Hewlett Packard Enterprise

Combining scientific simulations and AI with SmartSim

Andrew Shao, PhD
Principal HPC&AI Research Scientist | HPE Canada

10 September 2024
Joint OLCF/NERSC Workshop

Outline of this Workshop

Introduction: Why combine AI and scientific simulation?

ML-in-the-loop: Applications in Climate Modeling and CFD

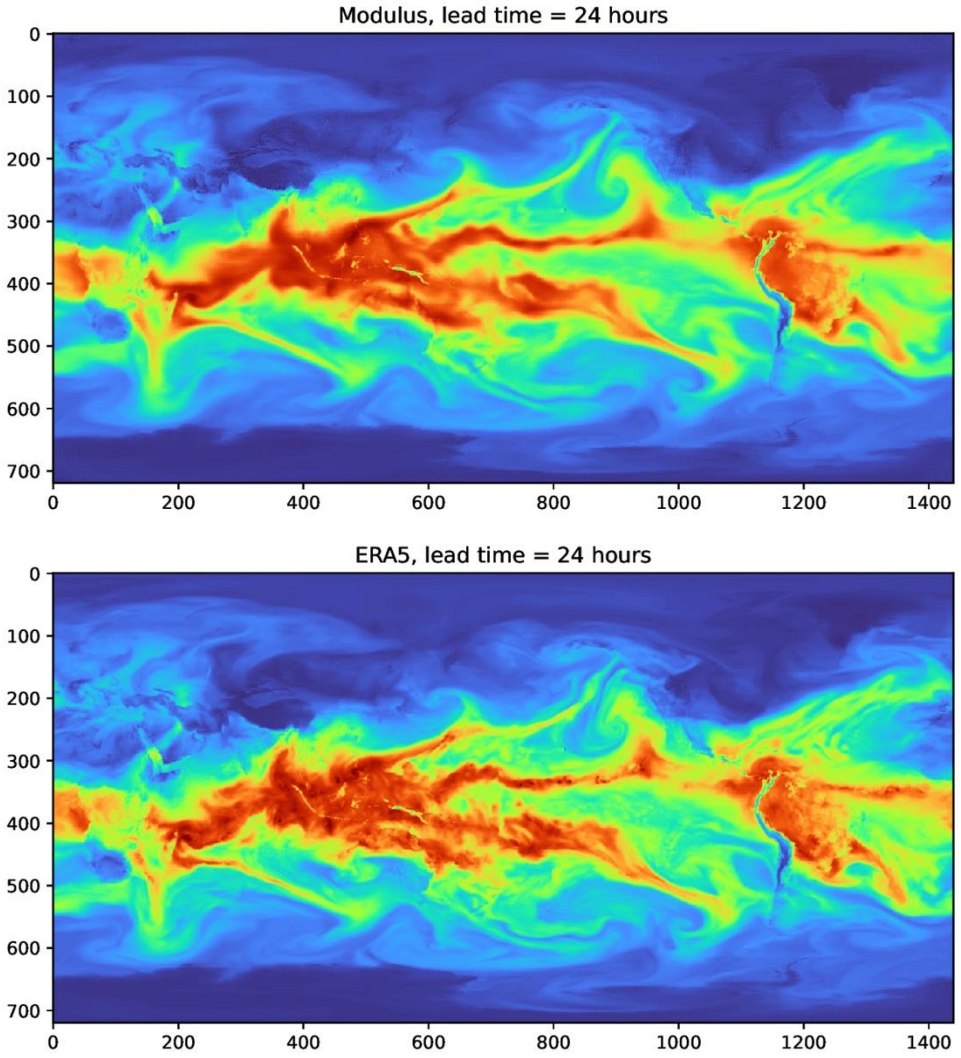
ML-around-the-loop: Applications in Molecular Dynamics and CFD

Hands on: Building simple SmartSim applications

Hands on: Building a complex workflow to train a neural network online

Why HPC and AI instead of HPC VS. AI?

- **Can AI replace numerical-based approaches?**
  - Short answer: no, still limited by data
- **Benefits of AI models**
  - Can be run more quickly than traditional numerical models
  - Simpler to run, does not need complicated software infrastructure and HPC resources
- **Downsides of AI models**
  - How do you add add process complexity?
  - Extrapolation beyond training dataset?
- **Challenges to combining HPC&AI**
  - Numerical: How can you characterize the stability and accuracy of an ML model in that context
  - Technical:
    - How do you connect Fortran/C/C++ codebases to ML packages?
    - How do you appropriately balance high-value/cost GPU resources in predominantly CPU-based code?



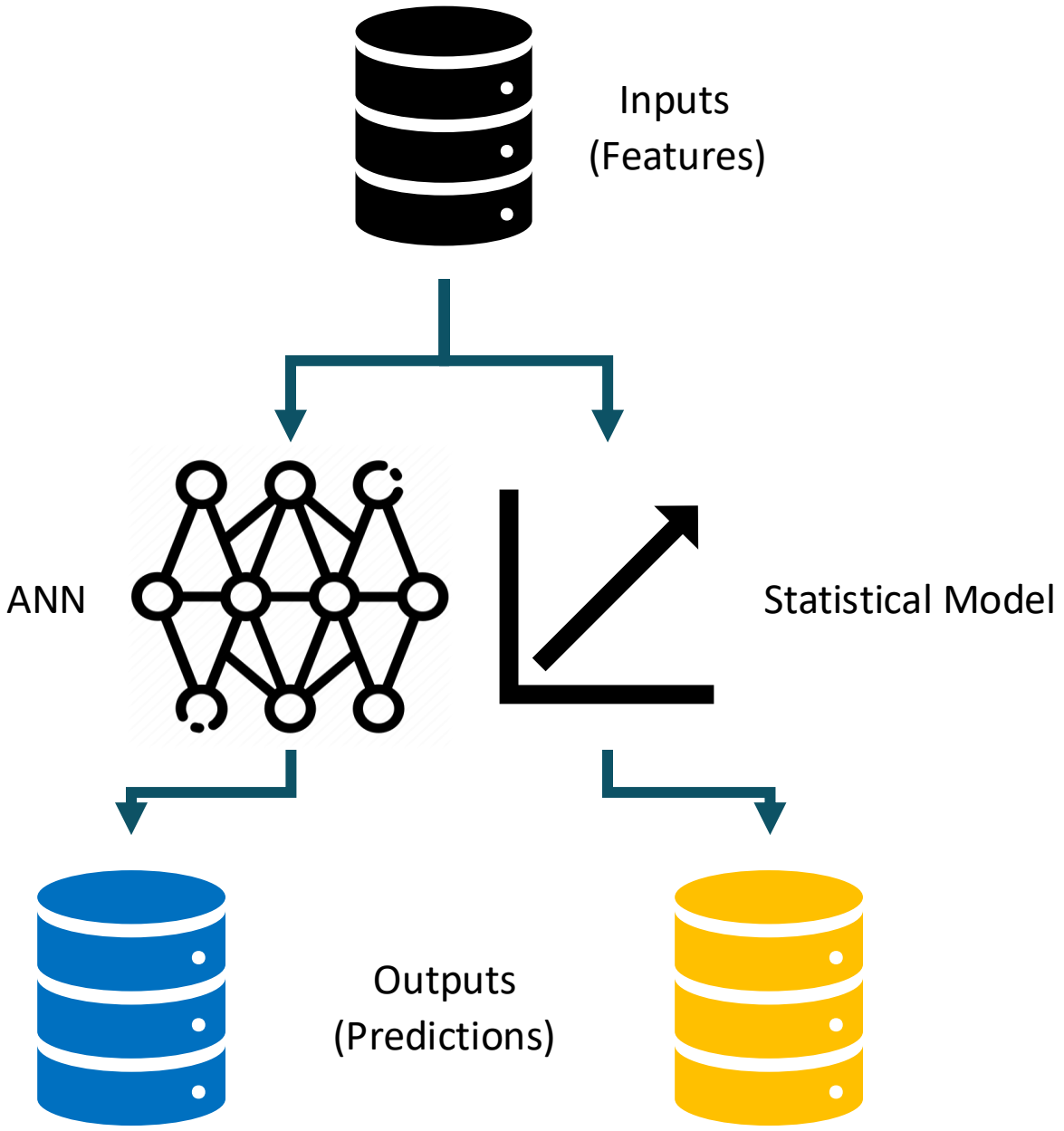
https://docs.nvidia.com/deeplearning/modulus/modulus-sym/user_guide/neural_operators/fourcastnet.html#introduction

Archetypes of machine learning

- **Supervised learning:**
  - Inputs and known outputs, derive a relationship
  - Linear regression falls under supervised learning
  - **Artificial neural network:**
    - Linear and non-linear transformations
    - Challenge: Many free parameters requires data to over-constrain the problem
- **Reinforcement learning**
  - Train model to take actions within a given ruleset based on predefined reward function

- Unsupervised Learning (not discussed today)
  - Find relationships in unlabeled data

All archetypes rely on data to learn and generalize



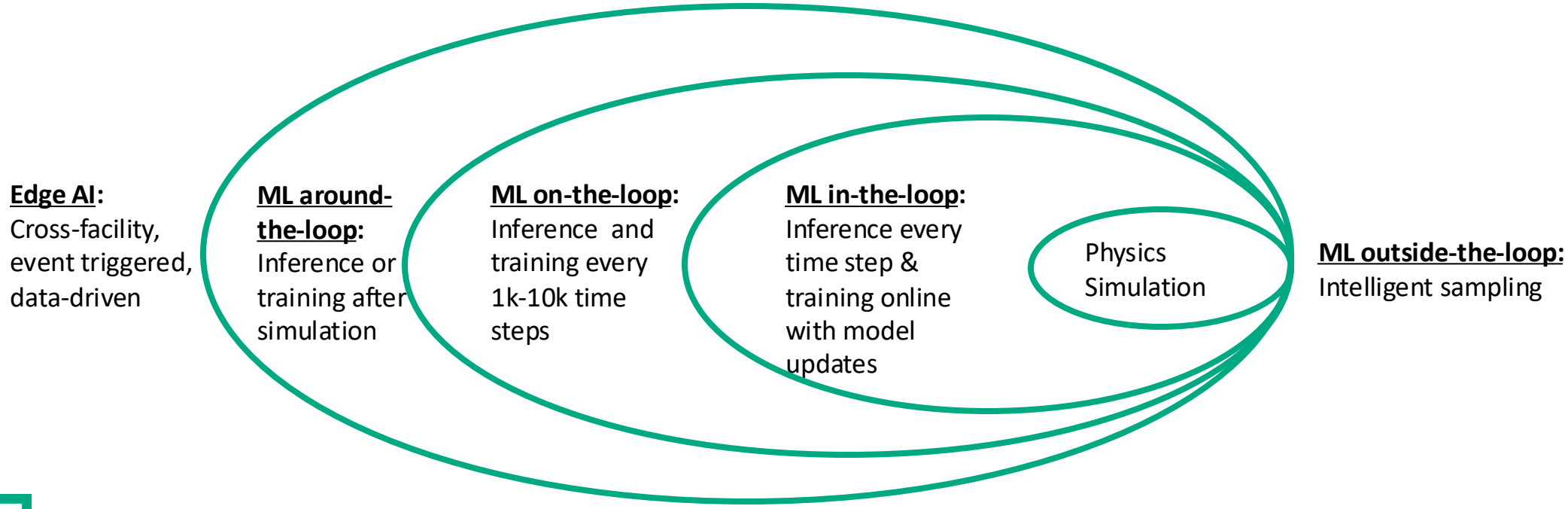
Combining AI/ML with Scientific Simulation

ML in-the-loop

- Embedding machine-learning predictions within numerical solvers
- On-the-fly analysis and visualization (e.g. principal component analysis via streaming SVD)

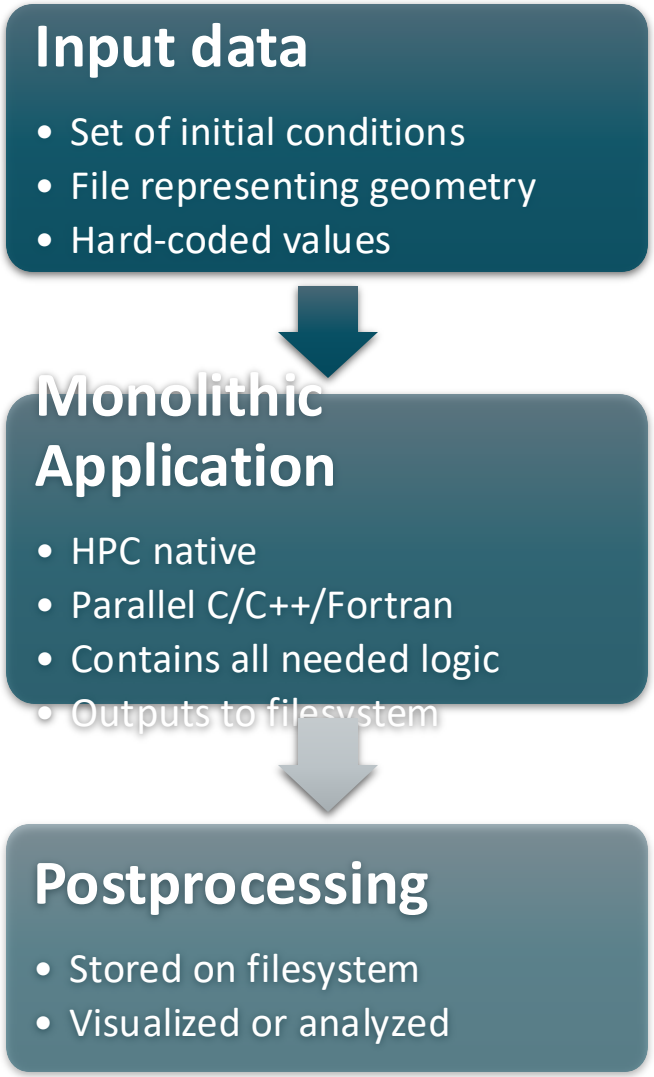
ML around-the-loop

- Automatic parameter tuning
- Reinforcement learning using the simulation as a testing environment



The standard way of running simulations

Typical Numerical Workflows



Characteristics

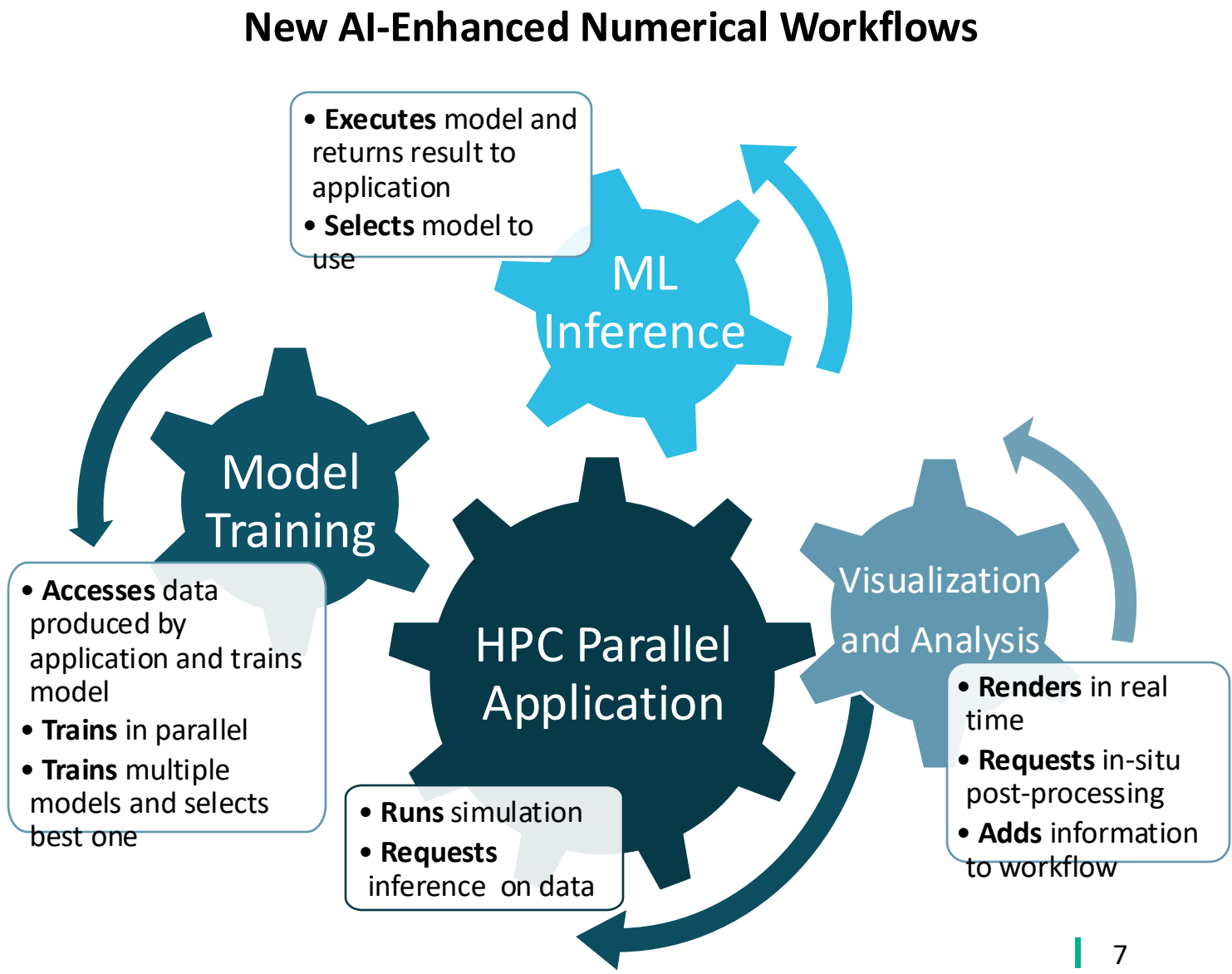
- Workflow has defined, serial dependencies
- Representable with pipeline or directed acyclic graph

Leading question

- Can this rigid structure accommodate the scale and desired applications of numerical simulations?
  - How do we couple inputs/outputs across multiple applications?
  - What happens when the data becomes too big to output?
  - How do you define non-unrollable loops or branches?

The new scientific tool: Workflows

- **Paradigm shift: The simulation is one component of a larger application**
  - Simulation requires inputs from other components during execution
  - Outputs from simulation needed by other components
- **Traditional pipeline?**
  - Requires branching logic, difficult to coordinate
  - Return to file-based signalling
  - Data passed through stages must be stored
- **SmartSim's role**
  - Provide a central location to share data
  - Allow scientists to define components of workflow



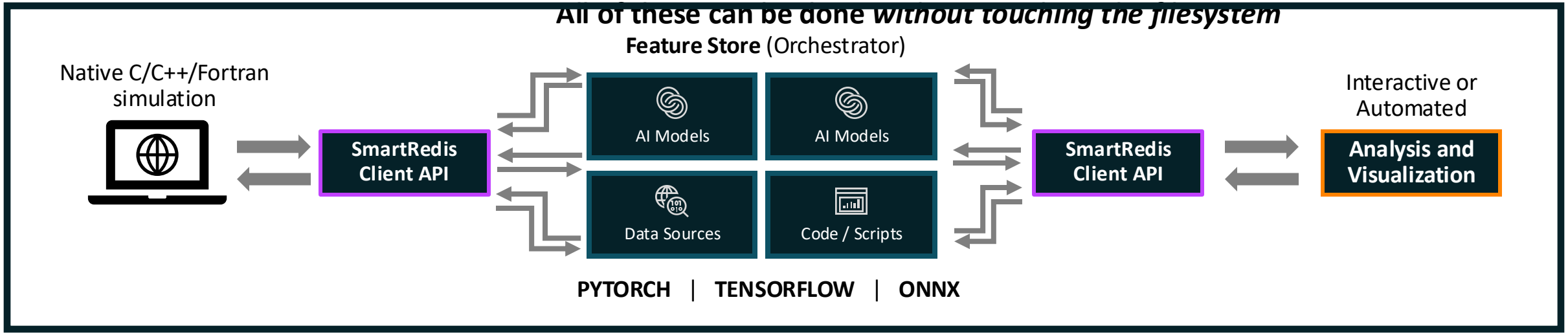
About SMARTSIM

SmartSim is an open-source library

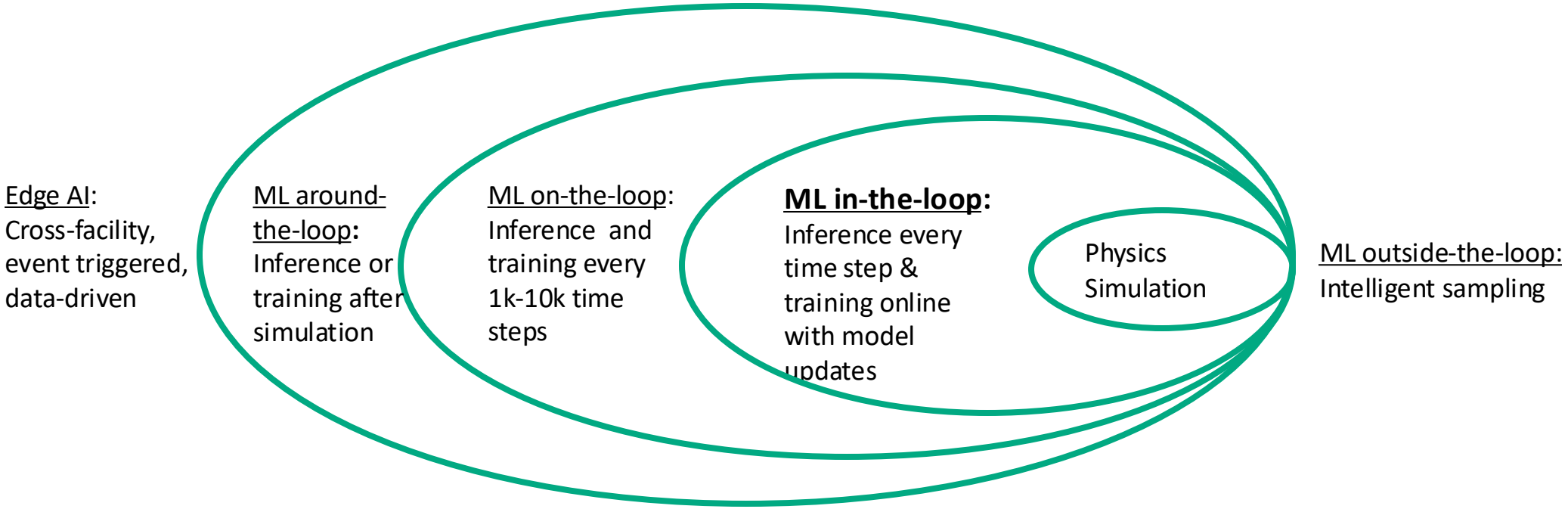
- bridging the divide between traditional numerical simulation and data science
- providing a **loose-coupling** philosophy for combining HPC & AI

SmartSim allows scientists to create complex workflows, with simulations and machine learning producing, exchanging, and consuming data

- **Call** Machine Learning (ML) inference in existing Fortran/C/C++ simulations
- **Exchange data** between C, C++, Fortran, and Python applications
- **Train ML models online** and make predictions using TensorFlow, PyTorch, and ONNX
- **Analyze data** streamed from HPC applications while they are running



ML-in-the-loop: Scientific Applications



Turbulence modelling in ocean models

Problem 1:

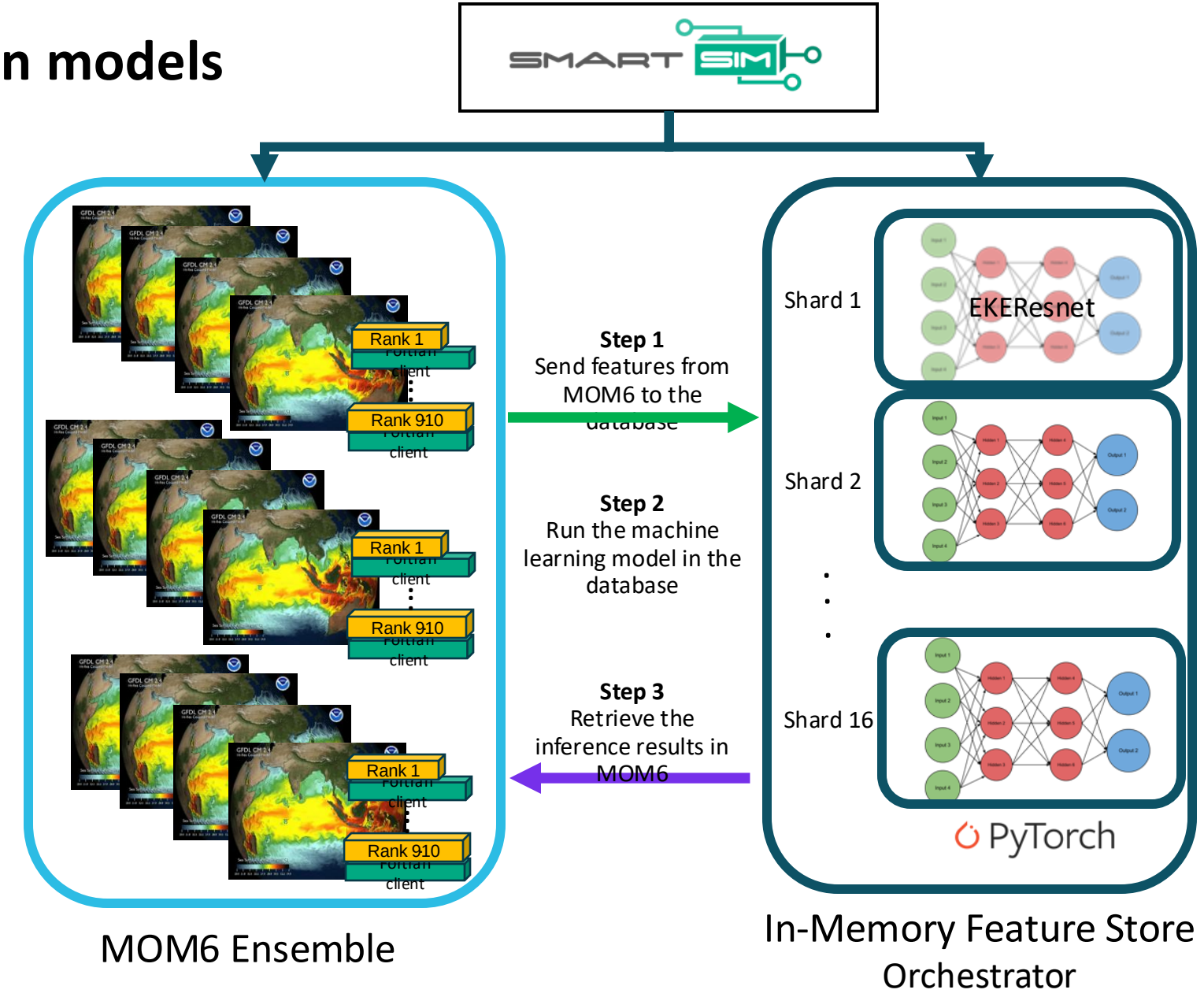
- Eddy kinetic energy computed via a prognostic equation (Jansen et al., 2015)
- EKE equation has terms which are tunable and/or have errors which may be first-order

Problem 2

- Ocean turbulence energizes large-scale flow
- Coarse simulations overly diffusive

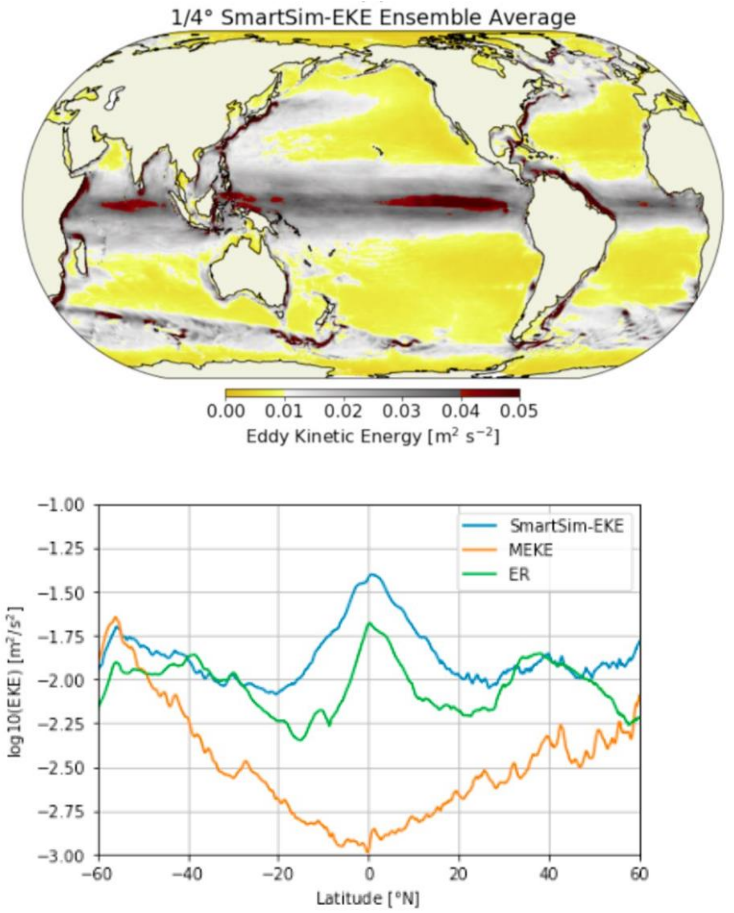
Proposed ML-based solution

- Train an ML model to estimate EKE
- Train an ML model from high-res data to add energy (increase velocities) to the system
- Embed predictions in simulation

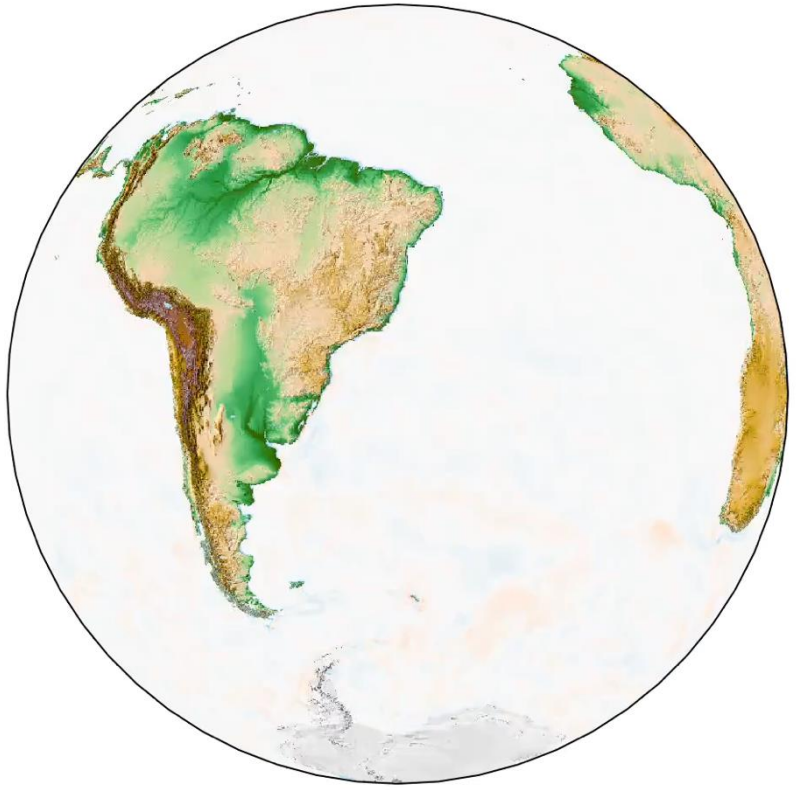


Online AI improves simulation accuracy

- Simulation accuracy improves with AI-based parameterizations
- Scales well with ensembles or large individual simulation
- Parameterizations add 10-20% cost when run on GPU
- Online accuracy of neural network better than offline
- May form the basis of science-oriented MLCommons benchmark [Brewer and von Laszewski]



AI-based Eddy Kinetic Energy
[Partee et al., 2022]

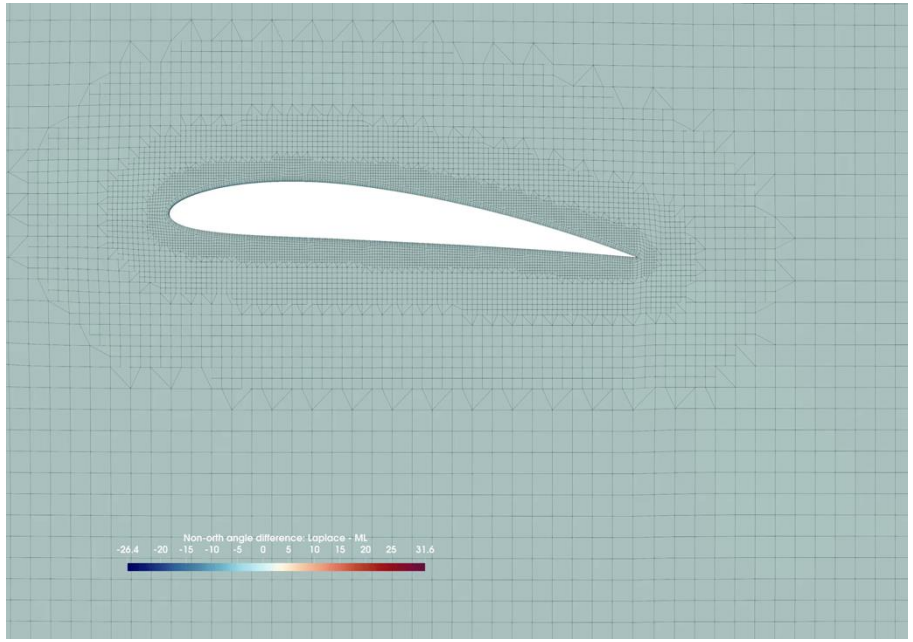


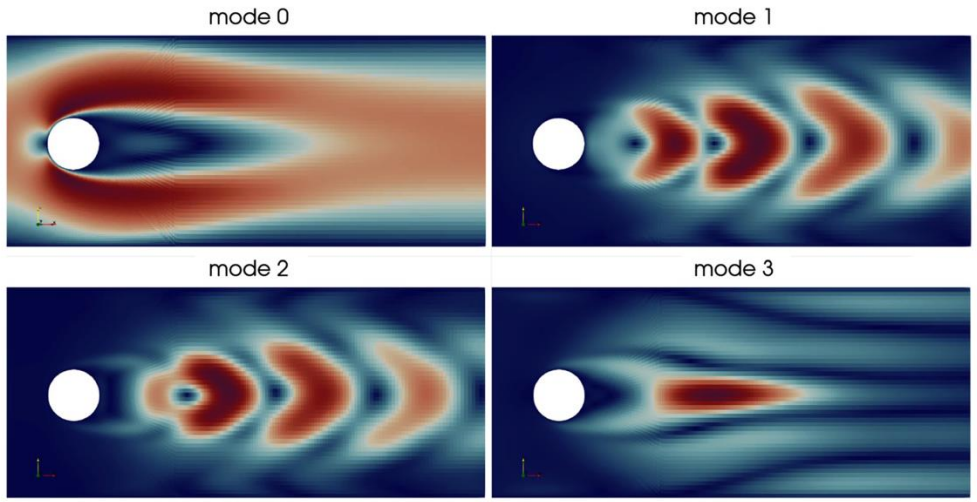
Surface relative vorticity from ¼-degree MOM6 with AI turbulence model
[Frontier 2023]

Integrating SmartSim with OpenFOAM

- **SmartSim Team and OpenFOAM Data-Driven Modeling Special Interest Group**
  - **Paper:**
    Combining machine learning with computational fluid dynamics using OpenFOAM and SmartSim
    Maric et al. [2024]

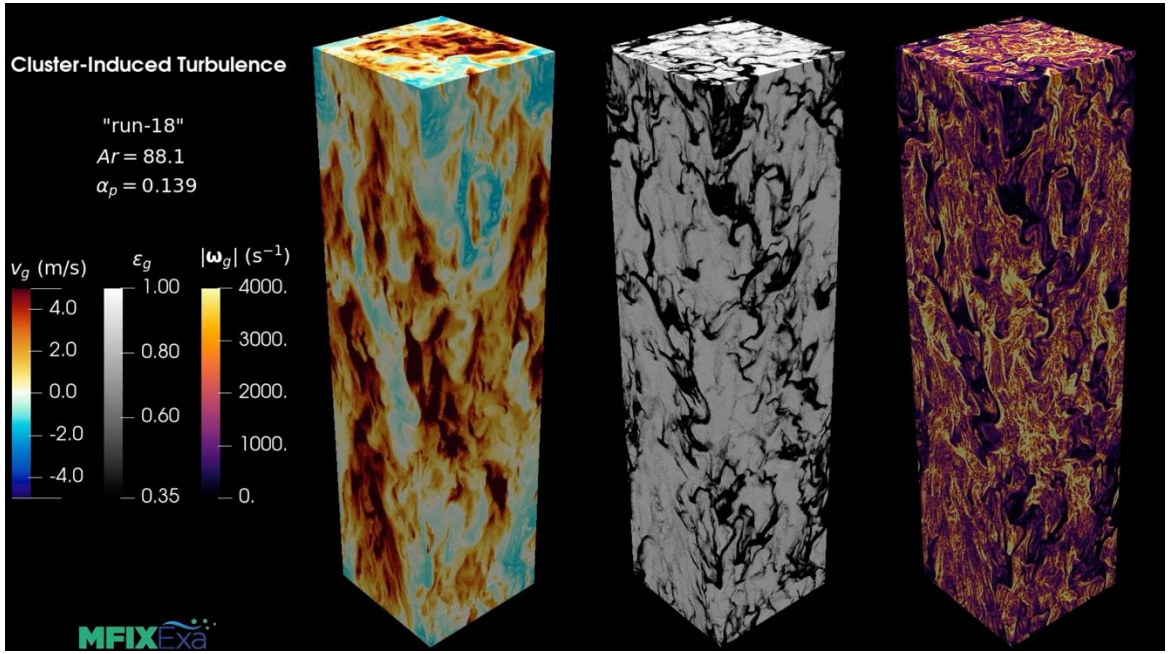
- **Examples with OpenFOAM**
  - Online training/inference for moving mesh
    - Train on boundary displacements
    - Predict on interior parts of the mesh
  - Use SmartSim to perform streaming calculations
    - PCA using distributed, partitioned SVD
    - Use cases: Data decimation, physical understanding





Training surrogate models of subgrid-scale physics online

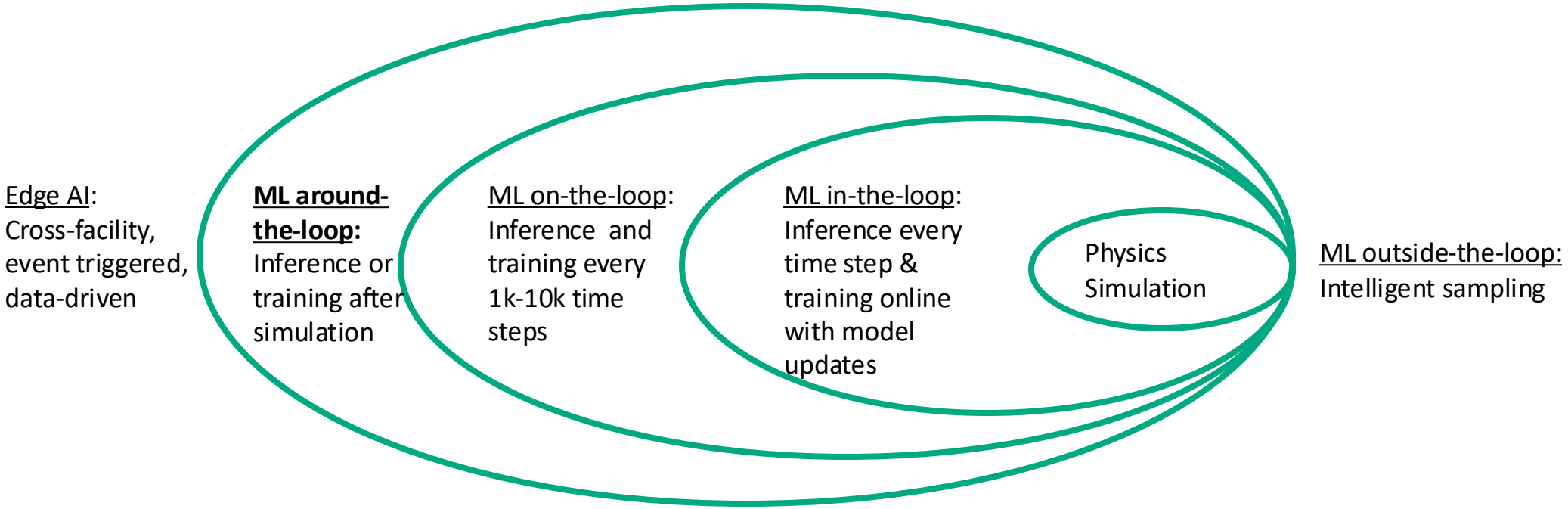
- **Typical offline training**
  - Relies on post-hoc simulation output
  - Data reduction (aliasing time/space)
  - Expensive to store
- **MFIX-Exa Application**
  - Parameter study (ensemble of 33 simulations)
  - Multi-phase, particle/fluid-based simulation
  - ~50TB of compressed data
- **Online training solution**
  - Stream timestep data from every ensemble member
  - Intelligent sampling to train on “interesting” data
  - Train ML model on data



Note: Toy version for hands-on portion of workshop

Gel, Musser, Fullmer, and Shao [2024] as part of a ASCR Leadership Computing Challenge project

ML-around-the-loop: Scientific Applications



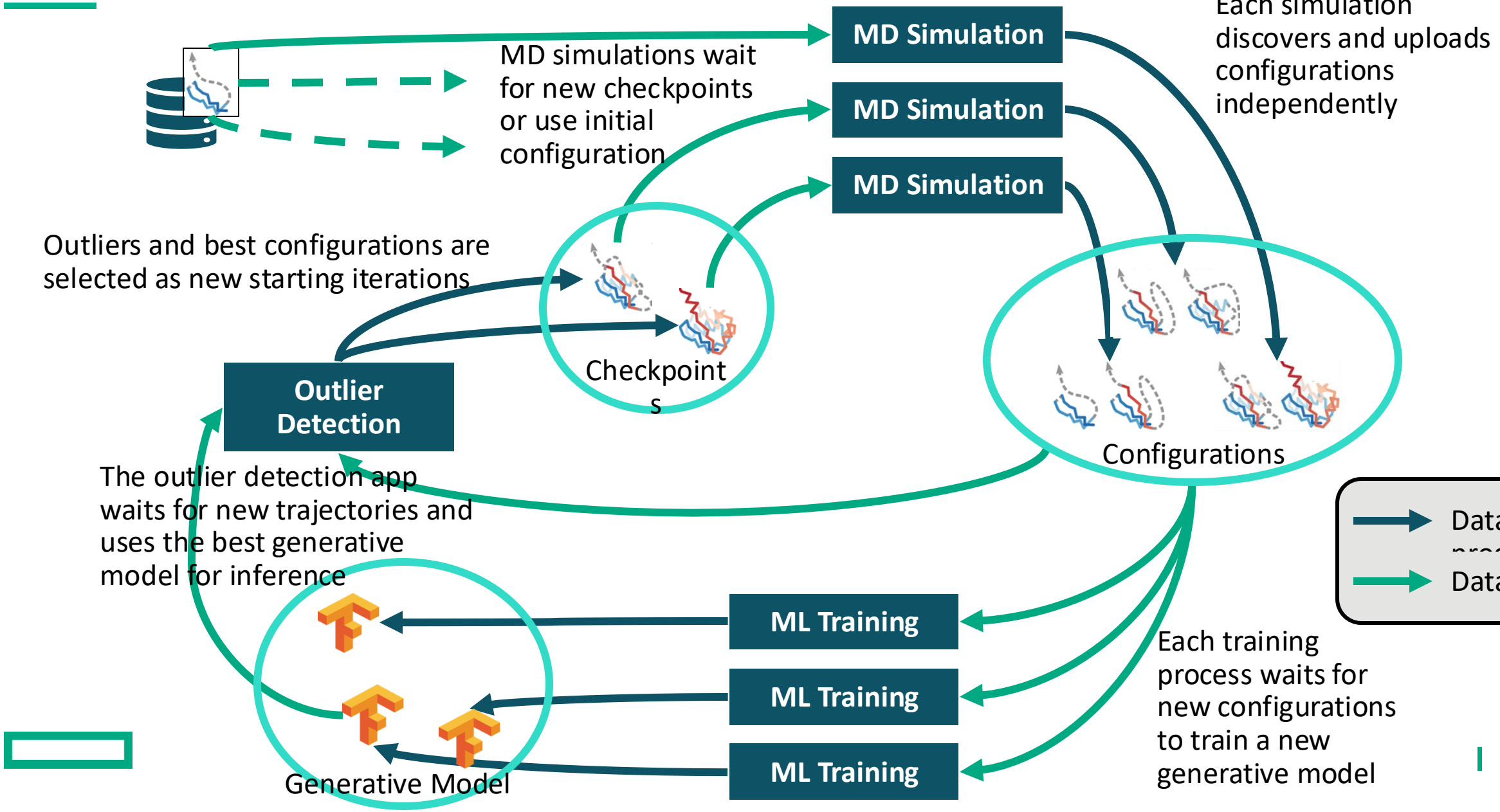
Molecular Dynamics with DeepDriveMD

- **Goal: predict the folded configuration of a protein starting from its atomic structure**
  - The original paper: DeepDriveMD: Deep-Learning Driven Adaptive Molecular Simulations for Protein Folding

- **Problems:**
  - Protein folding happens in discrete steps driven by energy minimization, but with random fluctuations
  - How can one efficiently explore the space of all possible shapes of the protein [conformation]?
    - Most trajectories will end up in suboptimal states
    - Trajectories far apart may collapse on the same one after a sufficiently large number of steps

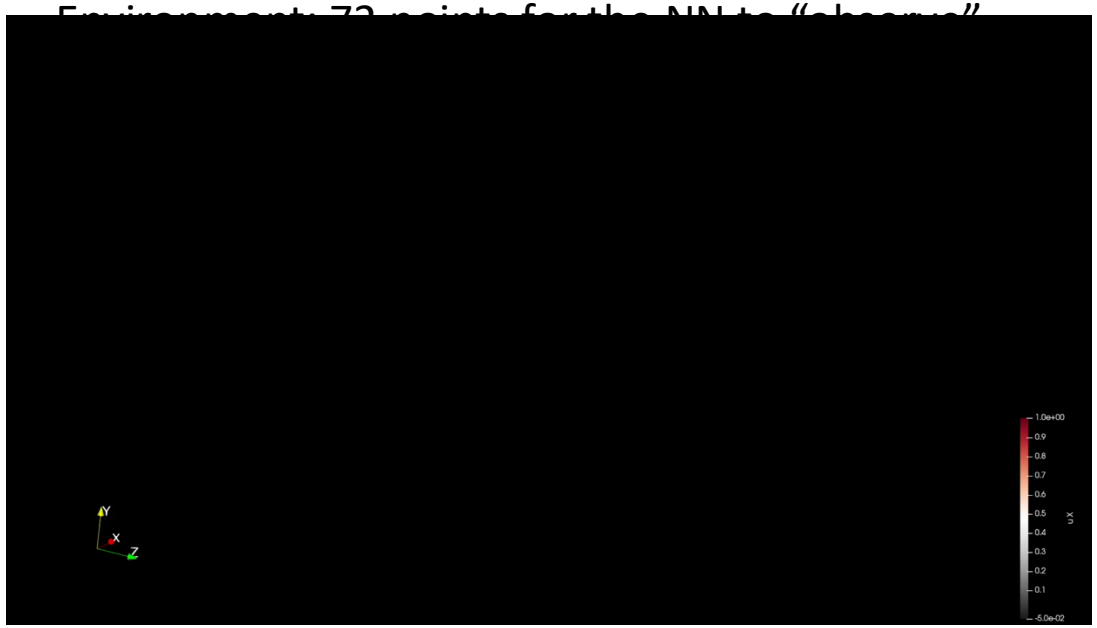
- **Solution:**
  - Run short simulations, store steps and predict possible conformations
  - Cluster discovered conformations
  - Explore less sampled regions

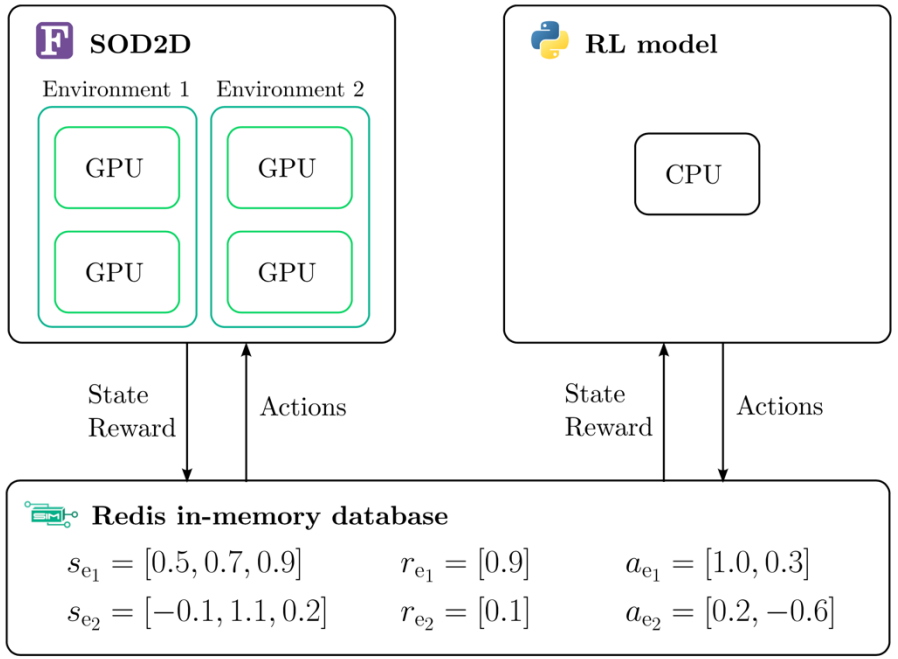
DeepDriveMD Data flow

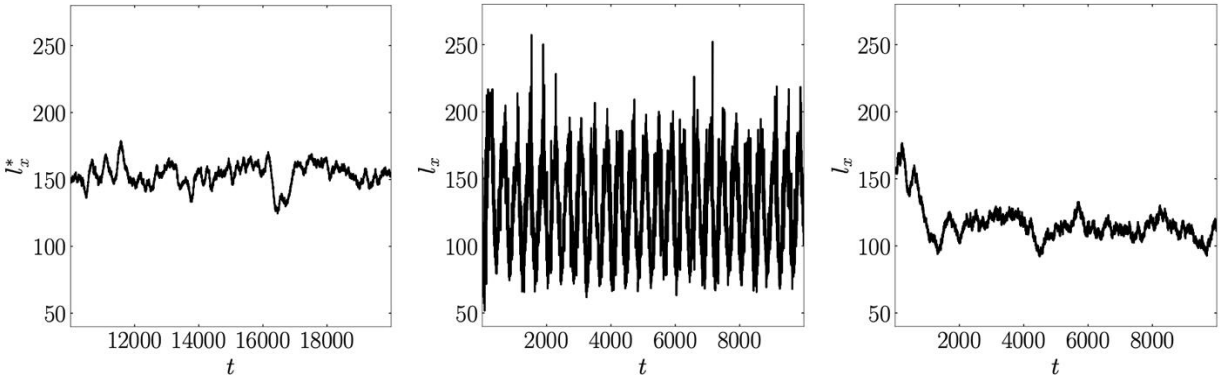


Active flow control through deep Reinforcement learning

- Goal: Reduce Turbulent Separation Bubble formation
- Method: Deep Reinforcement Learning with small NN
  - Reward based on recirculation length of turbulent bubble
    - Aim: minimize recirculation area
    - Environment: 72 points for the NN to “observe”







Scientific advancement with Simulation and AI

- **A new paradigm is emerging for computational science: Workflows**
  - A scientific simulation is only a part of a larger, more complex workflow
  - AI may be a component of the workflow
  - These workflows are difficult to describe as a directed acyclic graph (loops, conditionals)

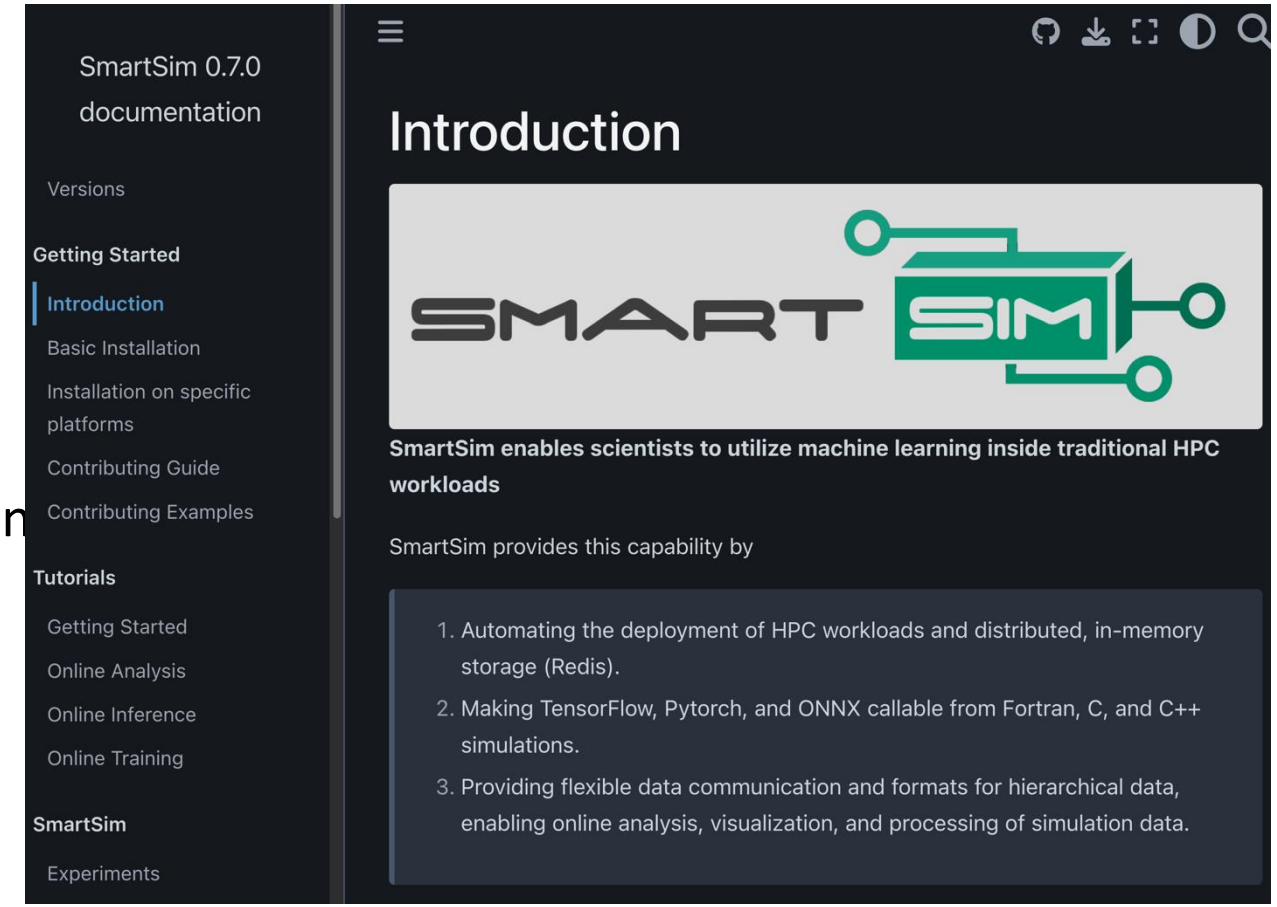
- **Most scientists can pickup ML fairly quickly**
  - ML for science can be simpler than most “splashy” AI (e.g. LLMs, generative AI)
  - Speedbumps:
    - Overcoming technical challenges for connecting simulation/AI
    - Expressing the workflow

- **AI+Simulation is open for innovation and experimentation**
  - Even “simple” applications provide new opportunities for scientific discovery

- **SmartSim team is open to collaboration**
  - Thanks to all our existing collaborators at MLCommons, OpenFOAM, NCAR, GFDL, National Energy Technology Laboratory, Argonne National Lab, Oak Ridge National Lab, M2Lines, NEMO, and the MOM6 communites

Learning more about SmartSim

- To get more information about SmartSim you can
  - Read the documentation: https://craylabs.org
  - Star SmartSim Repository: https://github.com/CrayLabs/SmartSim
  - Star SmartRedis Repository: https://github.com/CrayLabs/SmartRedis
  - SmartSim Slack workspace: https://join.slack.com/t/craylabs/shared_invite/zt-2pvvwwjjq-_f~gGxYcJVUxfoD7t5Dkfw



Hewlett Packard
Enterprise

SmartSim: Hands-on introduction

\

SmartSim Basics

```
import argparse
import pathlib

from smartsim import Experiment

# Define the top-level SmartSim object
exp = Experiment("hello_world", launcher="slurm")


# Define the settings to run a perroquet with
perroquet_run_settings = exp.create_run_settings(
   exe="echo",
   exe_args=["Hello", "World!"],
   run_command="mpirun"
)
perroquet_run_settings.set_tasks(1)

# Create a SmartSim representative of a numerical model
perroquet = exp.create_model(
   "hello_world",
   perroquet_run_settings,
)
exp.start(perroquet, block=True, summary=True)
```

- Using SmartSim
  - User writes a driver script in python
  - Add SmartRedis codes to simulation code
- SmartSim Driver
  - Create components of the workflow
    - Model (e.g. simulation)
    - RunSettings (e.g. how to run)
    - Database
  - Create run directories
  - Launches components

Fortran Application Example

```
use smartredis_client, only : client_type

! Format the suffix for a key as a zero-padded version of the rank
write(key_suffix, "(A,I1.1)") "_",pe_id

! Initialize a client
result = client%initialize("smartredis_mnist")

! Set up model and script for the computation
if (pe_id == 0) then
   result = client%set_model_from_file(model_key,            model_file, "TORCH", "CPU")
   result = client%set_script_from_file(script_key, "CPU", script_file)
endif

result = client%put_tensor(in_key, array, shape(array))
! Prepare the script inputs and outputs
inputs(1) = in_key
outputs(1) = script_out_key
result = client%run_script(script_name, "pre_process", inputs, outputs)
inputs(1) = script_out_key
outputs(1) = out_key
result = client%run_model(model_name, inputs, outputs)
result = client%unpack_tensor(out_key, output_result, shape(output_result))
```

C++ Application Example

```
#include "client.h"

// Get our rank
int rank = 0;
MPI_Comm_rank(MPI_COMM_WORLD, &rank);
std::string logger_name("Client ");
logger_name += std::to_string(rank);

// Initialize a SmartRedis client
SmartRedis::Client client(logger_name);

// Put the tensor in the database
std::string key = "3d_tensor_" +          std::to_string(rank);
client.put_tensor(key, input_tensor.data(), dims,
                  SRTensorTypeDouble,
                  SRMemLayoutContiguous);
```

```
// Retrieve the tensor from the database using the
unpack feature.
std::vector<double> unpack_tensor(n_values, 0);
client.unpack_tensor(key, unpack_tensor.data(),
          {n_values},
          SRTensorTypeDouble,
          SRMemLayoutContiguous);


 // Retrieve the tensor from the database using the get
feature.
    SRTensorType get_type;
    std::vector<size_t> get_dims;
    void* get_tensor;
    client.get_tensor(key, get_tensor, get_dims,
          get_type, SRMemLayoutNested);
```

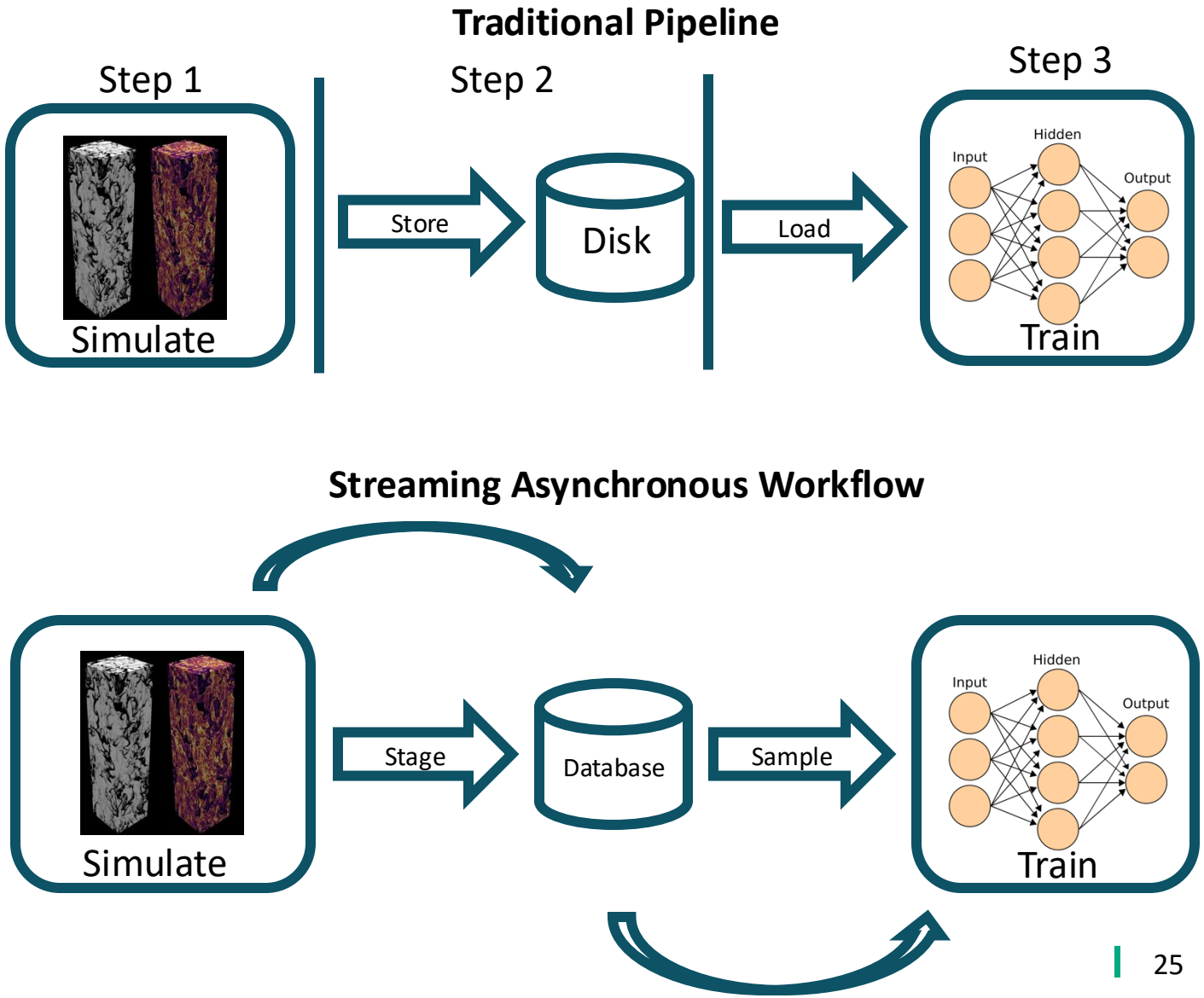
Hewlett Packard
Enterprise

SmartSim: Building a workflow for online training

\

Why online training?

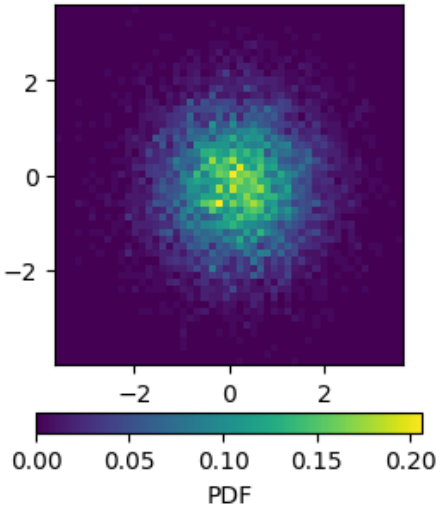
- **Numerical surrogates need timestep-level data**
  - Can lead to large data volumes
- **Generally want to map a function-space to another function space**
  - Oversampling a portion of the sample space biases model
- **Questions:**
  - How do you store this amount of data?
  - How do you train on this amount of data?
- **Solution:**
  - Sample the data in an ‘intelligent’ way
  - Train surrogate in a streaming manner

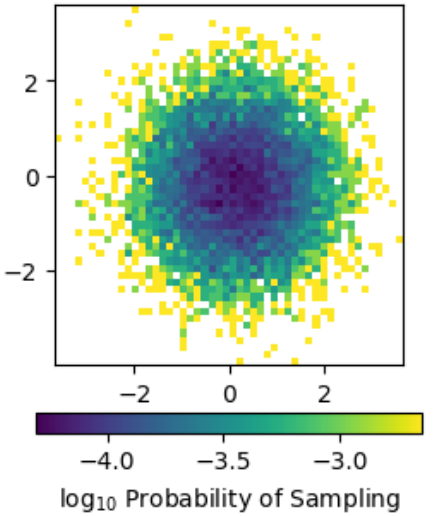


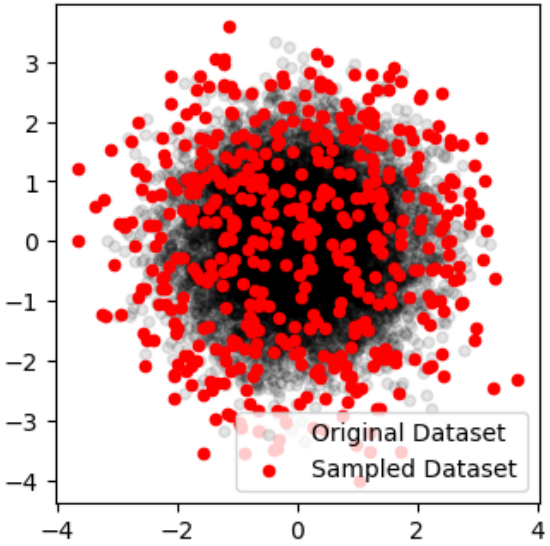
Intelligent sampling for point-by-point predictions

- **Problem: AI susceptible to sampling bias**
  - With most PDEs, no part of the solution space is more “valid”
  - Naïve training of AI models leads to
    - Fixating on the well-sampled parts of the domain
    - Ignoring the outliers
- **Solution: Intelligently sample the data to promote uniform sampling**
  - For low-dimension data
    - Calculate PDF of data
    - Use inverse of PDF as a sampling “chance”
  - For high-dimension data
    - PDF expensive to calculate
    - Use Generative AI techniques to estimate PDF

Hassanaly M, Perry BA, Mueller ME, Yellapantula S. Uniform-in-phase-space data selection with iterative normalizing flows. *Data-Centric Engineering*. 2023







Setting up the asynchronous, workflow

Mock Simulation

- Store 16 data points per “timestep”
- Stage in database
  - Sent using SmartRedis
  - Stored in SmartSim database

Intelligent Sampler

- Polls database for new datasets
- Performs statistical comparison to accept/reject new samples
- Stores downsampled data in database
- Delete original data

Trainer

- Check for new training data
- Whenever new data is available, do a training step

Hands-on Portion of Workshop

https://github.com/CrayLabs/smartsim_workshops/tree/nersc_olcf_2024

Objectives:

- Create a workflow with multiple components based on data-availability
- Instrument a C++/Fortran code with SmartRedis
- Use SmartSim's integration with pyTorch dataloaders

Questions?

Andrew.Shao@hpe.com

For more information about SmartSim: https://craylabs.org