LUMI AI Course:

Coupling machine learning with HPC simulation

Harvey Richardson & Alessandro Rigazzi, HPE

November 26–27, 2024

Hewlett Packard Enterprise

Agenda

- Where might machine learning play a role in simulation?
- Workflows for coupling HPC simulation with AI
- Challenges and approaches
- SmartSim
- SmartSim Examples

Where might machine learning play a role with Simulation

Completely replace a simulation
- AI model learns to produce output by observing simulation inputs/outputs

Use simulation as one input to a machine learning model
- For example use an ML model to account for location-specific history (weather)

Replace modeling of physical processes or parameterised models with machine learning, Examples..
- Reduce search space for Drug discovery
- a turbulence model in an ocean simulation
- Particle physics: particle tracks
- Atomic potentials (computational chemistry)

- We will concentrate on the case where AI is coupled with simulation

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
  - Skillful models can be considered lower-order **representations of 'true' simulation**
    - Useful for exploring parameter space/uncertainties
- Downsides of AI models
  - How do you add process complexity?
  - Can they extrapolate beyond the data they have been trained on?
- 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?
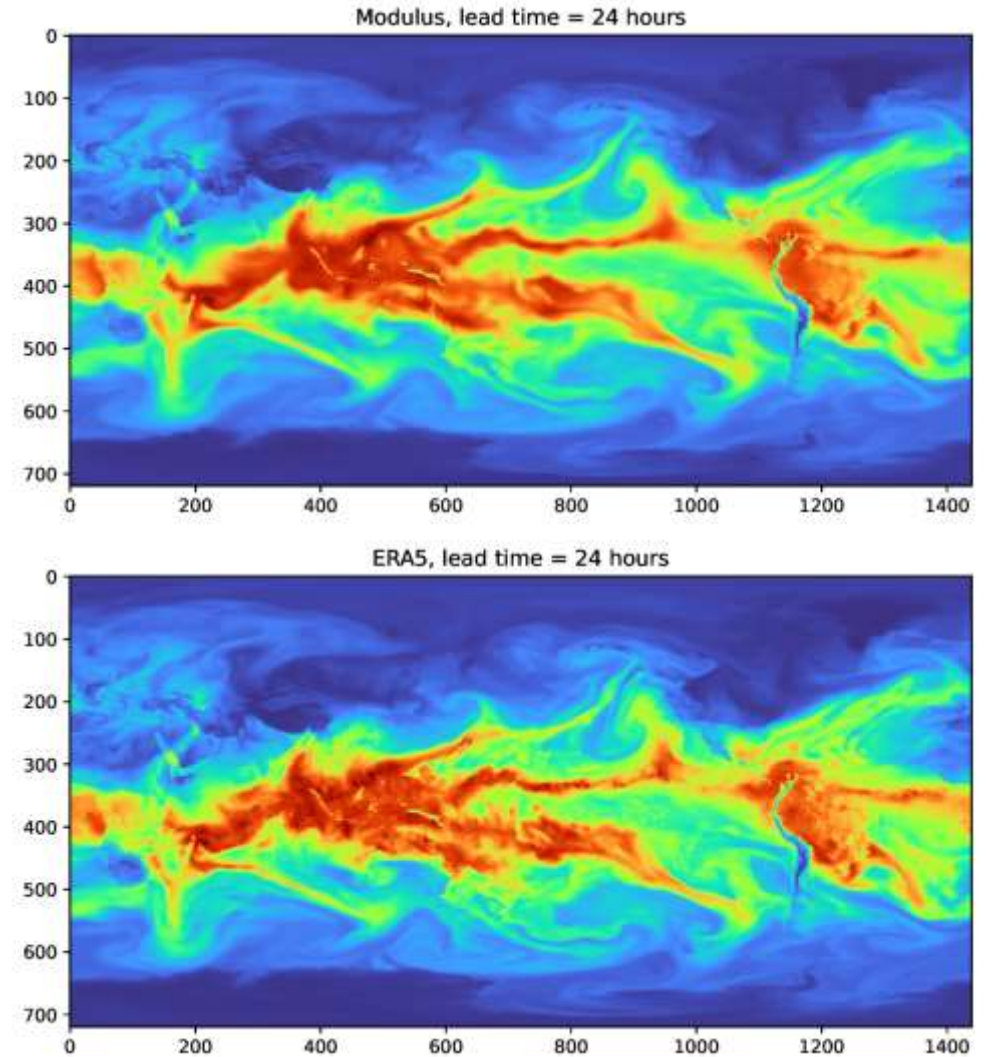
https://docs.nvidia.com/deeplearning/modulus/modulus-sym/user_guide/neural_operators/fourcastnet.html#introduction

HPC Applications combined with AI software drive innovation

Combining AI software and traditional HPC applications at different levels of a workflow unlocks innovative solutions

ML around-the-loop

- Automatic parameter tuning
- New data assimilation techniques

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)

Edge AI:
Cross-facility,
event triggered,
data-driven

ML around-
the-loop:
Inference or
training after
simulation

ML on-the-loop:
Inference and
training every
1k-10k time steps

ML in-the-loop:
Inference every
time step &
training online with
model updates

Physics
Simulation

ML outside-the-loop:
Intelligent sampling

Challenges and approaches

- Machine learning frameworks are invariably accessed via Python
- HPC simulation is most likely written in C/C++/Fortran
- **We could implement ML in our simulation language but...**
  - A lot of work for something likely already done and likely more efficiently than you will
  - **We don't get access to tools to train models**
  - Hard to integrate a model externally developed

Some approaches
- Use language-interoperability to interface between simulation and Machine Learning
- Couple ML components to our simulation (sockets, messaging transports, files)
- Use a framework designed to provide such interoperability (via network transport or APIs)
  - Fortran Keras Bridge
  - SmartSim (originally from Cray)

Language Interoperability

Interoperability by calling conventions
- Fortran and Python
  - f2py and fmodpy or forpy can help build wrappers to call Fortran from Python
  - ISO C bindings on the Fortran side interfaced to ctypes/Cython on the python side
  - Really helpful if what you are interfacing to has direct support for the Numpy C API
- C++/C and Python
  - Cython, pybind11, SWIG

Interoperability at Framework Level (Fortran)
- Directly call Tensorflow or Torch APIs from Fortran using ISO C interoperability
  In both cases you may have to save model in a special format

Alternatively
- Communicate workflow components via filesystem or network

About SmartSim

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

- Provides a loose-coupling philosophy for combining HPC & AI

SmartSim enables simulations to be used as engines within a system, producing data, consumed by other services to 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*
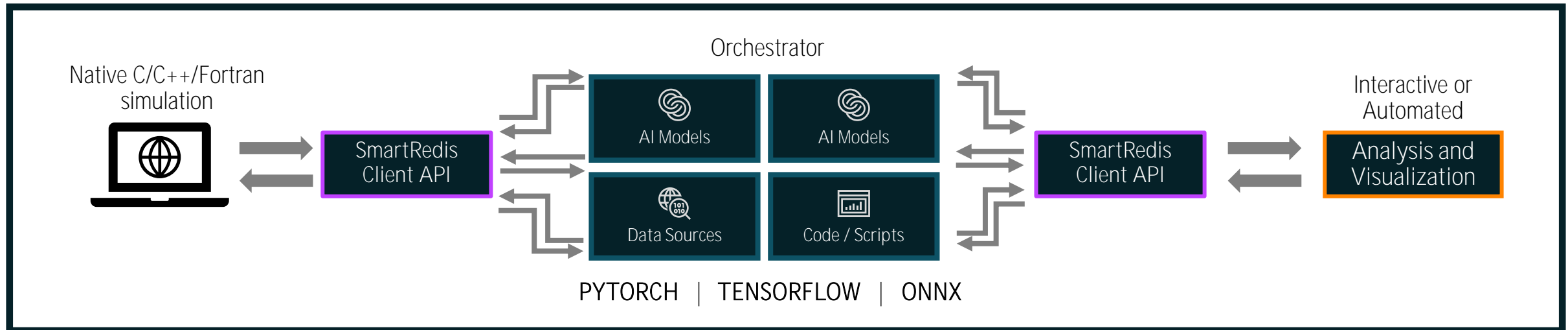
Creating a SmartSim experiment

Integration steps

1. Embed SmartRedis calls (C/C++/Fortran) into the application (~10 lines of code)
2. Write a driver script using the SmartSim Python library to describe and launch the workflow

Driver script can check status of components

Added simulation code

```
client = smartredis_CS%client

...

sr_return_code = client%put_tensor("features"//CS%key_suffix, CS%features_array, shape(CS%features_array))
model_out(1) = "EKE"//CS%key_suffix
model_in(1) = "features"//CS%key_suffix
sr_return_code = smartredis_CS%client%run_model(CS%model_key, model_in, model_out)

sr_return_code = client%unpack_tensor( model_out(1), CS%MEKE_vec, shape(CS%MEKE_vec) )

...
```

SmartSim Driver Script

```python
# Create experiment

experiment = Experiment("AI-EKE-MOM6", launcher="auto")

# Create ensemble

ensemble_batch_settings = experiment.create_batch_settings(
    nodes      = ensemble_size*nodes_per_member,
    time       = walltime,
    batch_args = mom6_batch_args
)

mom6_run_settings = experiment.create_run_settings(mom6_exe_path)
mom6_run_settings.set_tasks_per_node(tasks_per_node)
mom6_run_settings.set_tasks(nodes_per_member*tasks_per_node)

mom_ensemble = experiment.create_ensemble(
    "MOM",
    batch_settings = ensemble_batch_settings,
    run_settings   = mom6_run_settings,
    replicas       = ensemble_size
)

mom_ensemble.attach_generator_files(
    to_configure=glob("../MOM6_config/configurable_files/*"),
    to_copy="../MOM6_config/OM4_025",
    to_symlink="../MOM6_config/INPUT"
)
```

```python
# Configure ensembles

MOM6_config_options = {
    "SIM_DAYS": 15, # length of simulations
    "EKE_MODEL": eke_model_name,
    "EKE_BACKEND": eke_backend,
    "DOMAIN_LAYOUT": domain_layout,
    "MASKTABLE": mask_table
}

MOM6_config_options.update( {
    "SMARTREDIS_COLOCATED":"False",
    "SMARTREDIS_COLOCATED_STRIDE":10,
    "SMARTREDIS_CLUSTER": "False"
})

MOM6_config_options.update( {'SMARTREDIS_CLUSTER':'True'} )

for model in ensemble:
    model.params = MOM6_config_options
    model.register_incoming_entity(model)

# Create in-memory database

orchestrator = exp.create_database(
    port = orchestrator_port,
    interface = orchestrator_interface,
    db_nodes = orchestrator_nodes,
    time=walltime,
    threads_per_queue=2,
    batch=True)

orchestrator.set_cpus(18)
orchestrator.set_batch_arg("constraint", orchestrator_node_features)
orchestrator.set_batch_arg("exclusive",None)

experiment.generate( mom_ensemble, orchestrator, overwrite=True )

experiment.start(mom_ensemble, orchestrator, block=True, summary=True)
```
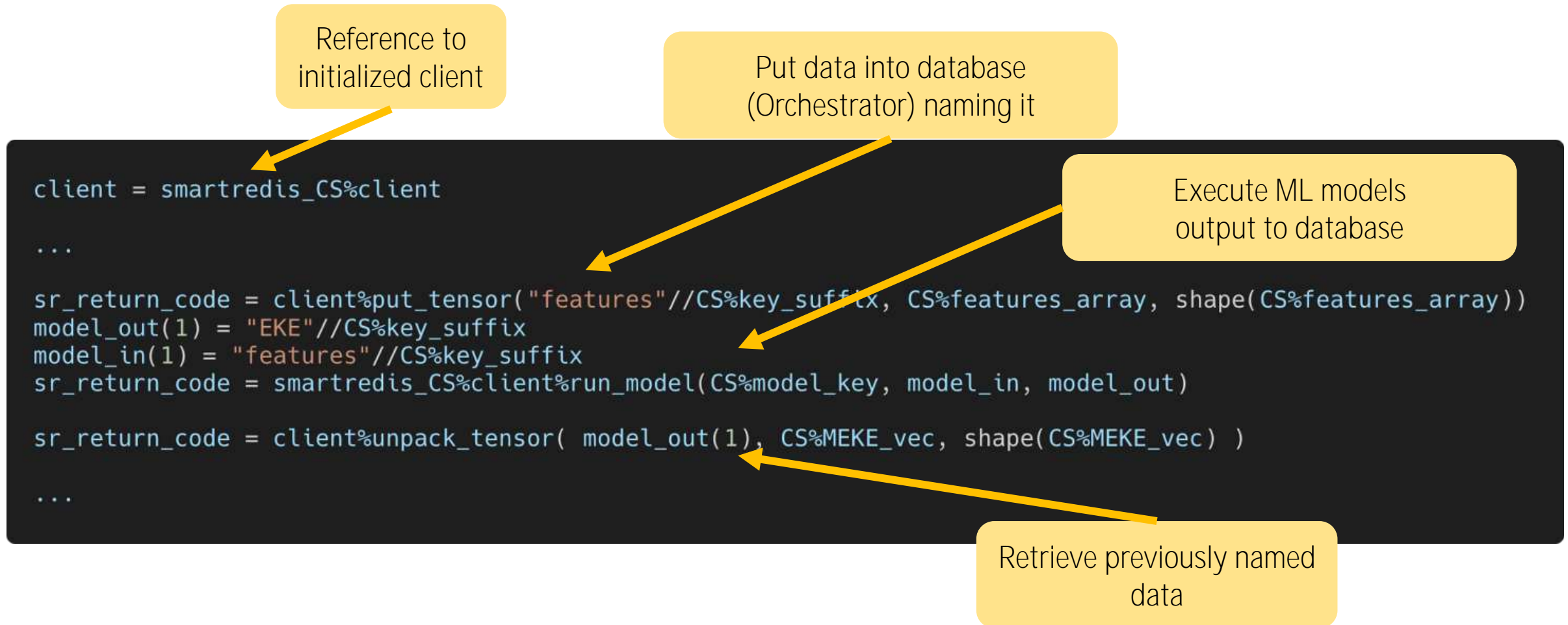
Added client simulation code

Reference to initialized client

Put data into database (Orchestrator) naming it

Execute ML models output to database

Retrieve previously named data

```
client = smartredis_CS%client

...

sr_return_code = client%put_tensor("features"//CS%key_suffix, CS%features_array, shape(CS%features_array))
model_out(1) = "EKE"//CS%key_suffix
model_in(1) = "features"//CS%key_suffix
sr_return_code = smartredis_CS%client%run_model(CS%model_key, model_in, model_out)

sr_return_code = client%unpack_tensor( model_out(1), CS%MEKE_vec, shape(CS%MEKE_vec) )

...
```

SmartSim driver script (create ensemble)

Driver Script
    Describes, launches and manages workflow with applications and ML infrastructure

Experiment
    Top level object that provides factory methods to create workflow objects

Batch Settings
    Can be used if application is to be launched non-interactively, including as ensembles

RunSettings object
    Describes system-specific resources (nodes, accelerators, cpus etc.

Model object
    Holds information on user application

Application file handling
    Can be parameterized (also args)

```python
# Create experiment

experiment = Experiment("AI-EKE-MOM6", launcher="auto")


# Create ensemble

ensemble_batch_settings = experiment.create_batch_settings(
    nodes       = ensemble_size*nodes_per_member,
    time        = walltime,
    batch_args  = mom6_batch_args
)


mom6_run_settings = experiment.create_run_settings(mom6_exe_path)
mom6_run_settings.set_tasks_per_node(tasks_per_node)
mom6_run_settings.set_tasks(nodes_per_member*tasks_per_node)

mom_ensemble = experiment.create_ensemble(
    "MOM",
    batch_settings = ensemble_batch_settings,
    run_settings   = mom6_run_settings,
    replicas       = ensemble_size
)


mom_ensemble.attach_generator_files(
    to_configure=glob("../MOM6_config/configurable_files/*"),
    to_copy="../MOM6_config/OM4_025",
    to_symlink="../MOM6_config/INPUT"
)
```

SmartSim **driver script... (configure ensemble members)**

```python
# Configure ensembles

MOM6_config_options = {
    "SIM_DAYS": 15, # length of simlations
    "EKE_MODEL": eke_model_name,
    "EKE_BACKEND": eke_backend,
    "DOMAIN_LAYOUT": domain_layout,
    "MASKTABLE": mask_table
}

MOM6_config_options.update( {
    "SMARTREDIS_COLOCATED":"False",
    "SMARTREDIS_COLOCATED_STRIDE":0,
    "SMARTREDIS_CLUSTER": "False"
})


MOM6_config_options.update( {'SMARTREDIS_CLUSTER':'True'} )

for model in ensemble:
    model.params = MOM6_config_options
    model.register_incoming_entity(model)
```

Ensemble members have identical parameters

SmartSim **driver script... (configure and start ensemble members)**

```python
# Create in-memory database

orchestrator = exp.create_database(
    port = orchestrator_port,
    interface = orchestrator_interface,
    db_nodes = orchestrator_nodes,
    time=walltime,
    threads_per_queue=2,
    batch=True )

orchestrator.set_cpus(18)
orchestrator.set_batch_arg("constraint", orchestrator_node_features)
orchestrator.set_batch_arg("exclusive",None)

experiment.generate( mom_ensemble, orchestrator, overwrite=True )

experiment.start(mom_ensemble, orchestrator, block=True, summary=True)
```

Setup Orchestrator (stores ML models/tensors and executes models

Write all files needed for workflow entities

GO

Online Inference, multiple languages

Fortran

```fortran
call client%put_tensor(in_key, array, shape(array))

inputs(1) = in_key
outputs(1) = out_key
call client%run_model(model_name, inputs, outputs)
result(:,:) = 0.
call client%unpack_tensor(out_key, result, shape(result))
```

C++

```cpp
// Put the image tensor on the database
client.put_tensor(in_key, img.data(), {1,1,28,28},
                  SmartRedis::TensorType::flt,
                  SmartRedis::MemoryLayout::contiguous);

// Run model already in the database
client.run_model(model_name, {in_key}, {out_key});

// Get the result of the model
std::vector<float> result(1*10);
client.unpack_tensor(out_key, result.data(), {10},
                     SmartRedis::TensorType::flt,
                     SmartRedis::MemoryLayout::contiguous);
```

Python

```python
client.put_tensor("input", torch.rand(20, 1, 28, 28).numpy())

# put the PyTorch CNN in the database in GPU memory
client.set_model("cnn", net, "TORCH", device="GPU")

# execute the model, supports a variable number of inputs and outputs
client.run_model("cnn", inputs=["input"], outputs=["output"])

# get the output
output = client.get_tensor("output")
print(f"Prediction: {output}")
```
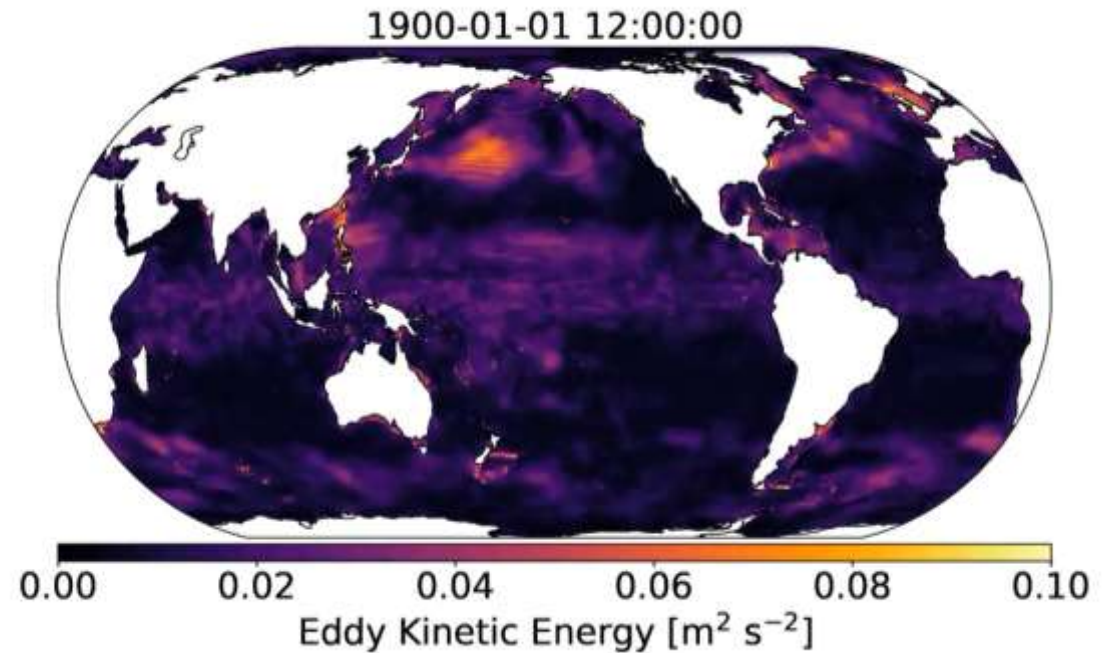
EXAMPLES OF USING AI IN-THE-LOOP IN THE MOM6 OCEAN MODEL

Improving MOM6's eddy kinetic energy

Original MEKE Scheme (Jansen et al. [2015]):
- Integrate a prognostic eddy kinetic energy equation with parameterized sources/sinks
- Use length-scale relations to convert EKE to Gent-McWilliams, Redi, and viscosity coefficients

Known shortcomings
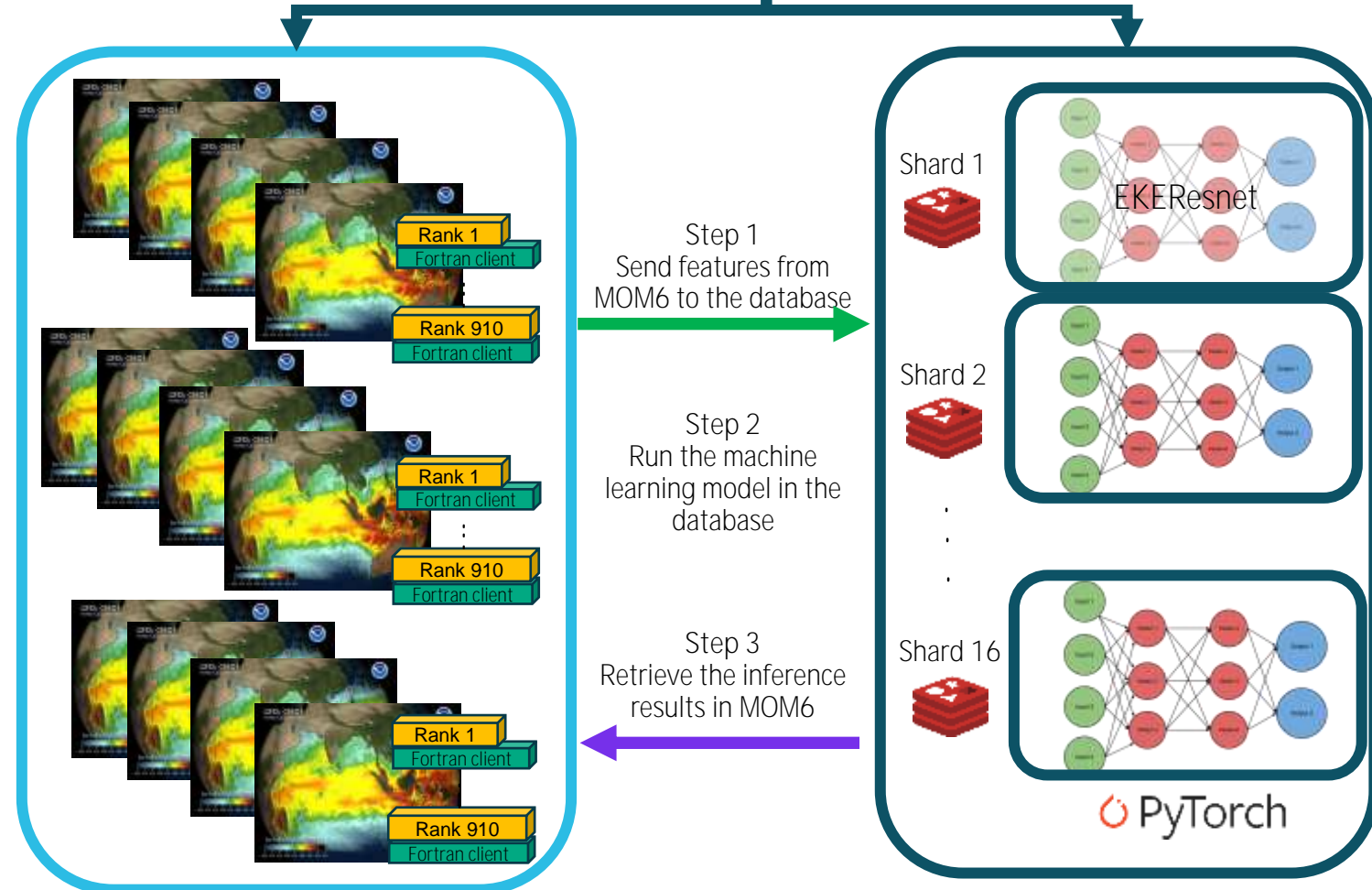- EKE equation has terms which are tunable and/or have errors which may be first-order

Propose ML-based solution
- Use an eddy-resolving simulation to train a neural network to learn the relationship between large-scale quantities and eddy kinetic energy
- Embed neural network predictions in eddy-permitting simulations

SmartSim IL launching MOM6 and Orchestrator

MOM6 sends input features

EKEResNet predicts and returns turbulent kinetic energy values

Step 1
Send features from MOM6 to the database

Step 2
Run the machine learning model in the database

Step 3
Retrieve the inference results in MOM6

EKEResnet

PyTorch

MOM6 Ensemble

In-Memory Data Store

"Orchestrator"

Using SmartSim in mom6 for turbulence modelling [NCAR+HPE]

- Experiment setup
  - 12 ensemble members
  - 10,920 CPUs (~200 nodes) and 16 P100s (16 nodes)
  - Inference embedded at the tracer timestep (3hr)
    - 970 billion inferences over 10 simulation years
    - 1.6 million inferences per second

- Key results:
  - Offloading ML inference to dedicated nodes improves GPU utilization while incurring small communication cost
  - With SmartSim, the accuracy over the current state of the art improved by over 20%
  - Neural network more accurately predicts (20% RMSE improvement) rather than the prognostic approach
  - Overall performance only decreased by 10% while only minimally increasing hardware footprint of the model

Partee et al. [2022]: Using Machine Learning at scale in numerical simulations with SmartSim: An application to ocean climate modeling
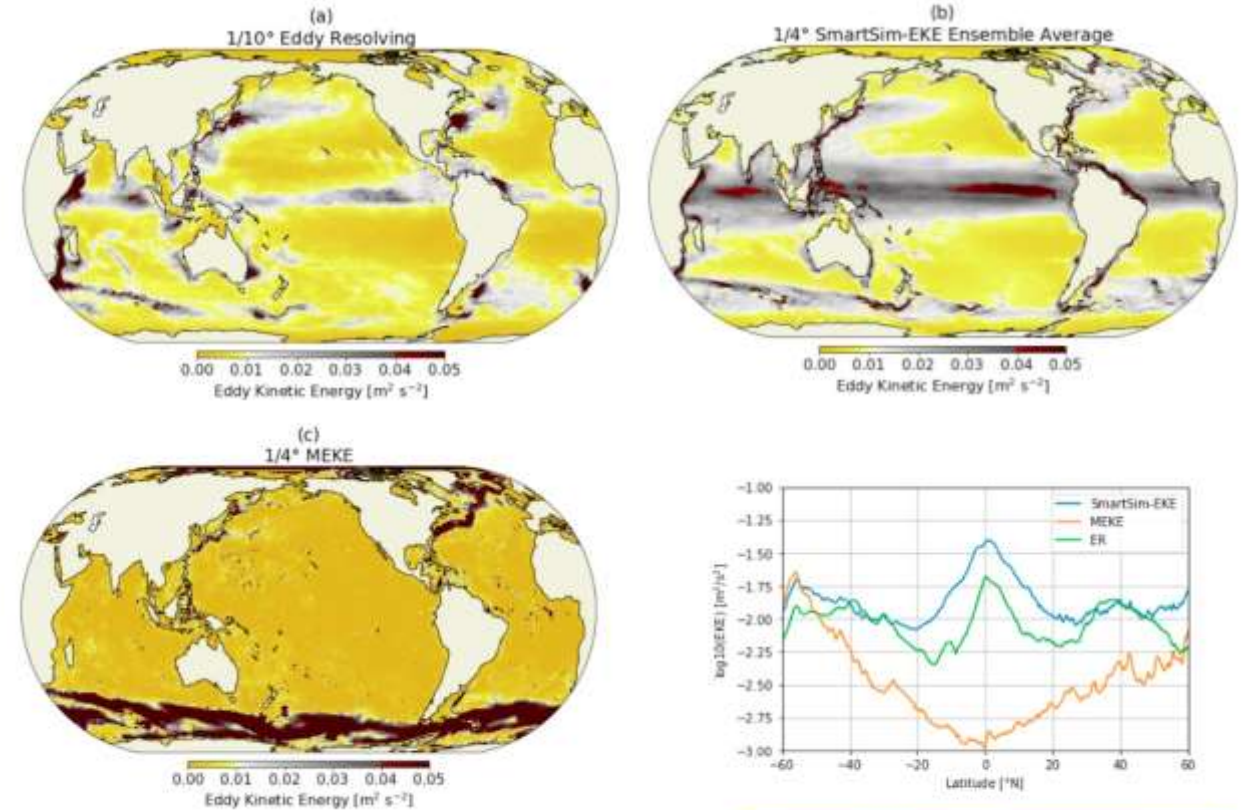

Fig. 7. Zonally averaged EKE (on a log10 scale) as a function of latitude from the ER, SmartSim-EKE, and MEKE.

EXAMPLES OF USING AI AROUND-THE-LOOP

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"**
  - Action: NN can control actuators upwind of bubble
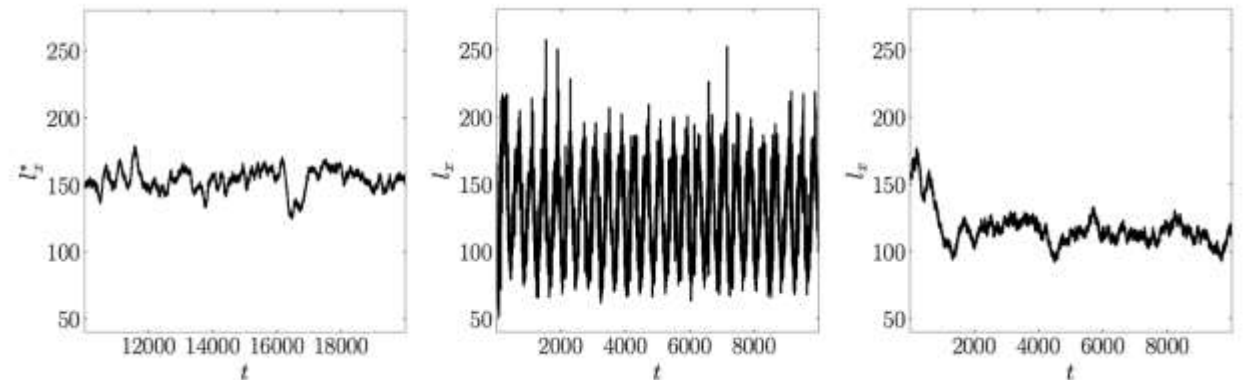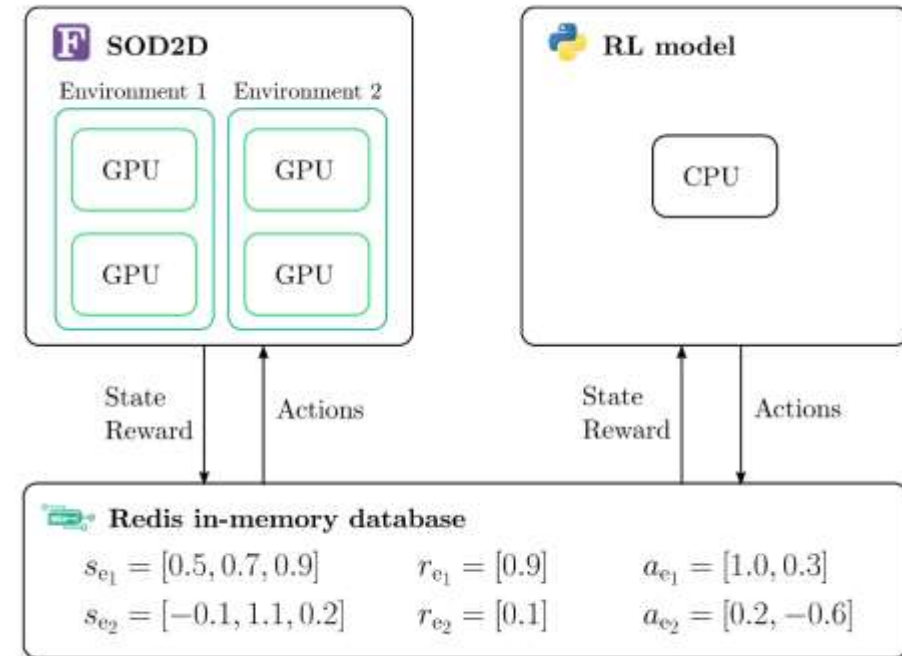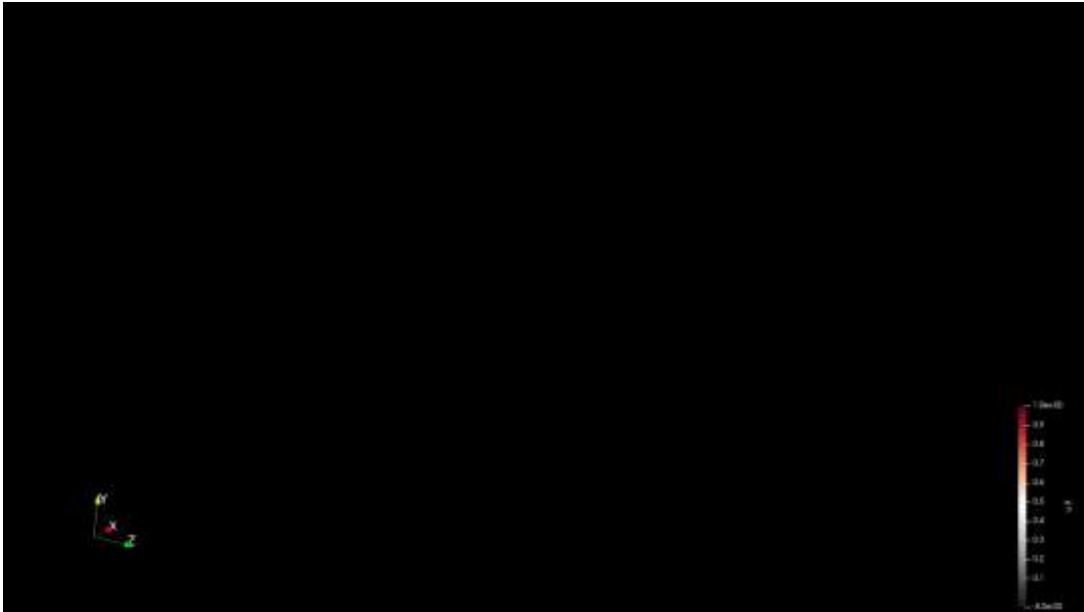
# Dynamic turbine wake steering in wind farms

- Reference: Project is work in progress, Andrew Moat, Imperial College London
- Turbine wakes can reduce efficiency of downstream turbines
- We can steer the wake by adjusting angle (yaw) of turbines
- In general, the system is very complex, changing direction/speed, multiple turbines, interaction of wakes etc.
- Use reinforcement learning to find some optimal set of angles to optimize power output
- Direct coupling with SmartSim an improvement over file-based coupling

Reinforcement learning

- Agent interacts with an environment to optimize some quantity

Change in yaw angles

Agent

State s_t
Reward r_t

Action a_t

Environment

Total power or wind farm
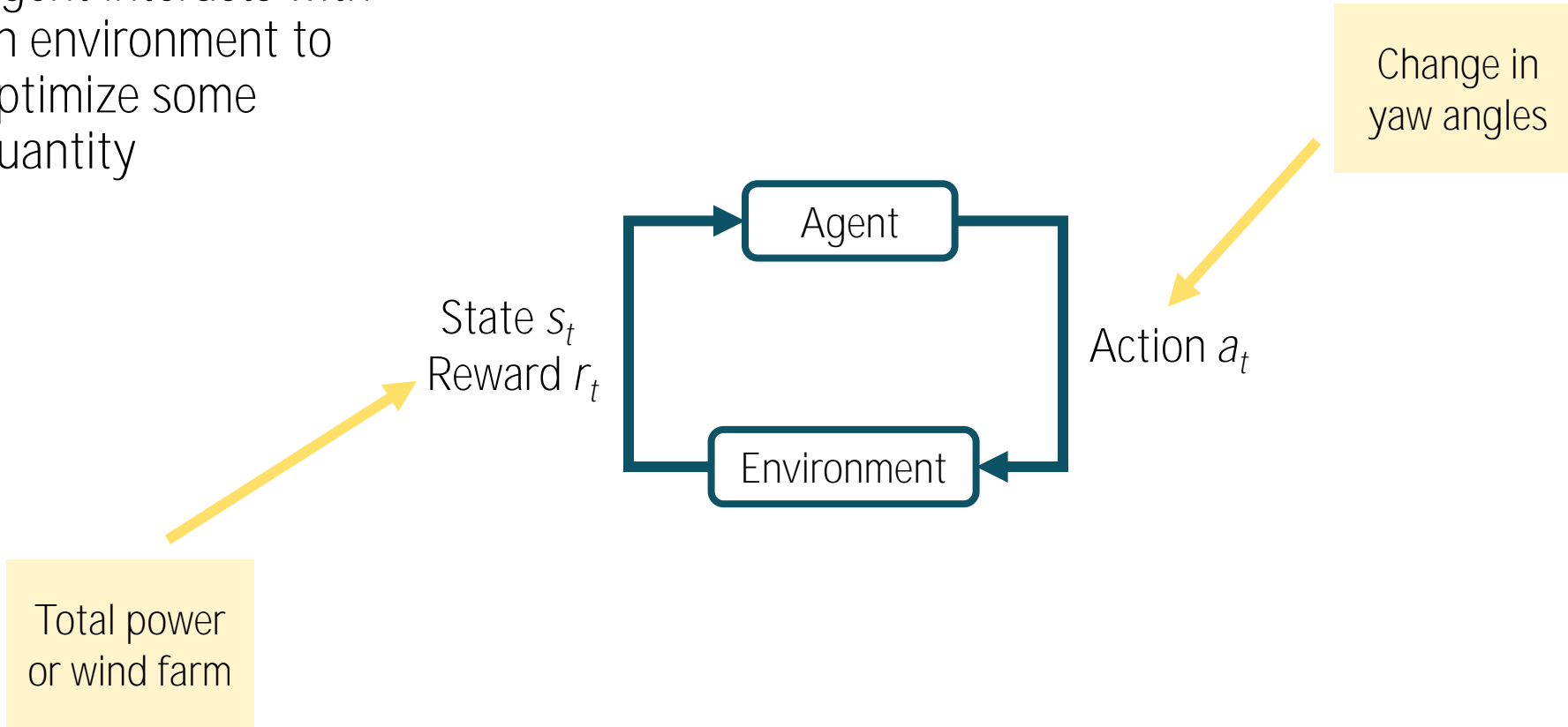
Implementation

- Initial approach was to couple the Large Eddy simulation of the Windfarm with pytorch reinforcement learning model
- Used file storage to communicate (a_t s_t r_t)
- Issues:
  - Large files often written was costly
  - Was not going to be practical when simulating larger wind farms
  - Made it difficult to train multiple environments at a time
- Moved to SmartSim
  - Now using Smart Redis database
- Modifications needed
  - XCompact3D Replace file ops with calls to send and retrieve data to smart-redis
  - Configured SmartSim to start XCompact3D environments/simulations (32 at a time for example)
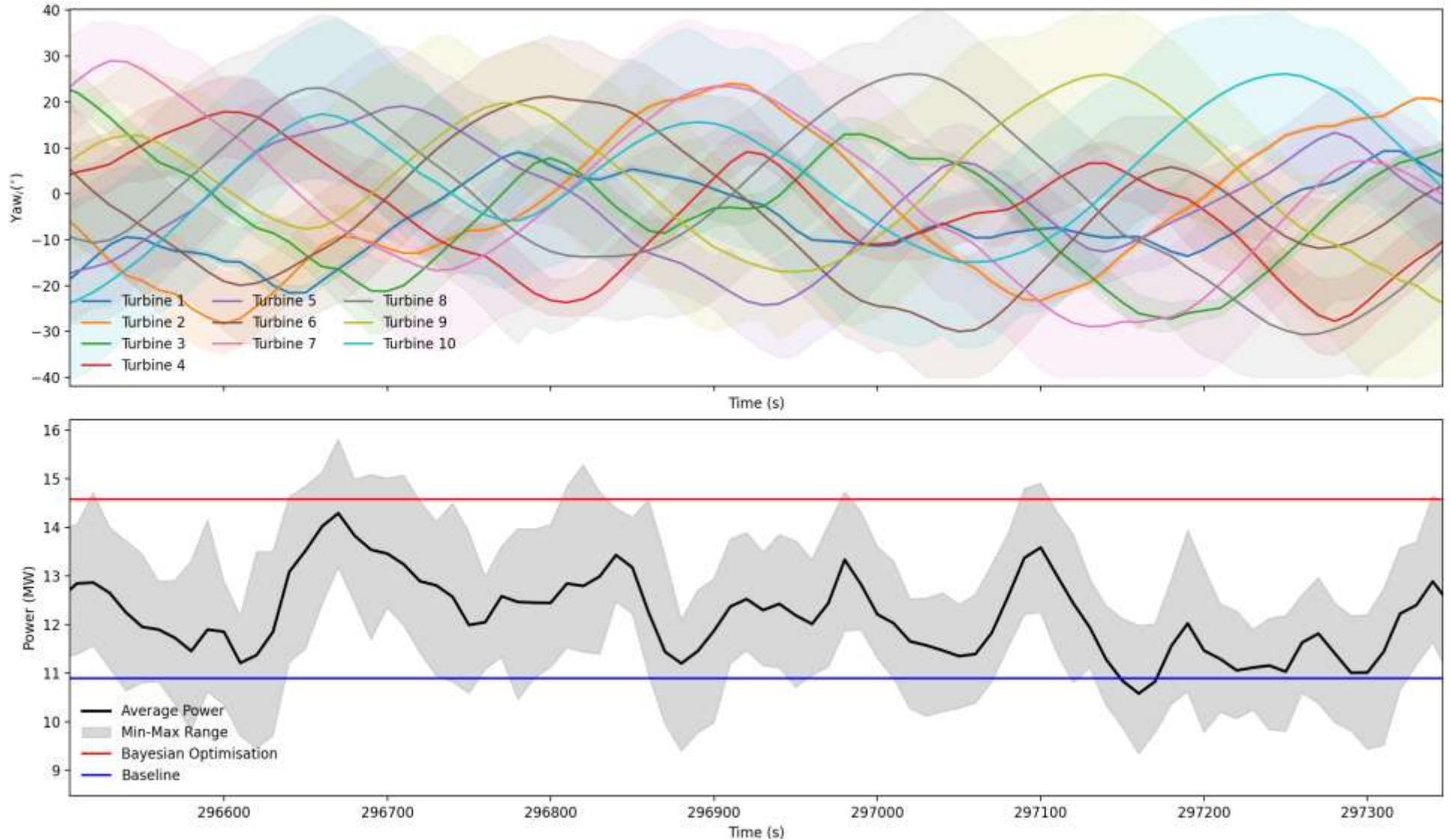
Experiment

- Setup
  - 32 Environments each with
  - 10 turbines
  - 128 cores per environment (a node)
  - 50 simulation timesteps per RL step
  - Batched into 512 frames for training

Initial Results

- After initial training the turbines settle down into being driven in sinusoidal pattern
- Better power output than without steering.

Other Examples of SmartSim use

Computational Fluid Dynamics

- OpenFOAM, FLEXI, PHASTA, libCEED, NekRS (in progress)

Climate and Weather

- MOM6, NEMO, CESM

Molecular Dynamics

- LAMMPS, OPENMM

In-situ Visualization workflow

References

- Language Interoperability
  - F2py and fmodpy intro https://www.matecdev.com/posts/fortran-in-python.html
  - forpy (https://github.com/ylikx/forpy)
  - pybind11 (https://github.com/pybind/pybind11)
  - Talk: Reducing the overhead of coupling machine learning models between Python and Fortran, https://www.youtube.com/watch?v=Ei6H_BoQ7g4 https://jackatkinson.net/slides/RSECon23/RSECon23.html#/title-slide

- Interoperability at framework level:
  - Fortran Keras Bridge, tensorflow but not very active
    - https://github.com/scientific-computing/FKB
    - https://arxiv.org/abs/2004.10652

- SmartSim
  - https://github.com/CrayLabs/SmartSim
  - https://github.com/CrayLabs/SmartSim-Zoo
  - ARCHER2 Webinar: Exploring new computational frontiers with SmartSim

Questions?