Hewlett Packard Enterprise

SMARTSIM

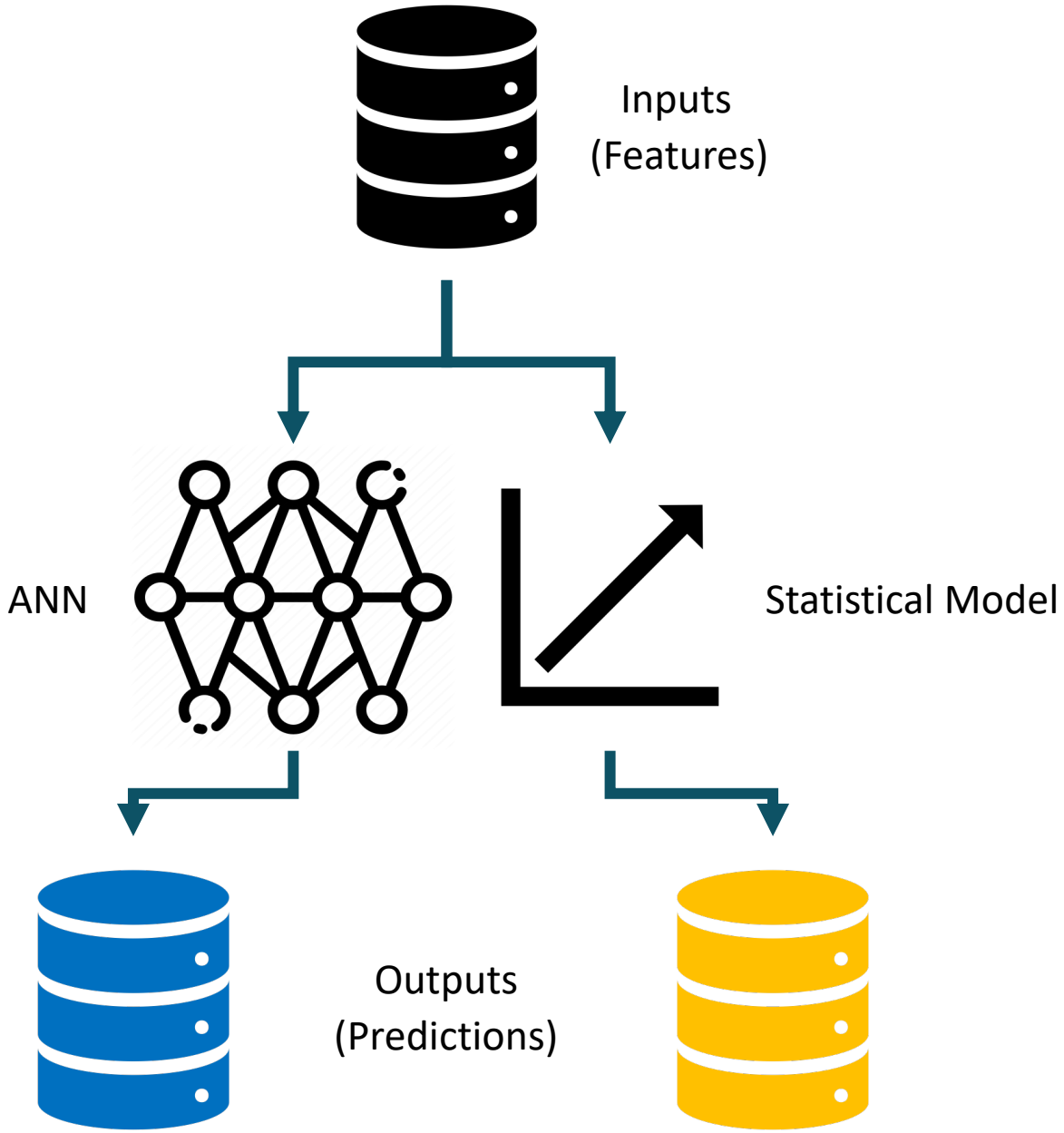
SMARTSIM: ENHANCING HPC WORKFLOWS WITH MACHINE LEARNING

Andrew Shao, PhD | Senior HPC&AI Research Scientist, HPE
13 July 2023
andrew.shao@hpe.com

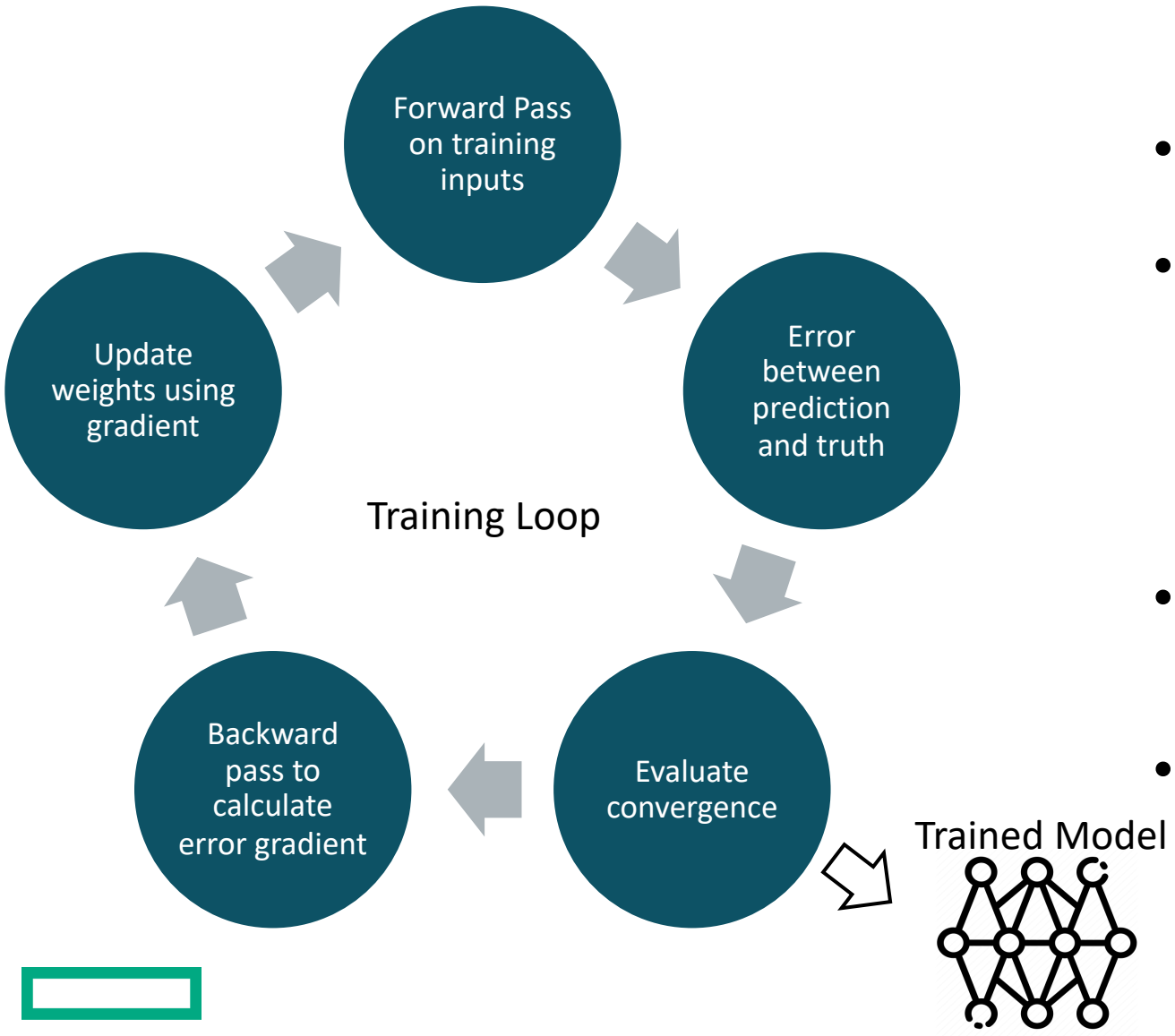
ARCHETYPES OF MACHINE LEARNING

- Supervised learning:
  - Given these inputs and known outputs, derive a relationship
  - Linear regression falls under supervised learning
  - **Artificial neural network:**
    - Linear and non-linear transformations with many free parameters
    - Useful especially when the basis is unknown
- Reinforcement learning
  - Train model to take actions within a given ruleset based on predefined reward function
    - Play a video game with these inputs that gives you a high score
- Unsupervised Learning (not discussed today)
  - Find relationships in unlabeled data

All archetypes rely on data to learn, generalize, and predict



MACHINE LEARNING TASKS



- Forward pass
  - Pass inputs through a model with a given set of weights (i.e. values) for every free parameter
- Backward pass
  - Calculate the gradient of the error w.r.t. the weights
- Training Loop (similar to optimization)
  - Perform a **forward pass** with given **weights** and features
  - Calculate error compared to expected output
  - Use **backward pass** to estimate gradient
  - Update the weights
- Trained model:
  - Architecture (layers, layer types, etc.)
  - Weights (values of the free parameters)
- Inference: Forward pass of a trained model to make a new prediction

CHALLENGES IN EMBEDDING ML INSIDE AND OUTSIDE OF SIMULATIONS

Technical

- How can we call an ML model from a Fortran/C/C++ code?

- How can we scale this to the needed compute scales?
  - ML models in physics simulations tend to be small
  - Want to efficiently allocate GPU resources

These technical challenges impede scientific progress!

- Naturally focuses the community on how to replace simulations with AI
- AI to solve physics problems tends to be in idealized contexts or static data
- Scientists are asking the questions, but do not have the tools to test this practically

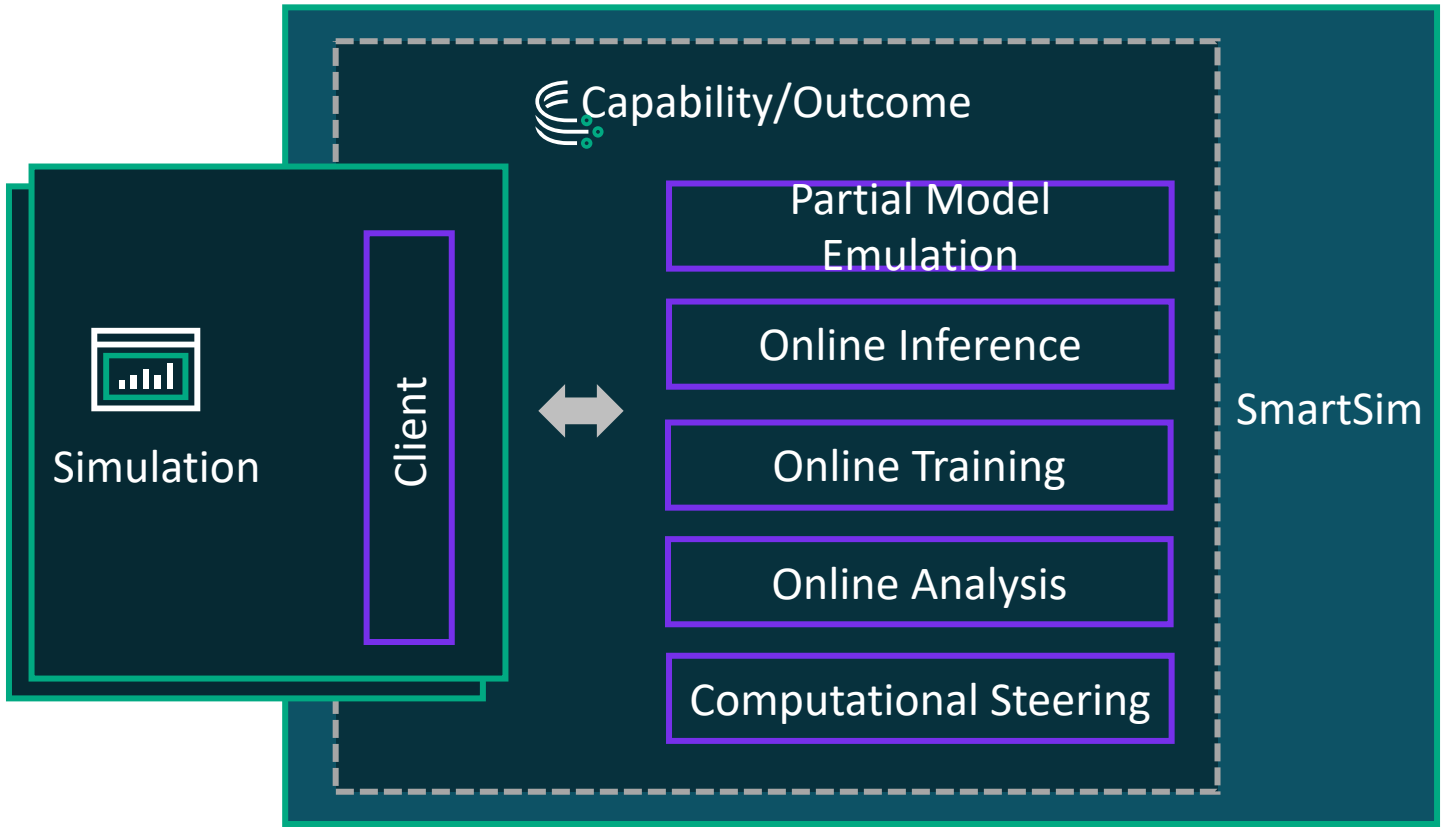
SMARTSIM ENABLES AI-ENHANCED SIMULATIONS

Lower the barriers to entry

- Enable Fortran, C, C++ simulations to interact with ML packages efficientl y
- Rapidly prototype and iterate on new ML models
- Allow scientists to focus on building applications *not* code or infrastructure

Introduce a new workflow paradigm

- Simulations as producers and consumers of data and ML models
- Data is kept in-memory and available in-flight



SMARTSIM OVERVIEW

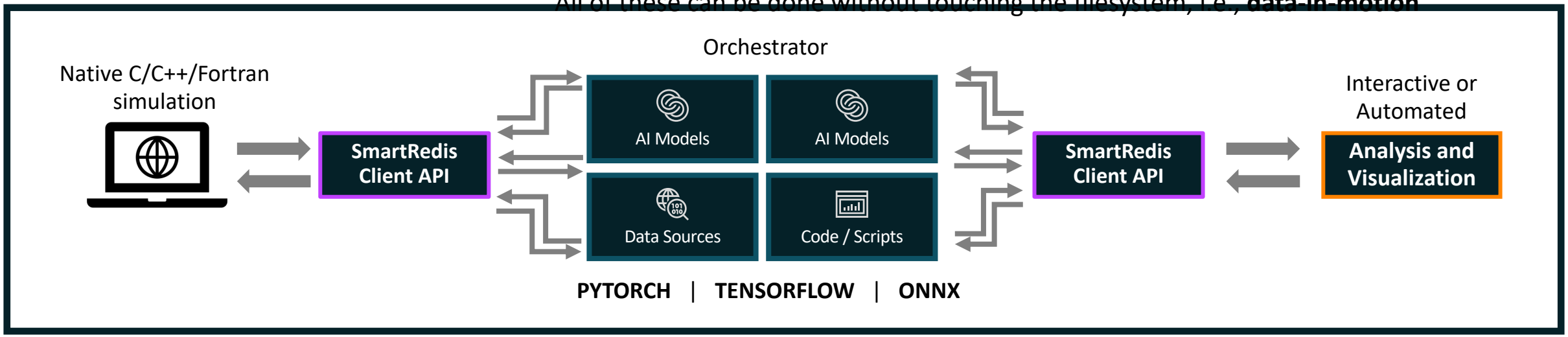
ABOUT SMARTSIM

The SmartSim open-source library bridges the divide between traditional numerical simulation and data science

SmartSim enables simulations to be used as engines within a system, producing data, consumed by other services to add value and create **new applications**

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

All of these can be done without touching the filesystem, i.e., **data-in-motion**



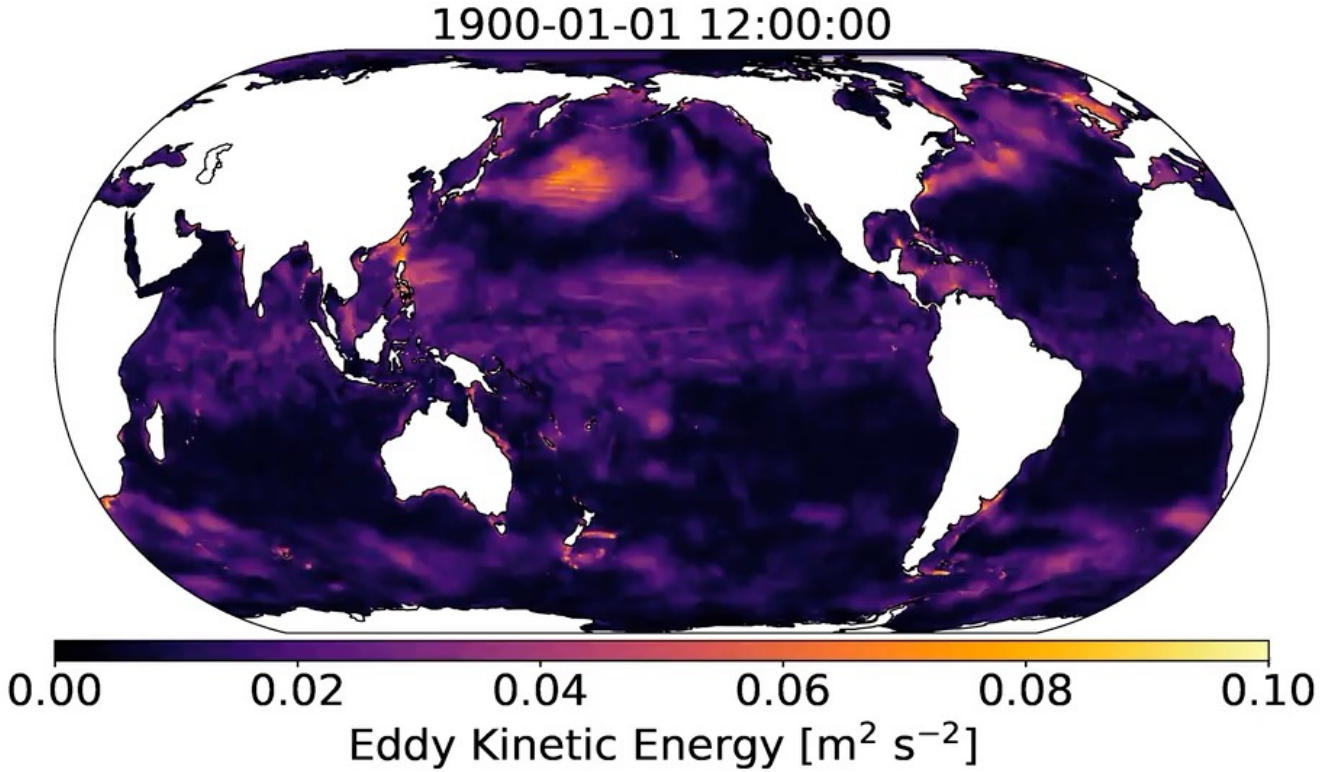
SMARTSIM AND SIMULATION INTEGRATIONS

Computational Fluid Dynamics:
OpenFOAM, FLEXI, PHASTA, libCEED, NekRS (in progress)

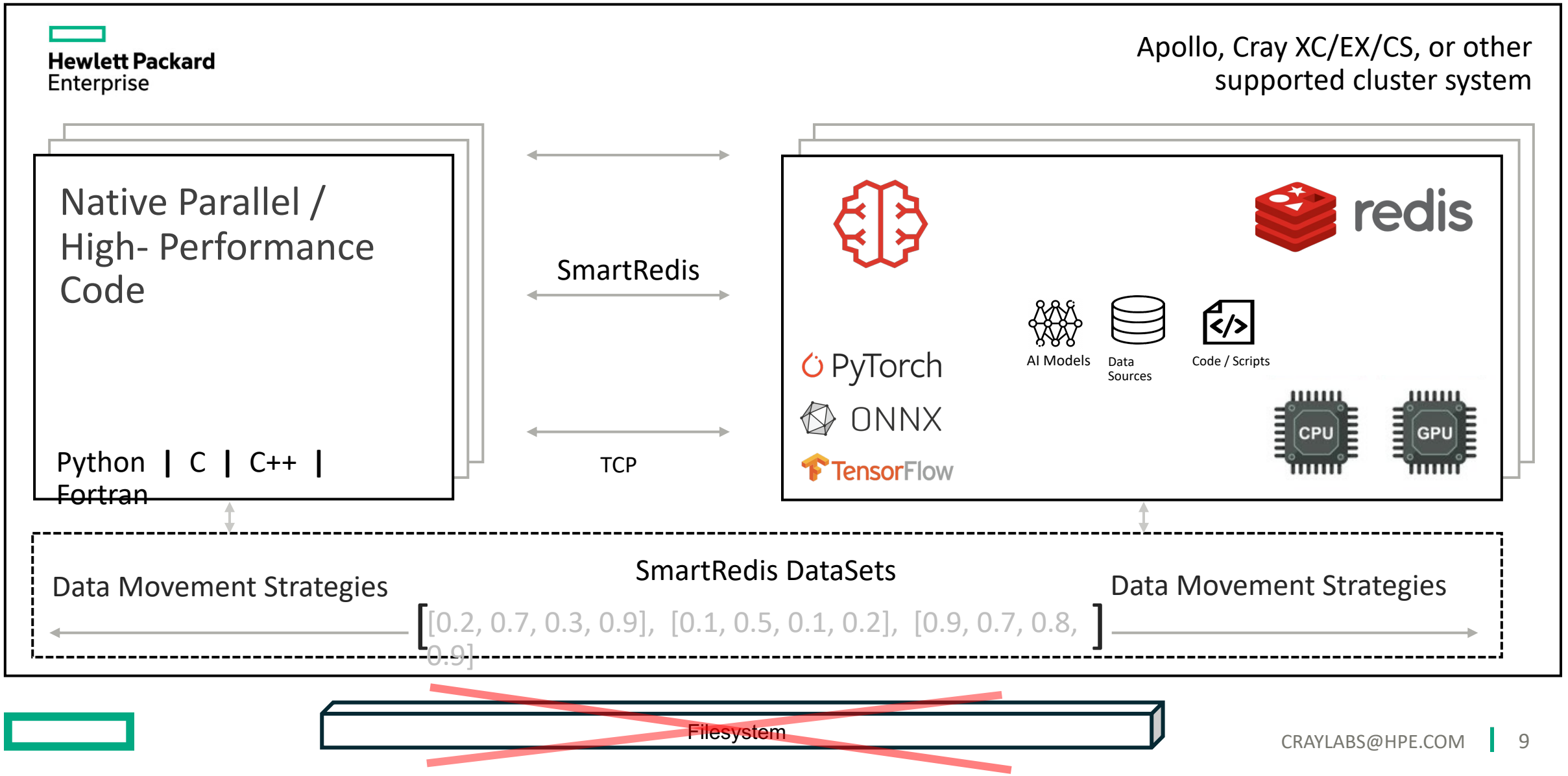
Climate and Weather:
MOM6, NEMO, CESM

Molecular Dynamics
LAMMPS, OPENMM

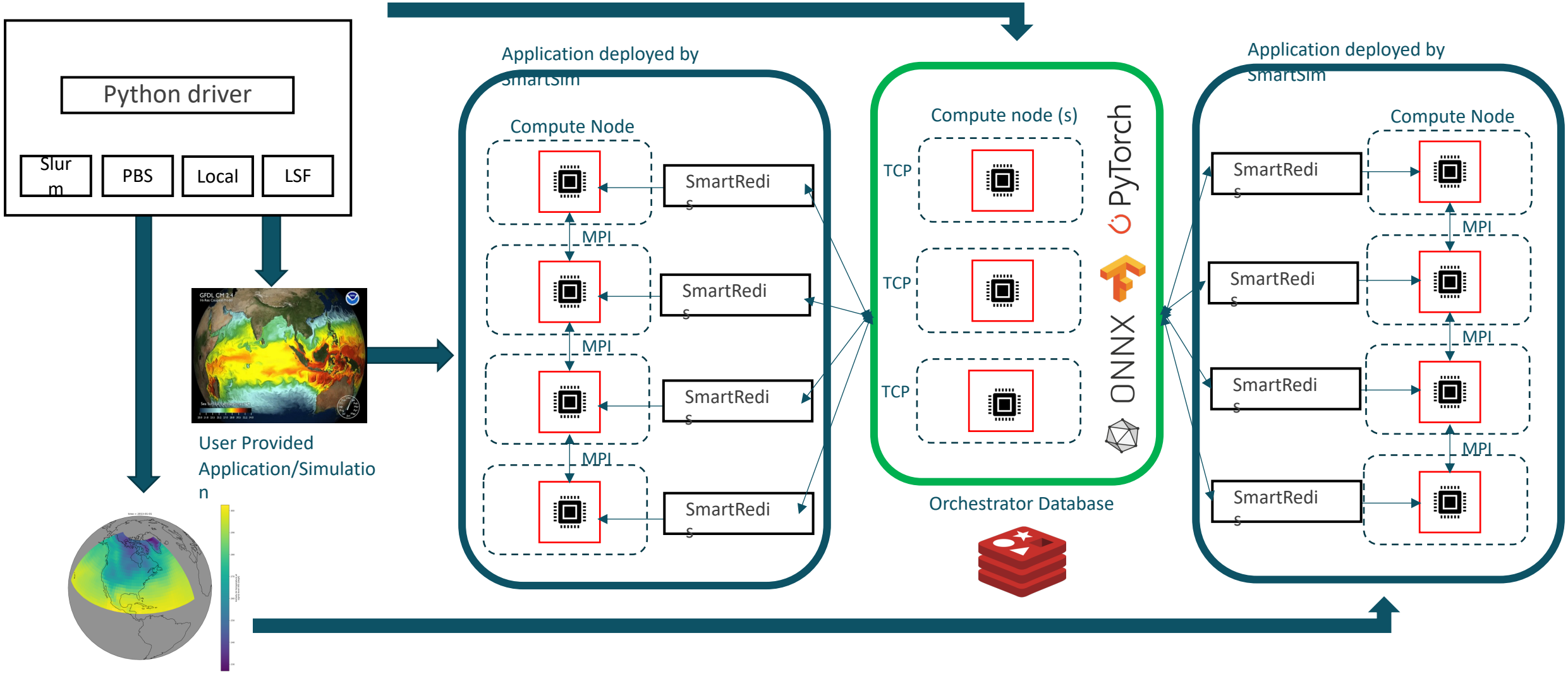
Easy to instrument an **existing codebase** to enable new AI/ML/data science **applications**
- Very little new code needs to be added



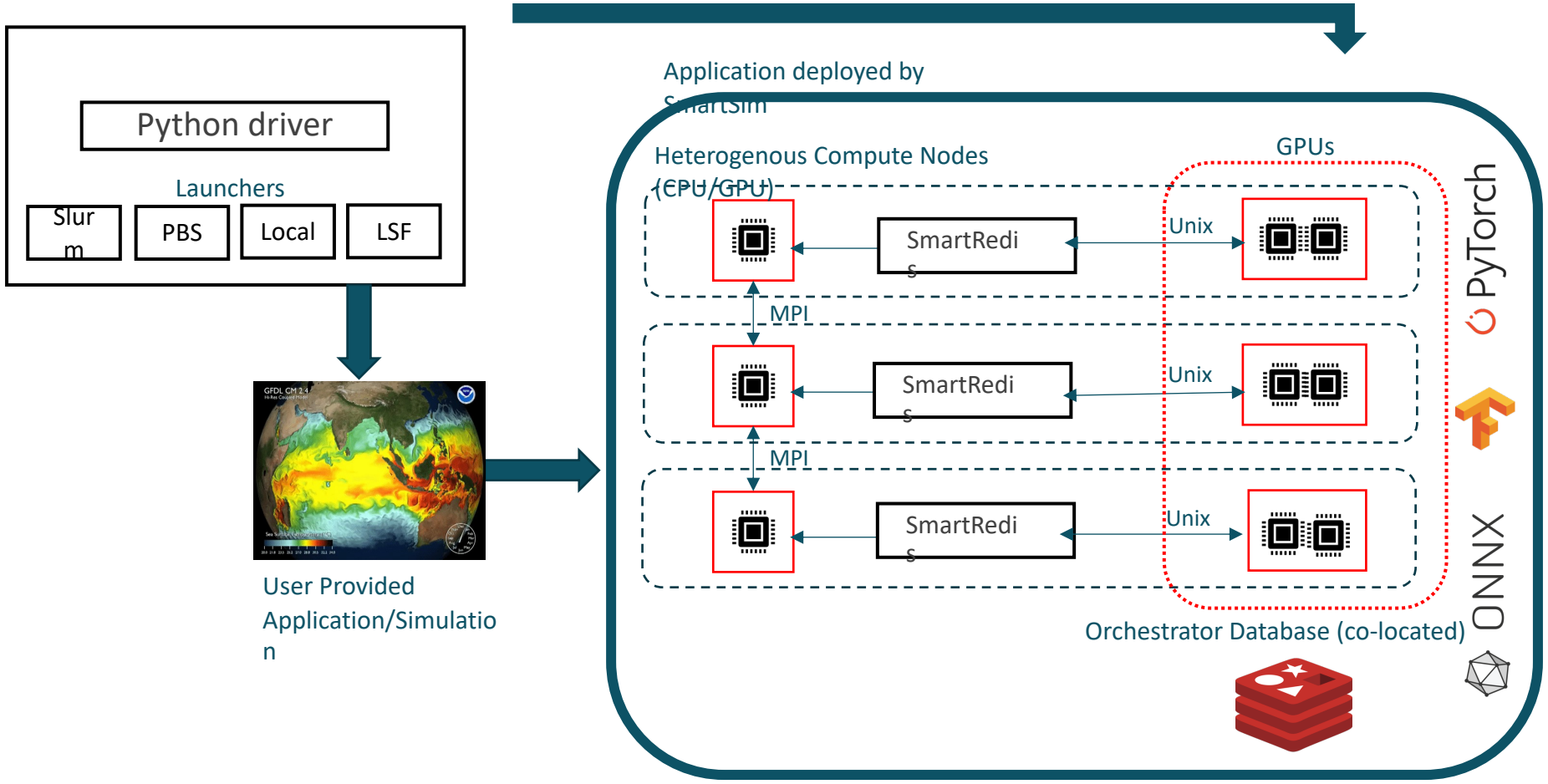
HIGH-LEVEL ARCHITECTURE



CLUSTERED DATABASE



CO-LOCATED DATABASE

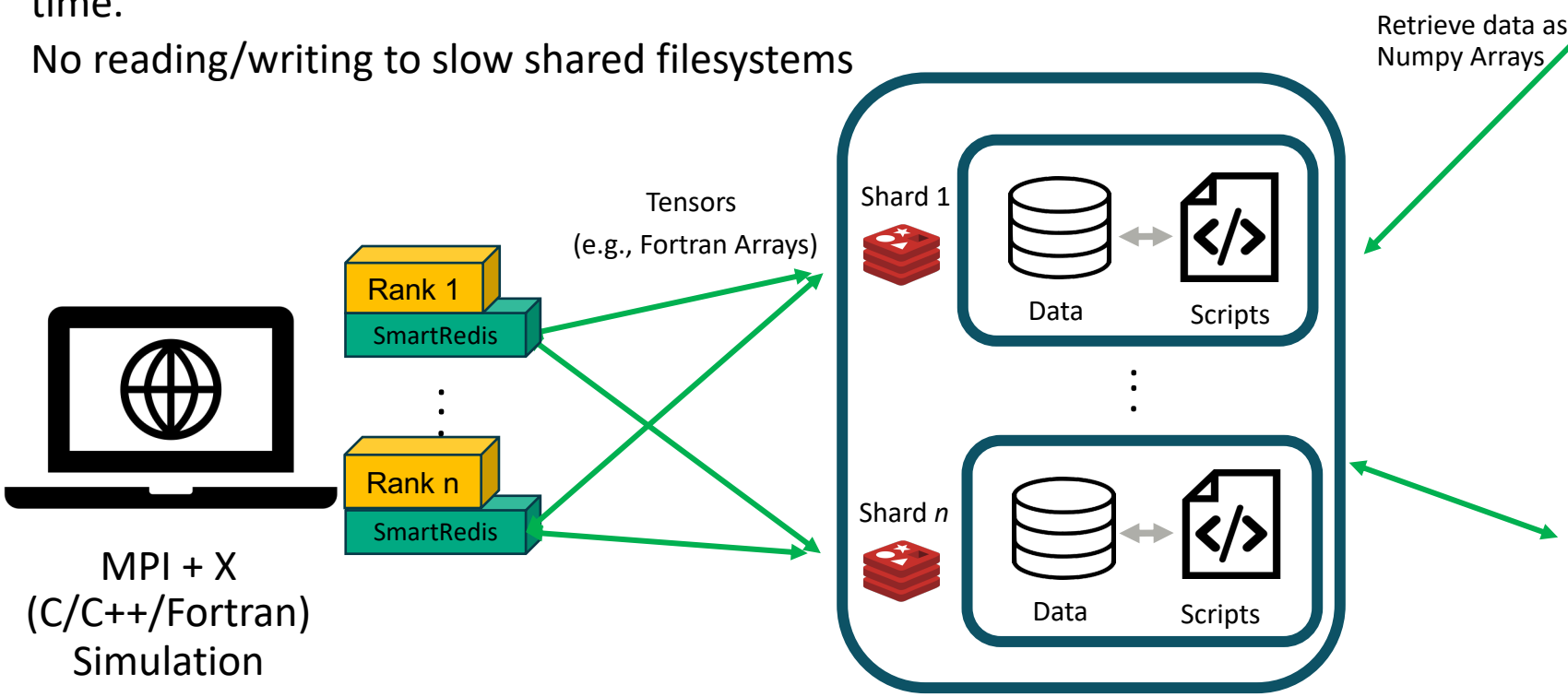


PROTOTYPICAL USE CASES

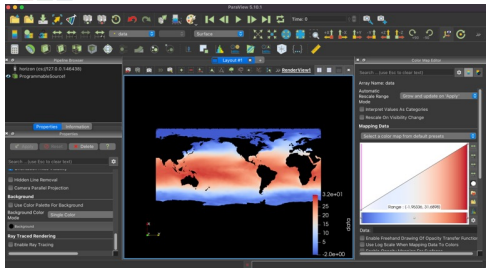
ONLINE VISUALIZATION AND ANALYSIS

I want to perform visualization and analysis while my simulation is running

- Stream data from C/C++/Fortran simulations for analysis, and visualization in real time.
- No reading/writing to slow shared filesystems



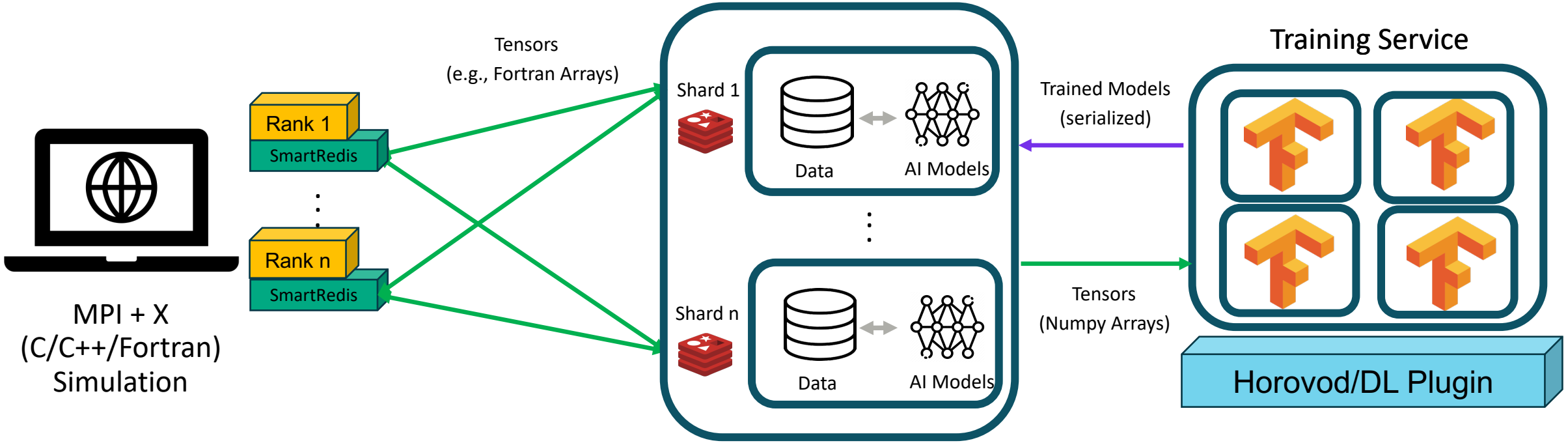
Retrieve data as Numpy Arrays



CONTINUOUS ONLINE TRAINING

I want to create machine learning models on simulation data without performing application collectives or writing to the filesystem

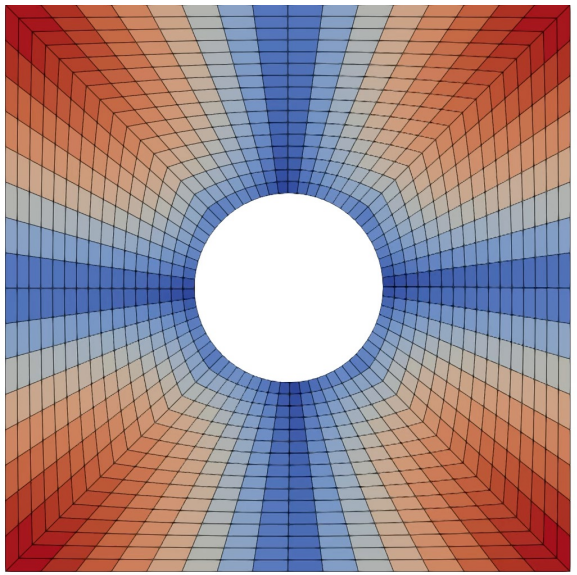
- SmartRedis Client can be used inside DataLoaders for TensorFlow and PyTorch to perform Stream Training
- Trained models can be checkpointed, saved and later sent to the database for online inference.
- Can be set from any language and called from any language (e.g., model set from Python, called from Fortran)

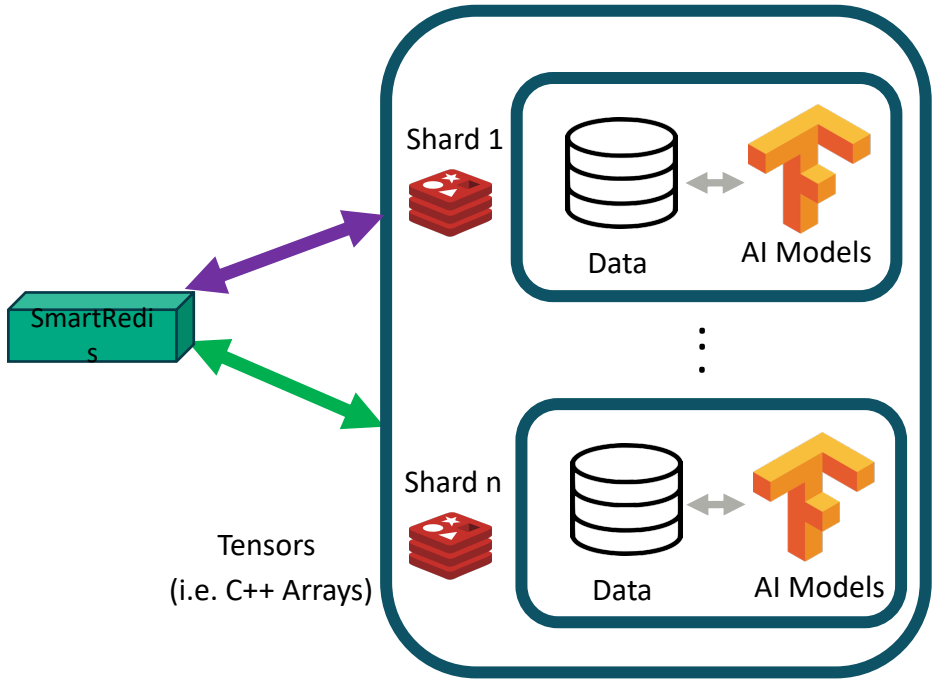


ONLINE MACHINE LEARNING TRAINING/INFERENCE

- **I want to use continuously train a machine-learning model from my simulation and use the updated model for inference inside my solver**

OpenFOAM





"Orchestrator"

Datastore and inference engine

- Task:
  Need to use velocities to understand how to optimally deform the mesh
- ML Workflow
  1. Extract displacements
  2. Train model on displacement between boundary and mesh boundary
  3. Use ML model to predict mesh displacement for unstructured mesh

Collaboration with OpenFOAM Data-Driven Modelling Special Interest Group (Maric and Weiner)

Legend

Data to Orchestrator

ML inference results to simulation

Molecular Dynamics with DeepDriveMD

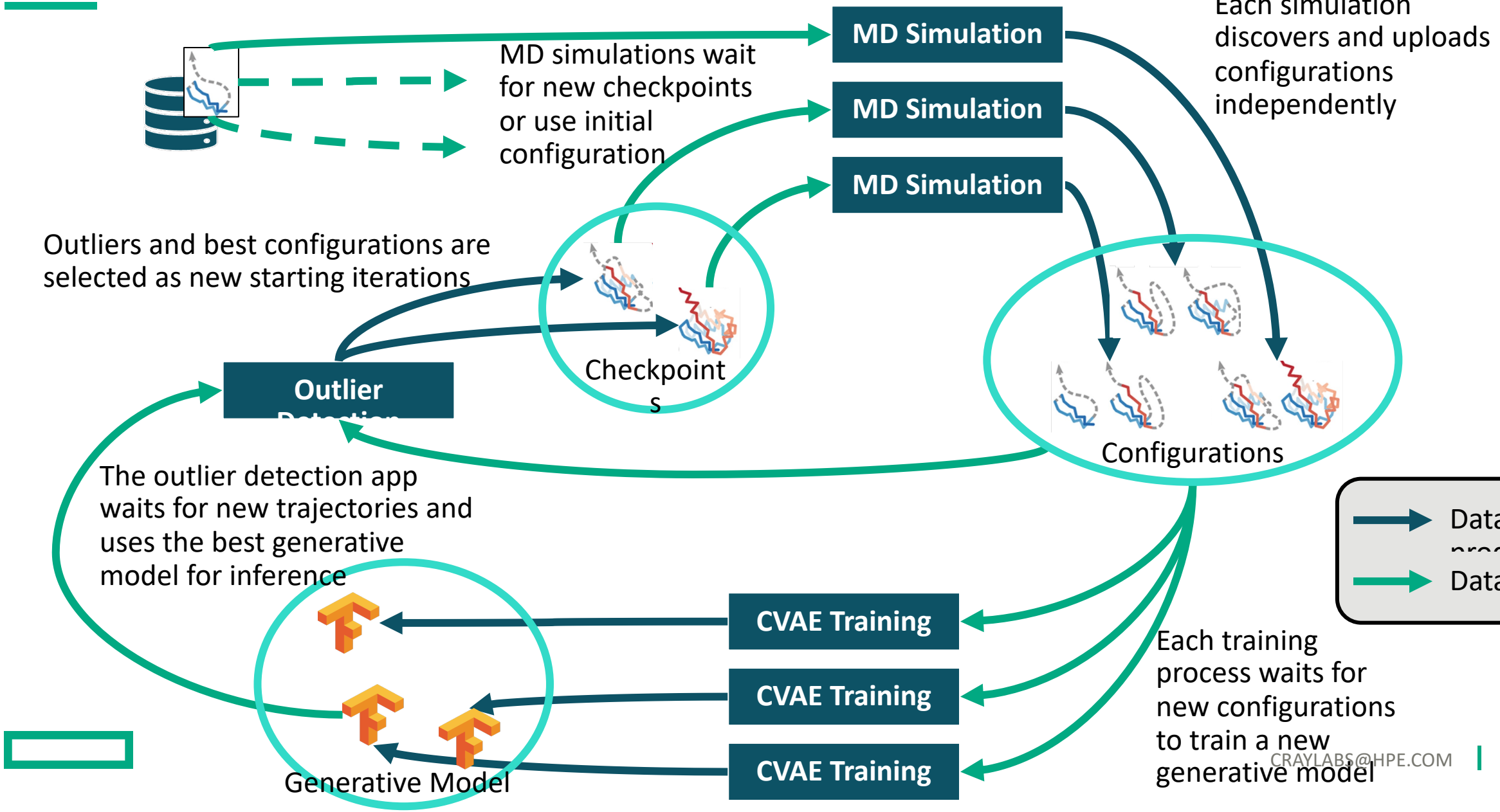
Molecular Dynamics with DeepDriveMD

- 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:
  - Each protein is composed of hundreds or thousands of atoms
  - Protein folding happens in discrete steps which are driven by energy minimization, but with random fluctuations
    - Molecular Dynamics simulation compute evolution of protein and store steps at regular time intervals: a collection of steps is called a trajectory
  - How can one efficiently explore the space of all possible conformations?
    - 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
  - Project conformations on low-dimensional space (latent space of a Convolutional Variational Autoencoder)
  - Cluster discovered conformations
  - Explore less sampled regions of the low-dimensional space

DEEPDRIVEMD DATA FLOW



RELEXI – RL+CFD+SMARTSIM

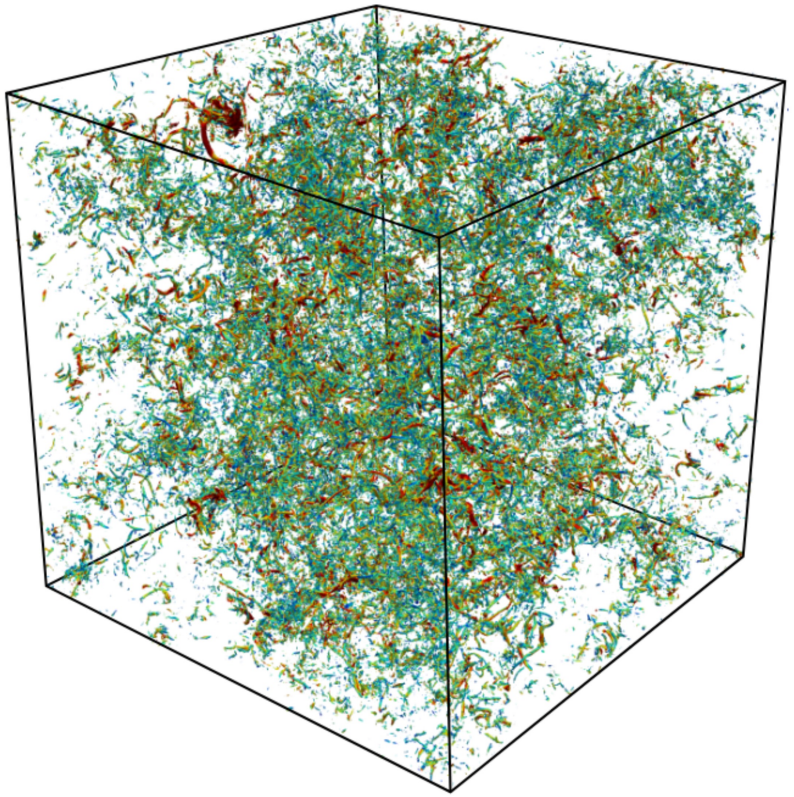
REINFORCEMENT LEARNING IN CFD

- **CFD problems that can be posed as RL**
  - Training a neural network to behave like a turbulence model
  - Design and test an active flow control system

Modeling Assumption:

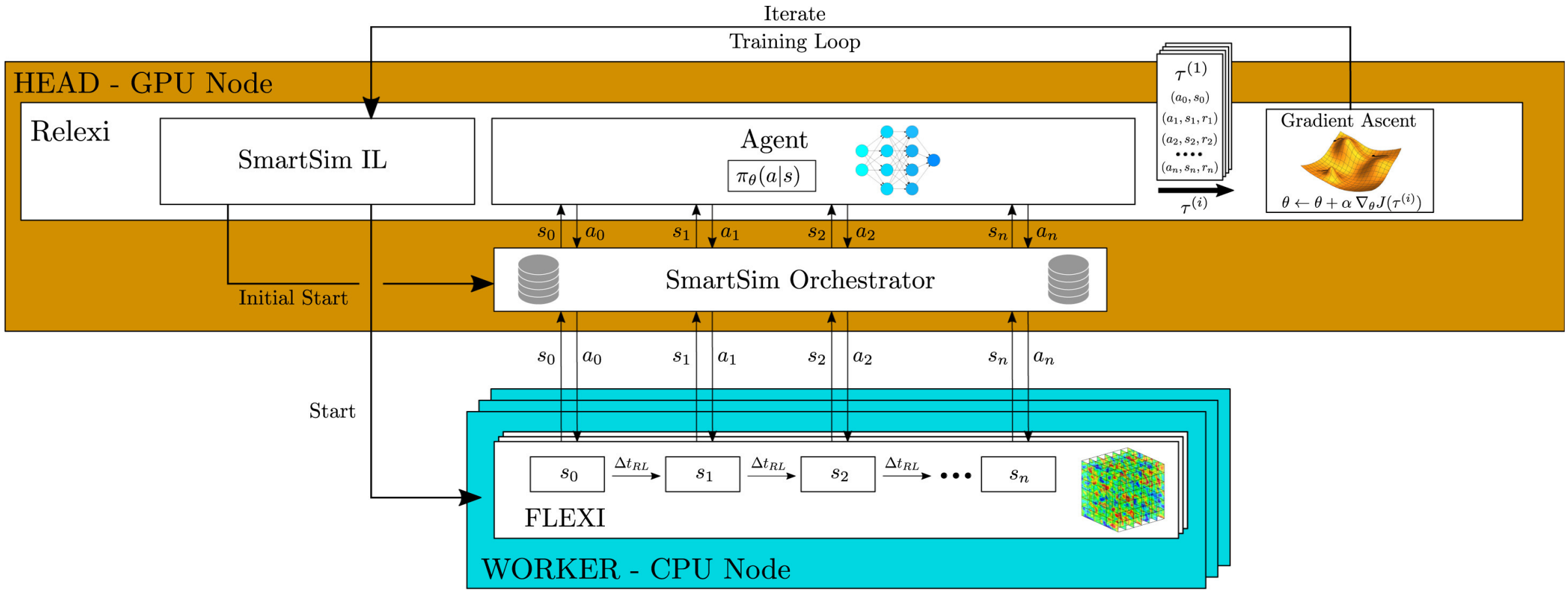
Full Solution ~ Coarse Solution + Turbulence Model

- Supervised Learning
  - Inherent risk because ML model does not interact with the simulation -> instability, crashes, etc.
- Reinforcement Learning solution:
  - Train a model of turbulence in-situ to ensure stability
  - Agent: predict Smagorinsky coefficient based on local u, v in LES
  - Reward function: Compare LES+ML turbulent spectra to DNS solution

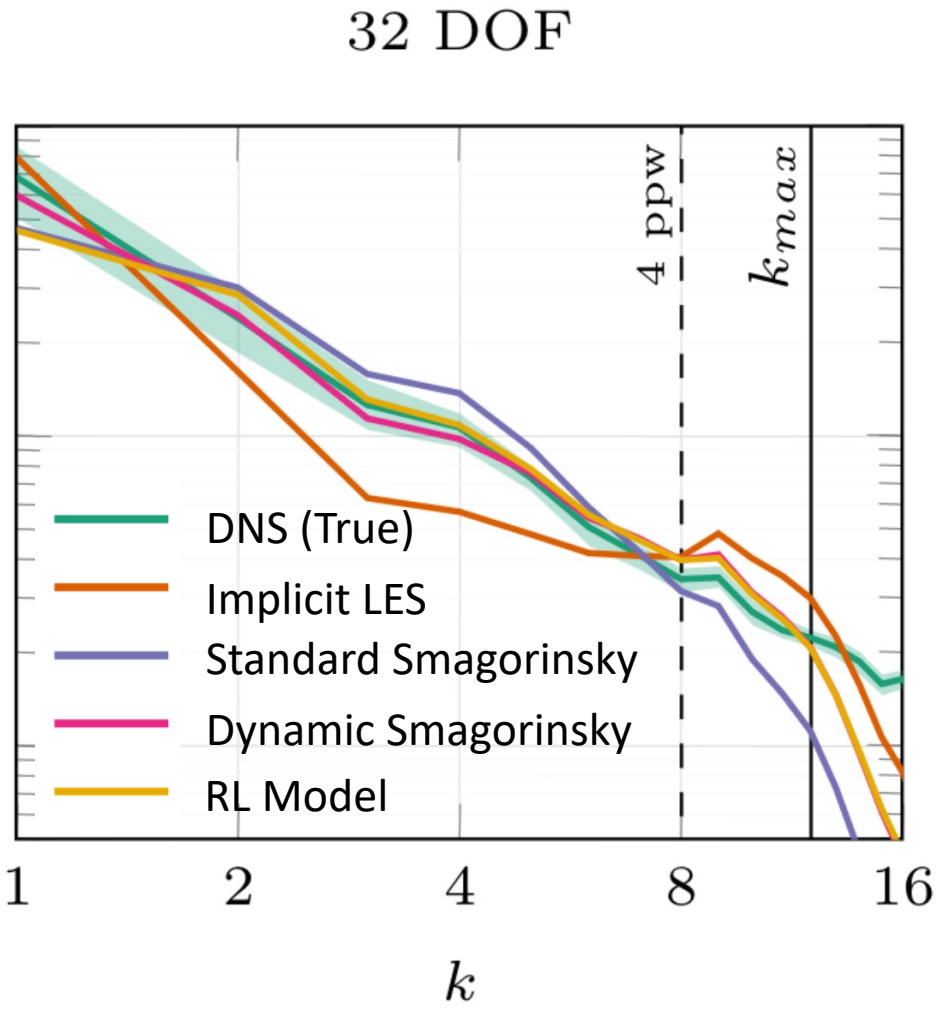


Work done by Kurz, Offenhauser, and Beck [2022], https://doi.org/10.1016/j.ijheatfluidflow.2022.109094

ARCHITECTURE OF RELEXI



RL MODEL REPRODUCES “TRUE” SPECTRA



- RL Model convergence takes ~1 day
  - 1,024 CPUs (8 nodes), 8 Nvidia A100s (1 node)
- Closely reproduces energy spectra down to resolved wavenumbers
  - Skill beyond even the ‘resolvable’ limit
  - Some generalizability to different Reynolds numbers
- Incurs about ~10% performance cost
  - Small in comparison to Implict LES
- Flexibility in defining “reward” function allows different/multiple objectives

RL FOR ACTIVE FLOW CONTROL

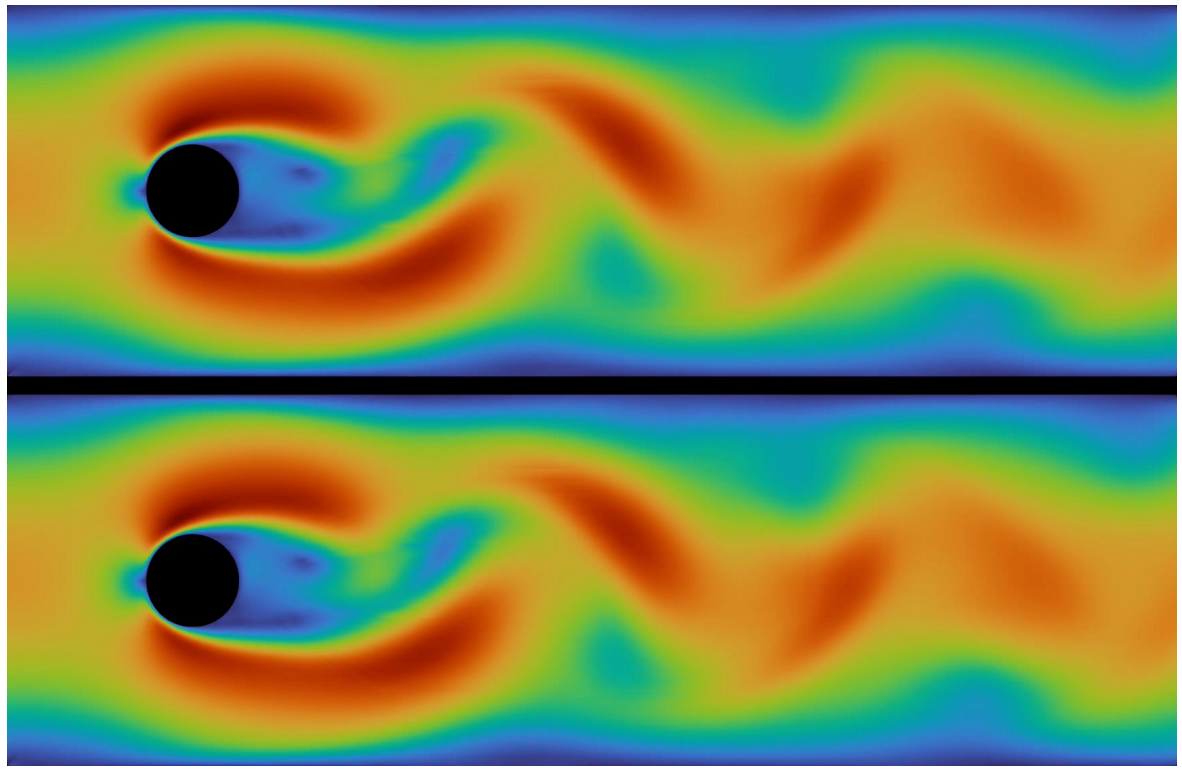
Problem Statement:

- Reduce drag by controlling ‘jets’ at top and bottom of cylinder that inject momentum

RL Solution

- Find an optimal policy to control ‘jets’ based on information at 5 probe locations that minimizes

$$r_t = -\langle c_D \rangle_{avg} - 0.2\,|\langle c_L \rangle_{avg}|$$



Uncontrolled

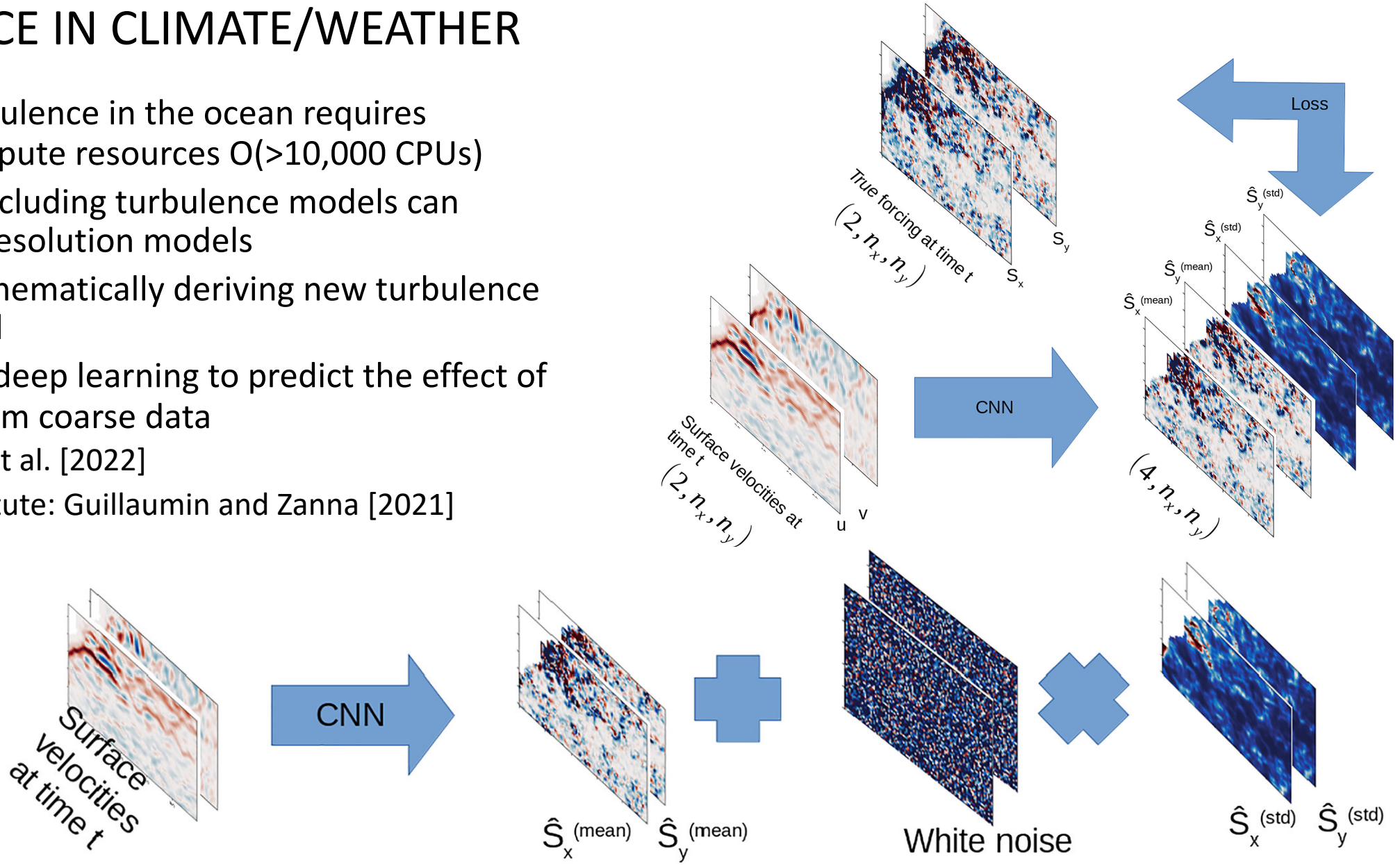
Optimal RL Policy

TURBULENCE MODELLING WITH ONLINE INFERENCE IN CLIMATE/WEATHER

Recent experiment done

TURBULENCE IN CLIMATE/WEATHER

- Resolving turbulence in the ocean requires excessive compute resources O(>10,000 CPUs)
- Hypothesis: Including turbulence models can improve low-resolution models
- Problem: Mathematically deriving new turbulence models is hard
- Solution: Use deep learning to predict the effect of turbulence from coarse data
  - HPE: Partee et al. [2022]
  - Courant Institute: Guillaumin and Zanna [2021]



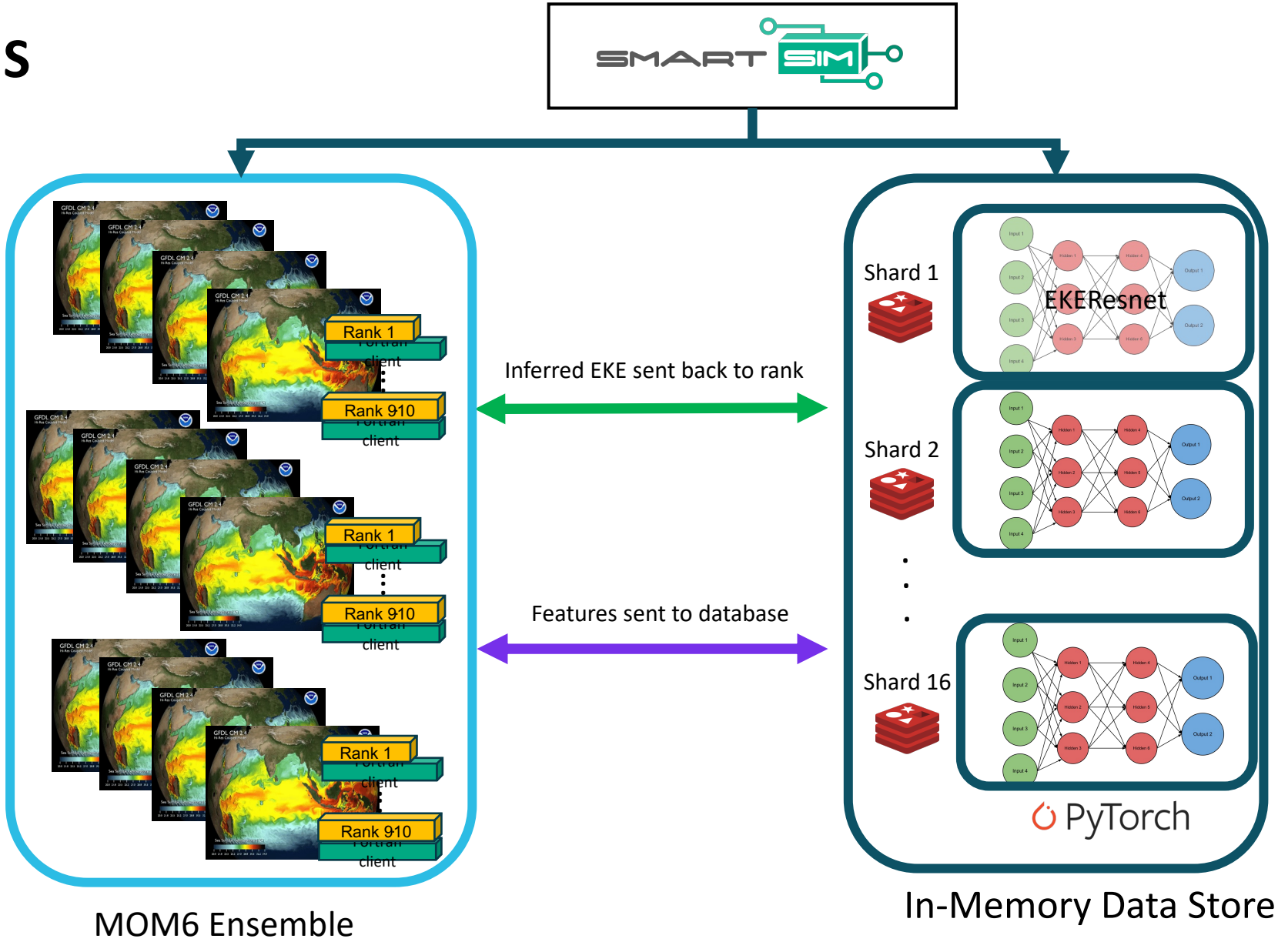
AI IN OCEAN SIMULATIONS

Goal: Improve representation of oceanic turbulence for quasi-LES simulations

Solution:

- Train machine-learning model to predict sub grid-scale turbulent kinetic energy
- Model 1: Use predictions within the model of eddy turbulence
  - 12 ensemble members
  - 10,920 CPUs, ~200 CPU-only nodes
  - 16 P100 GPUs

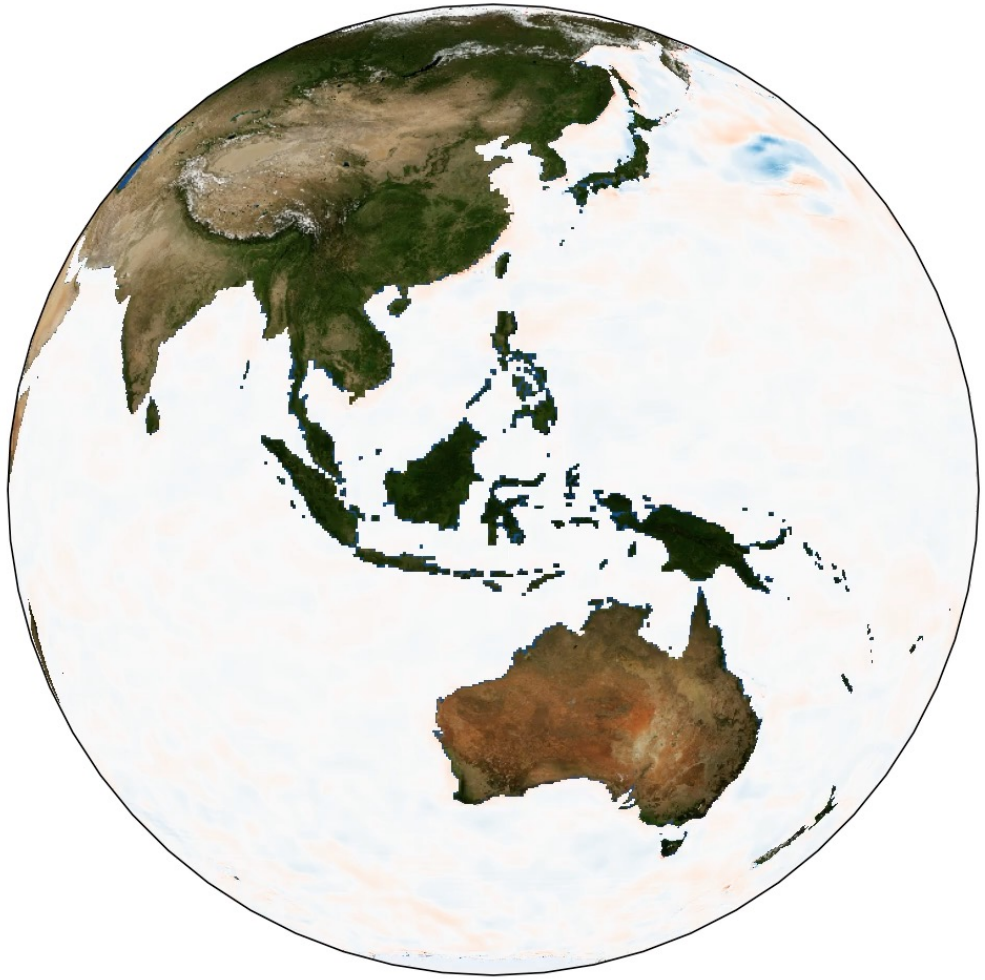
Milestone: First demonstration of AI embedded in the solver of a realistic ocean simulation



Collaboration between HPE/National Center for Atmospheric Research
Partee et al. [2022], doi.org/10.1016/j.jocs.2022.101707

DEPLOYING A SECOND TURBULENCE MODEL ON FRONTIER

- Prologue
  - Collaborators had a small, one-node ocean simulation with CNN parameterization
  - Successfully integrated SmartSim into MOM6
  - Cluster available to them did not have enough nodes to support a ‘realistic’ simulation
- June 12:
  - SmartSim engineers get access to allocation on Frontier
  - Install SmartSim from scratch
  - Run base case
- June 13:
  - Start doing scaling studies using SmartSim ensembles to understand how to efficiently use Frontier Resources
- June 14:
  - Run one year of realistic simulation on 50 nodes of Frontier
- June 15
  - Run application on 672 nodes of Frontier (~20,000 CPUs and 5,000 AMD GPUs)



RESOURCES AND CONCLUSIONS

DOCUMENTATION

www.craylabs.org

SmartSim 0.4.1 documentation

Search the docs ...

Versions

GETTING STARTED

Introduction
Installation
Community
Contributing Examples

TUTORIALS

Getting Started
Online Analysis
Online Inference
Online Training
Ray Integration

SMARTSIM

Experiments
Orchestrator
Launchers
SmartSim API

SMARTREDIS

SmartRedis
Integrating into a Simulation
Python
C++
Fortran
Data Structures
Runtime Requirements
SmartRedis API

Contents
Library Design

Introduction

SmartSim enables scientists to utilize machine learning inside traditional HPC workloads

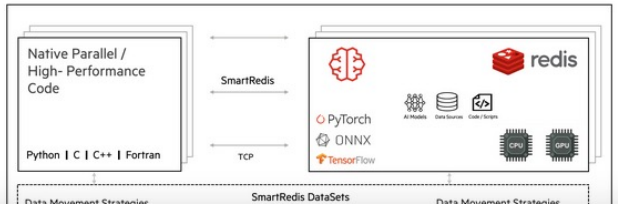
SmartSim provides this capability by

1. Automating the deployment of HPC workloads and distributed, in-memory storage (Redis).
2. Making TensorFlow, Pytorch, and ONNX callable from Fortran, C, and C++ simulations.
3. Providing flexible data communication and formats for hierarchical data, enabling online analysis, visualization, and processing of simulation data.

The main goal of SmartSim is to provide scientists a flexible, easy to use method for interacting at runtime with the data generated by simulation. The type of interaction is completely up to the user.

- Embed calls to machine learning models inside a simulation
- Create hooks to manually or programmatically steer a simulation
- Visualize the progression of a simulation integration from a Jupyter notebook

The figure below shows the architecture of SmartSim for a given use case. SmartSim can create, configure and launch workloads (called a `Model`), as well as groups of workloads (`Ensembles`). The data communication between a workload and in-memory storage is handled by the SmartRedis clients, available in Fortran, C, C++, and Python.



GETTING STARTED

Introduction
Installation
Community
Contributing Examples

TUTORIALS

Getting Started
Online Analysis
Online Inference
Online Training
Ray Integration

SMARTSIM

Experiments
Orchestrator
Launchers
SmartSim API

SMARTREDIS

SmartRedis
Integrating into a Simulation
Python
C++
Fortran
Data Structures
Runtime Requirements
SmartRedis API

REFERENCE

Changelog
Code of Conduct
Developer

Theme by the Executable Book Project

Contents

Experiment
Settings
Orchestrator
Model
Ensemble
Machine Learning
Slurm
Ray

SmartSim API

Experiment

`Experiment.__init__`(name[, exp_path, launcher])	Initialize an Experiment instance
`Experiment.start`(*args[, block, summary, ...])	Start passed instances using Experiment launcher
`Experiment.stop`(*args)	Stop specific instances launched by this `Experiment`
`Experiment.create_ensemble`(name[, params, ...])	Create an `Ensemble` of `Model` instances
`Experiment.create_model`(name, run_settings)	Create a general purpose `Model`
`Experiment.create_database`([port, db_nodes, ...])	Initialize an Orchestrator database
`Experiment.create_run_settings`(exe[, ...])	Create a `RunSettings` instance.
`Experiment.create_batch_settings`([nodes, ...])	Create a `BatchSettings` instance
`Experiment.generate`(*args[, tag, overwrite])	Generate the file structure for an `Experiment`
`Experiment.poll`([interval, verbose, ...])	Monitor jobs through logging to stdout.
`Experiment.finished`(entity)	Query if a job has completed.
`Experiment.get_status`(*args)	Query the status of launched instances
`Experiment.reconnect_orchestrator`(checkpoint)	Reconnect to a running `Orchestrator`
`Experiment.summary`([format])	Return a summary of the `Experiment`

class **Experiment**(*name*, *exp_path=None*, *launcher='local'*) [source]

Bases: `object`

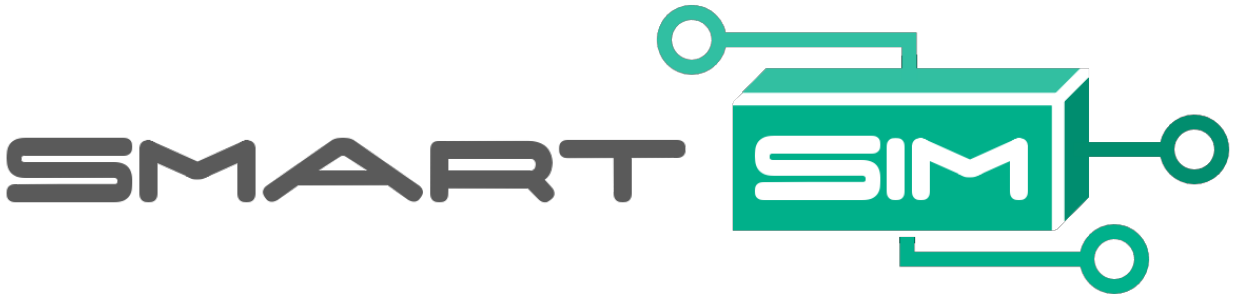
Experiments are the Python user interface for SmartSim.

Experiment is a factory class that creates stages of a workflow and manages their execution.

WHERE CAN I FIND MORE?

Other SmartSim Information

- Contact: CrayLabs@hpe.com
- SmartSim Repository: https://github.com/CrayLabs/SmartSi
  - Includes docker container with tutorials!
- SmartRedis Repository: https://github.com/CrayLabs/SmartRedis
- Seminar given at NCAR: https://www.youtube.com/watch?v=2e-5j427AS0
- SmartSim Case studies: https://github.com/CrayLabs/SmartSim-Zoo
- SmartSim Slack workspace (link on repo)



CLOSING WORDS

The value in combining AI and Simulations with SmartSim

- SmartSim can be used to create novel applications that involve both HPC and AI
- The lightweight SmartRedis client is easy to embed into simulation code
- Using a data-in-motion philosophy is key to creating complex AI/HPC workflows
- HPC and AI workflows have the potential to open up new branches of scientific discovery

Opportunities for Collaboration

Our team is open and eager to work with scientists across many domains

- Use simulations to generate data and symbolic regression for automatic equation discovery
- Building RL or blackbox optimizer to tune simulations for particular goals
- Improve accuracy of simulations by integrating ML models
- Adapt generative AI techniques for ML-assisted engineering design

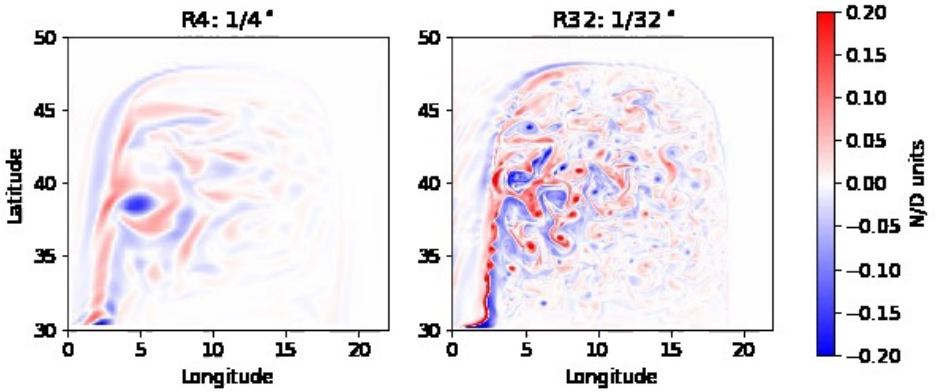
Questions, comments, other new ideas?

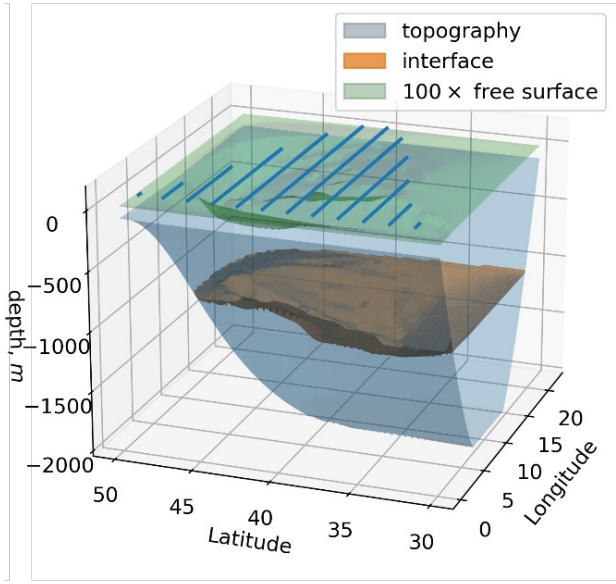
HANDS-ON WORKSHOP

RUNNING AN AI-ENABLED OCEAN SIMULATION

OVERVIEW OF THE WORKSHOP EXERCISE

- By the end of the workshop:
  - Understand how to instrument a simulation with SmartRedis
  - Write a SmartSim driver script to configure and launch a simulation
  - Be aware of Frontier-specific best practices
- Exercise
  - Run a toy version of the ocean example
    – Idealized North Pacific (‘double gyre’)
  - Components:
    – SmartSim modulefile
    – MOM6 ocean executable
    – Pre-trained ML model
    – SmartSim driver
  - Phase 1: Overview of implementation
  - Phase 2: Run the example
  - Phase 3: Scale the simulation
- Resources
  - https://github.com/CrayLabs/OLCF_SmartSim2023.git
  - Ask SmartSim Team Members in the Slack Channel for help





Courtesy of Cheng Zhang [Princeton]

SMARTREDIS IMPLEMENTATION IN MOM6

- MOM6 is written in pure Fortran

```
input(1) = "input"//CS%key_suffix
model_input(1) = "model_input"//CS%key_suffix
model_output(1) = "model_output"//CS%key_suffix
output(1) = "output"//CS%key_suffix

! script 1
db_return_code = CS%client%run_script(CS%script_key, "pre_process", input, model_input)
if (CS%client%SR_error_parser(db_return_code)) call MOM_error(FATAL, "Run script1 in the database failed")

! ML model
call cpu_clock_begin(CS%id_run_model)
db_return_code = CS%client%run_model(CS%model_key, model_input, model_output)
if (CS%client%SR_error_parser(db_return_code)) call MOM_error(FATAL, "Run model in the database failed")
call cpu_clock_end(CS%id_run_model)

! script 2
call cpu_clock_begin(CS%id_run_script2)
db_return_code = CS%client%run_script(CS%script_key, "post_process", model_output, output)
if (CS%client%SR_error_parser(db_return_code)) call MOM_error(FATAL, "Run script2 in the database failed")
call cpu_clock_end(CS%id_run_script2)

! extract the output from Python
call cpu_clock_begin(CS%id_unpack_tensor)
db_return_code = CS%client%unpack_tensor(output(1), out_for, shape(out_for))
if (CS%client%SR_error_parser(db_return_code)) call MOM_error(FATAL, "unpack tensor from the database failed")
call cpu_clock_end(CS%id_unpack_tensor)
```

THE MOM6 SMARTSIM DRIVER

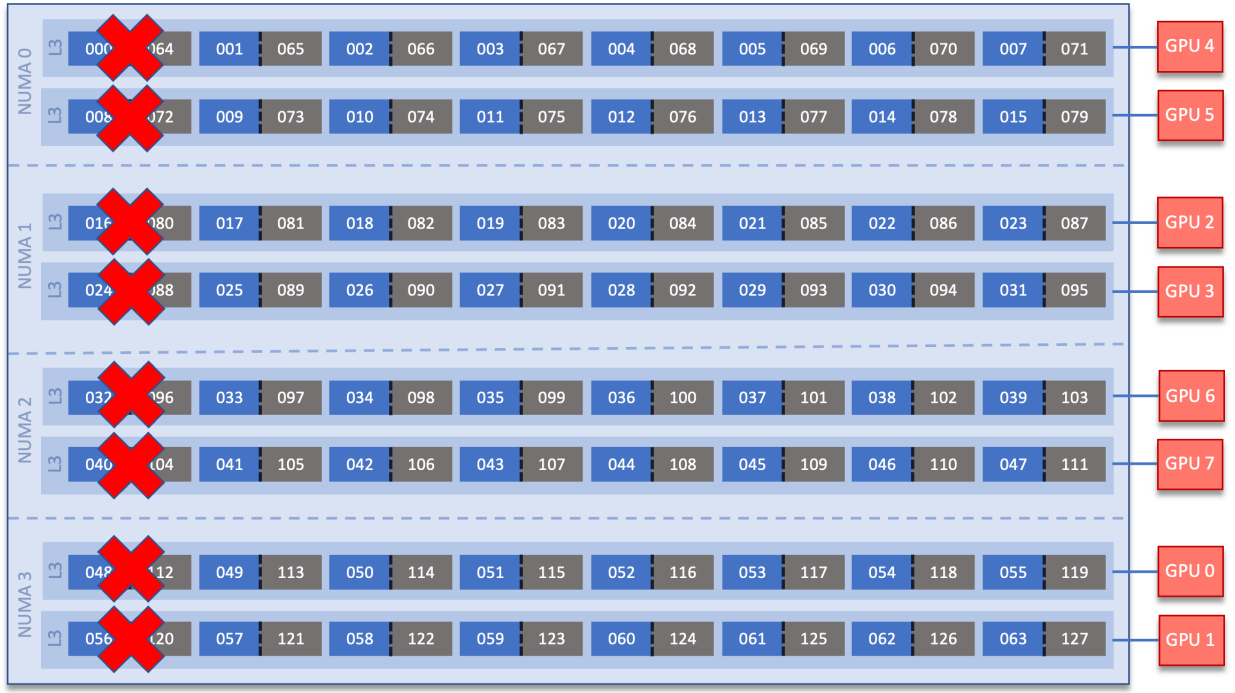
- Link to driver: https://github.com/CrayLabs/OLCF_SmartSim2023/blob/main/smartsim_drivers/double_gyre/call_MOM6.py

- Main steps:
  - Create a SmartSim Experiment
  - Set the resources needed to launch and run MOM6
  - Attach and configure a MOM6 simulation
  - Configure a ‘colocated’ database to run alongside the simulation
  - Add ML model and pre/post-processing scripts
  - Launch the model

- Variation 1:
  - Scale the number of CPUs used by the database and the simulation
- Variation 2:
  - Enable CPU ‘pinning’ according to Frontier’s reserved system CPUs

THE IMPORTANCE OF PINNING ON FRONTIER

- Colocated database and simulation code are slightly asynchronous
  - Simulation code waits for ML inference to complete
  - Pinning prevents processes from changing cores while waiting, reducing overall performance
- Frontier reserves cores for system processes
  - Pinning those CPUs is not allowed
- SmartSim provides custom pinning for collocated database, Slurm allows cpu pinning of the application
- Rerun:
  - `python call_MOM6.py --enable-pinning`



CLOSING AND TAKEAWAYS

- Use this SmartSim driver as a starting point for your own simulations
- Other examples can be found in the tutorials: https://github.com/CrayLabs/SmartSim/pkgs/container/smartsim-tutorials
- Please contact us at our Slack channel: https://join.slack.com/t/craylabs/shared_invite/zt-1z9cmf1yp-PrzmFadoofJ0hp5XHtDcMQ