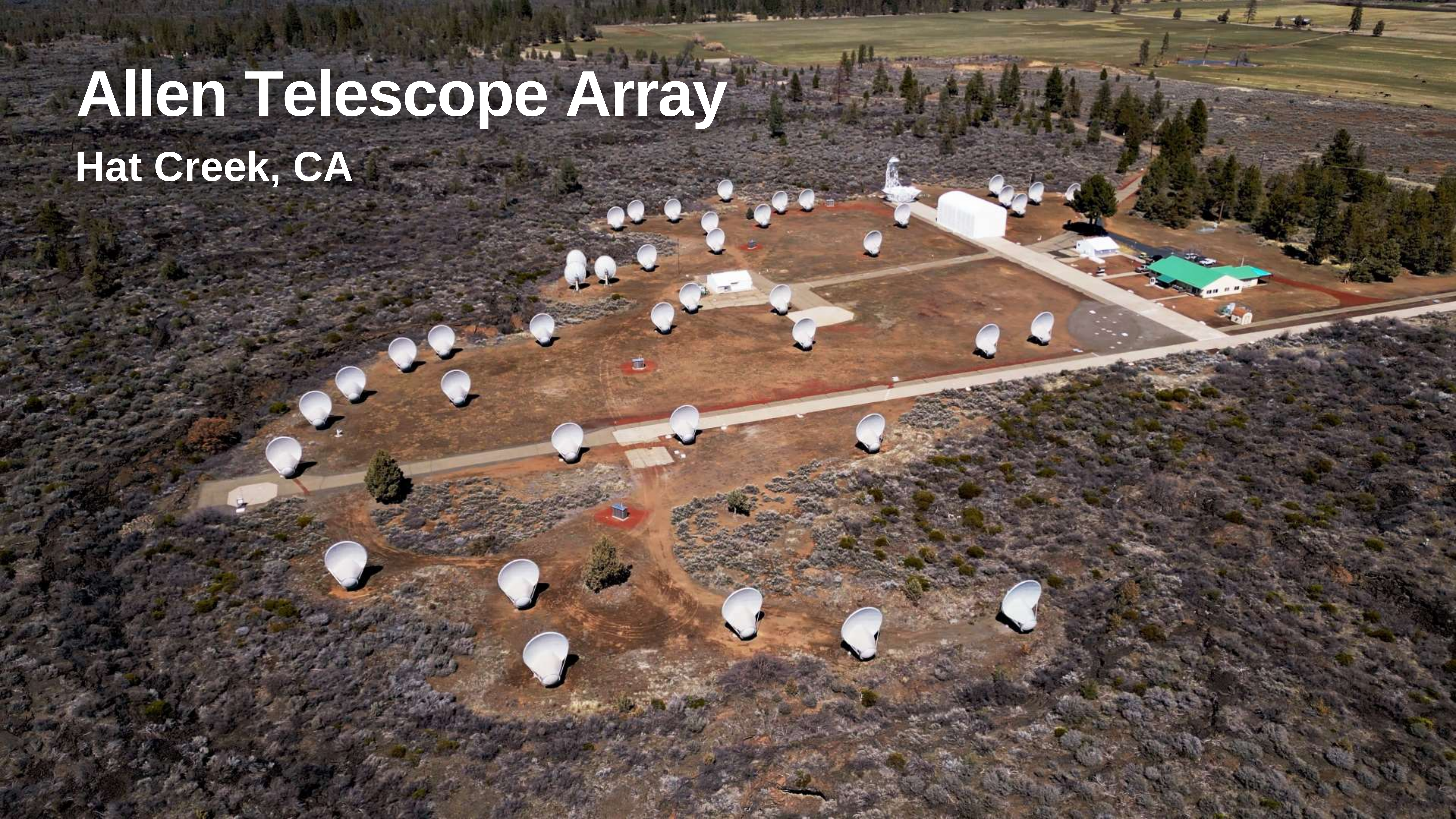
# Deep Space Demonstration

## Allen Telescope Array Holoscan Pipeline



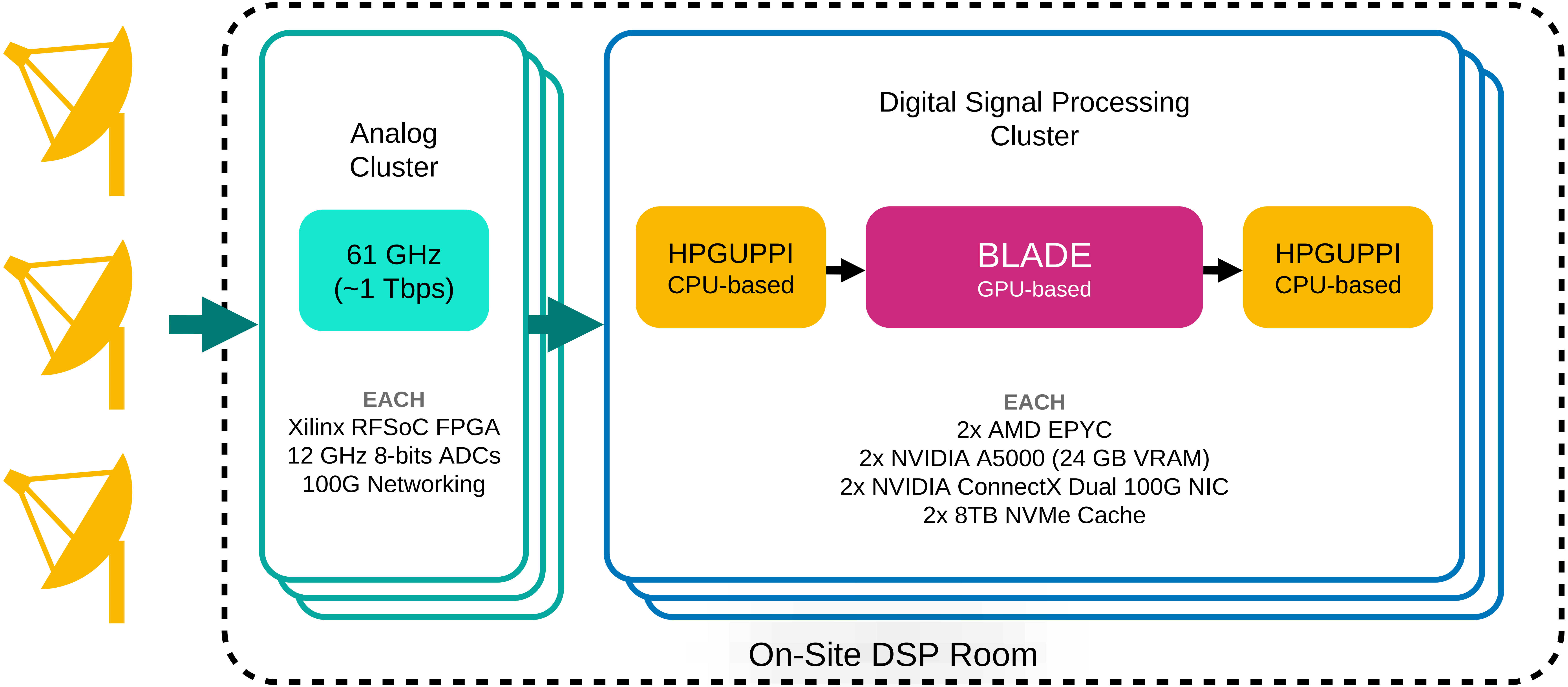
# Putting it all Together:

# Holoscan at the Allen Telescope Array


Allen Telescope Array
Hat Creek, CA

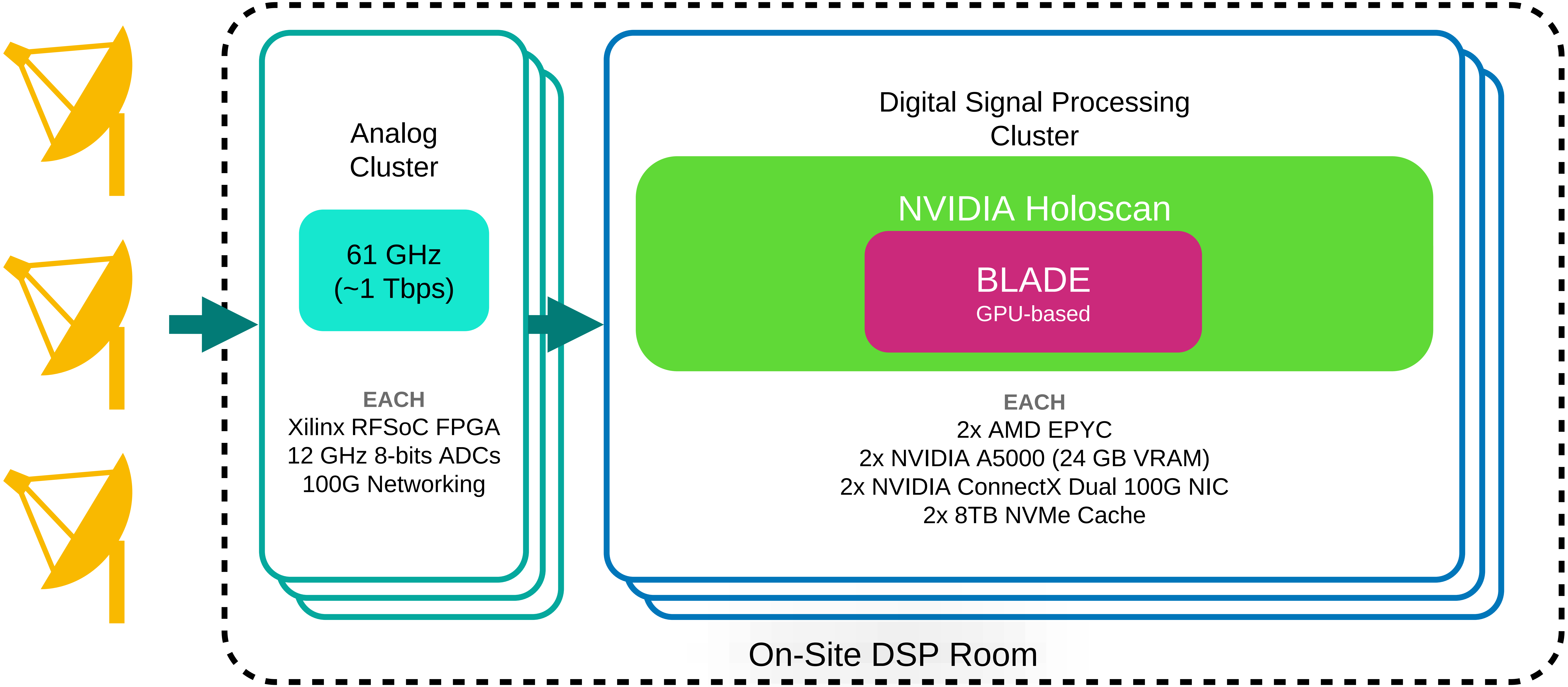
# Data Processing

## Current Pipeline



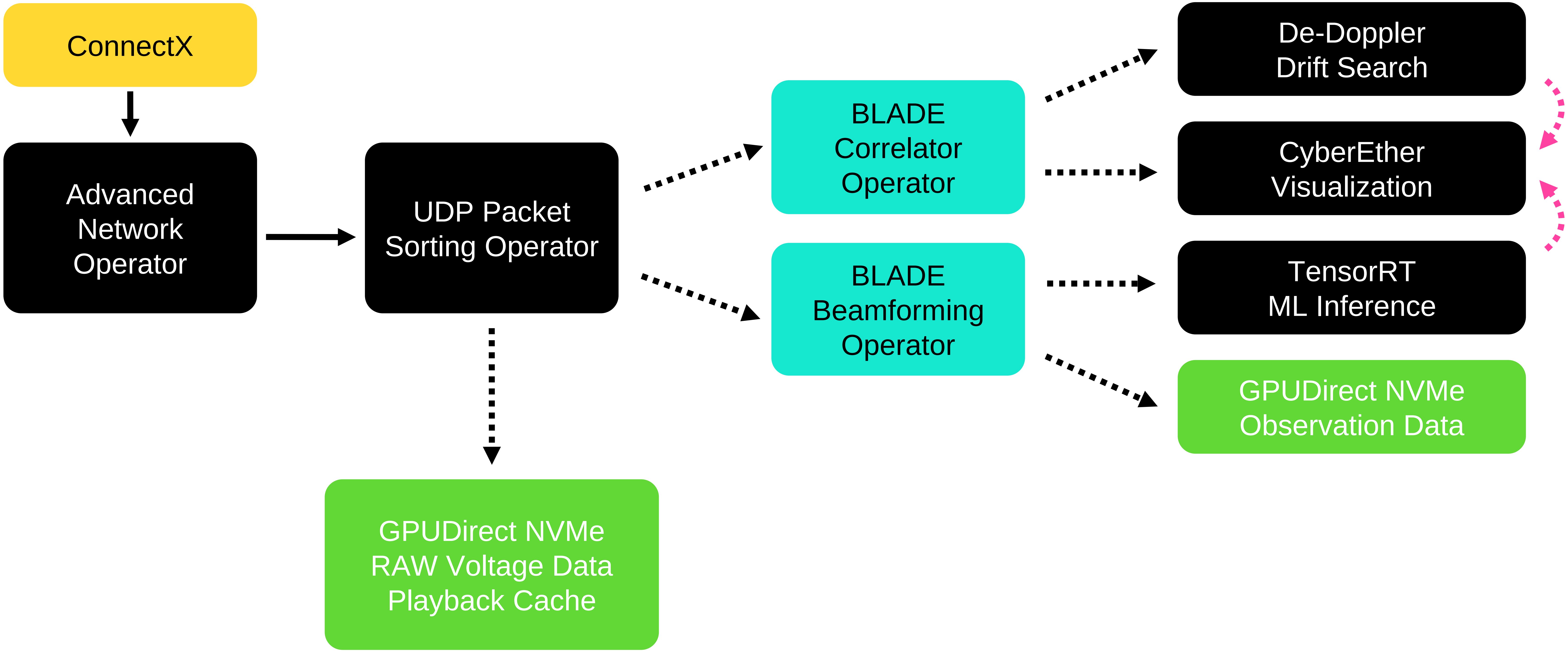
# Data Processing

## Future Pipeline



# Future

## Extending the ATA's Capabilities with Holoscan



# Getting Started

# Getting Started with Holoscan

**Holoscan References**

https://github.com/nvidia-holoscan/holoscan-sdk

```
docker pull nvcr.io/nvidia/clara-holoscan/holoscan:v2.2.0-dgpu
```

```
pip install holoscan
```

Debian Packages available on NGC

https://docs.nvidia.com/clara-holoscan/sdk-user-guide/index.html

# Holoscan Bootcamp and Open Hackathons

Half Day Free Tutorial – September 24th, 2024 – 1pm – 5pm Eastern

OPEN HACKATHONS

Events Attendees Mentors About Resources

Search...

Log In

## NVIDIA HOLOSCAN BOOTCAMP

September 24-24, 2024
Application Deadline: September 4, 2024
Virtual Event

## Event Overview

From radio telescopes to particle accelerators, scientific instruments produce tremendous amounts of data at equally high rates. To handle this data deluge and to ensure the fidelity of the instruments' observations, architects have historically written measurements to disk, enabling downstream scientists and researchers to build applications with pre-recorded files. The future of scientific computing is AI-centric, interactive, and streaming; how many Nobel Prizes are hidden on a dusty hard drive that a scientist didn't have time or resources to analyze? NVIDIA® Holoscan is the solution.

NVIDIA Holoscan is a domain-agnostic AI computing platform that delivers the accelerated, full-stack infrastructure required for scalable, real-time processing of streaming data.

Together with the OpenACC organization, NVIDIA will host a virtual Holoscan Bootcamp on September 24, 2024. This Bootcamp focuses on building an end-to-end AI-enabled streaming pipeline using the Holoscan SDK, handling sensor I/O, applying a trained AI model to a real time sensor stream, and building GPU accelerated applications. We will also discuss techniques to measure application performance and transition from prototype to production.

This online Bootcamp is a hands-on learning experience where attendees will be guided through step-by-step instructions with teaching assistants on hand to help throughout.

APPLY NOW CONTACT US

Share:

Register Here

# SETI Institute

Join the Search for Intelligent Life!

The SETI Institute is a non-profit research organization, located in the Silicon Valley close to the NASA Ames Research Center. Our mission is to lead humanity's quest to understand the origins and prevalence of life and intelligence in the universe and share that knowledge with the world.

- Check seti.org/jobs for open positions (researchers, data-analysis, etc).
- Apply to became an affiliate or investigator at seti.org/become-pi-or-affiliate.

NVIDIA