

Coupling Simulations with AI

Riccardo Balin and Bethany Lusch

With help from Christine Simpson

Argonne National Laboratory, LCF

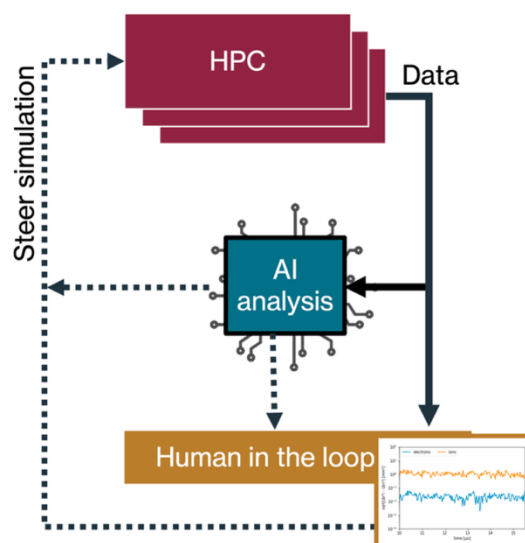
AI4Science Series:
Advanced Topics in AI for Science
October 28, 2025

Why Couple HPC Simulations with AI/ML?

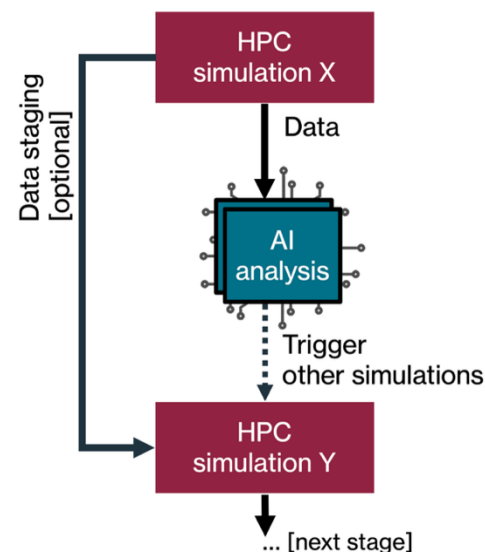
- ❑ Substitute inaccurate or expensive components of simulation with ML models
 - E.g.: Closure or surrogate modeling
- ❑ Optimize simulation parameters on-the-fly
 - E.g.: Select solver parameters at runtime based on AI inference
- ❑ Avoid IO bottleneck and disk storage issues during offline training
 - E.g.: In situ/online training through data streaming or in-memory staging
- ❑ Active learning and model fine-tuning
 - E.g.: Continuous fine-tuning and deployment of model
 - E.g.: Access training data not available during offline pre-training
- ❑ Steering of simulation ensembles
 - E.g.: Design space exploration or parameter optimization guided by AI

How to Couple HPC Simulations and AI/ML?

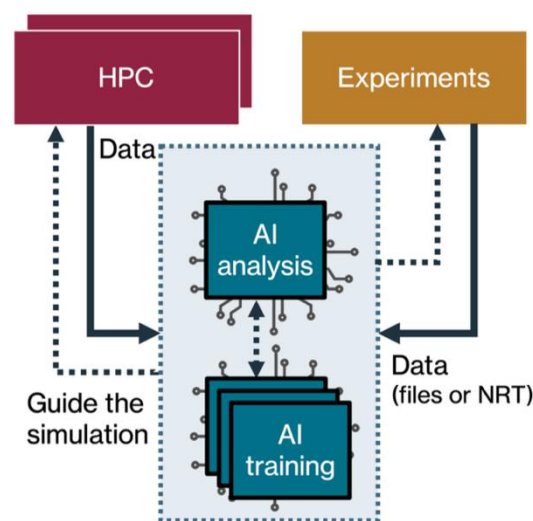
Steering of Ensembles



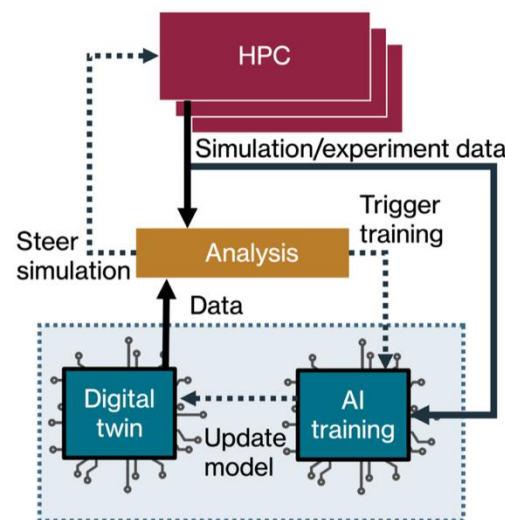
Optimize Simulation Parameters



Active Learning/
Online Fine-Tuning



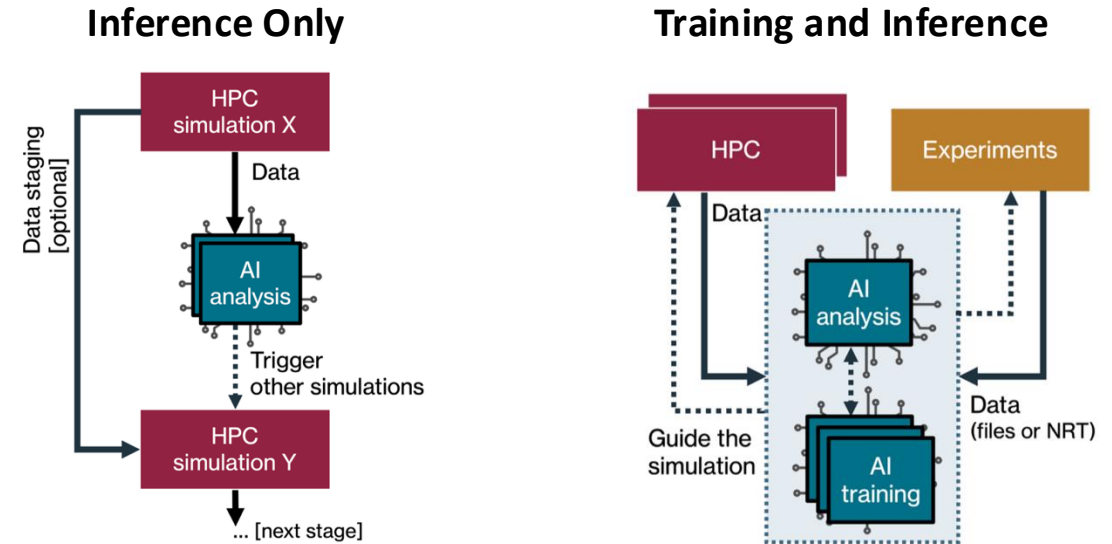
Digital Twin



How to Couple HPC Simulations and AI/ML?

Type of Coupling

- ☐ ML inference only
- ☐ ML training only
- ☐ Both training and inference



Programming Languages and Programming Models

- ☐ Simulation and ML components often written in different languages
- ☐ AI/ML relies heavily on vendor libraries (stuck with vendor language or programming model)
- ☐ Separate or shared parallelism strategies
 - Shared vs. separate MPI communicators
 - Domain decomposition vs. batch or model parallelism

How to Couple HPC Simulations and AI/ML?

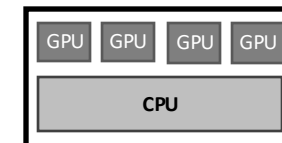
Execution Management

❑ Time division (tight coupling)

- Components run on same compute resources (may even use same processes)
- Staggered in time, execution of one component halts the other
- May allow for direct memory access and no data copy/transfer
- Idle time of individual components may be significant

❑ Space division (loose coupling)

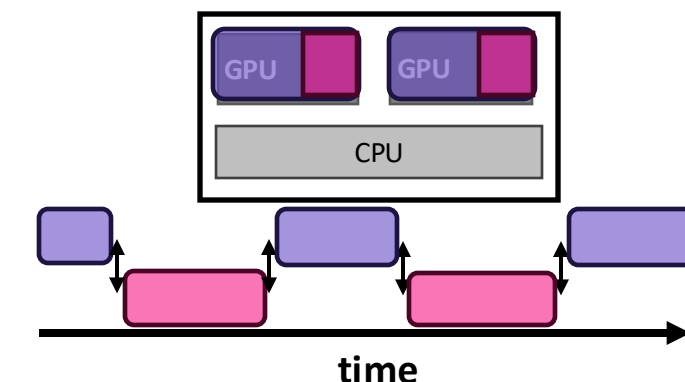
- Components run on separate compute resources
- Concurrent in time, components run simultaneously
- Minimal idle time of components for fast data copy/transfer
- Usually requires indirect memory access with data copy/transfer



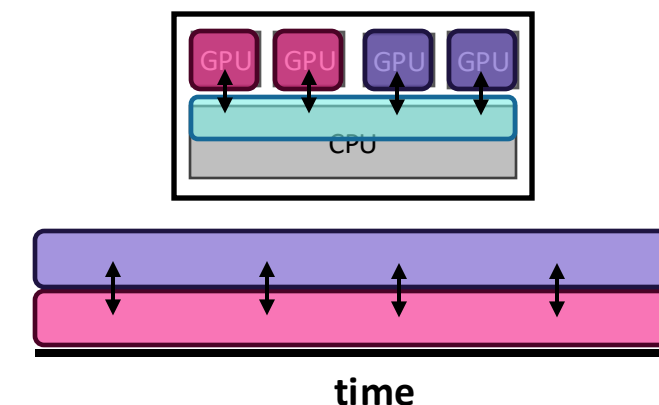
Simulation
Database

ML component
Data transfer

Time Division: Same Compute Hardware



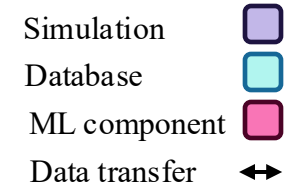
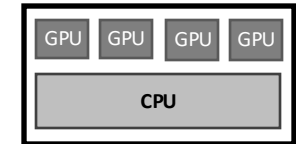
Space Division: Separate Compute Hardware



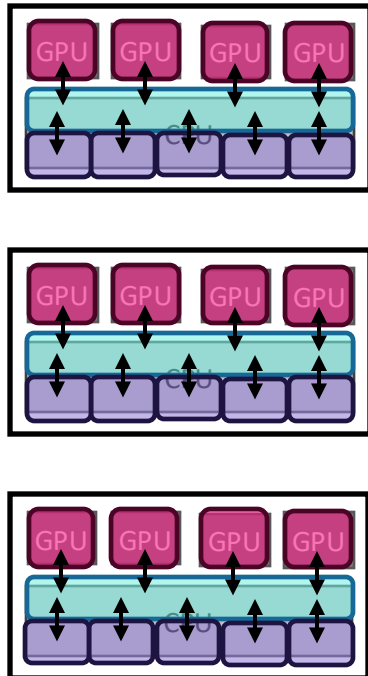
How to Couple HPC Simulations and AI/ML?

Physical Proximity

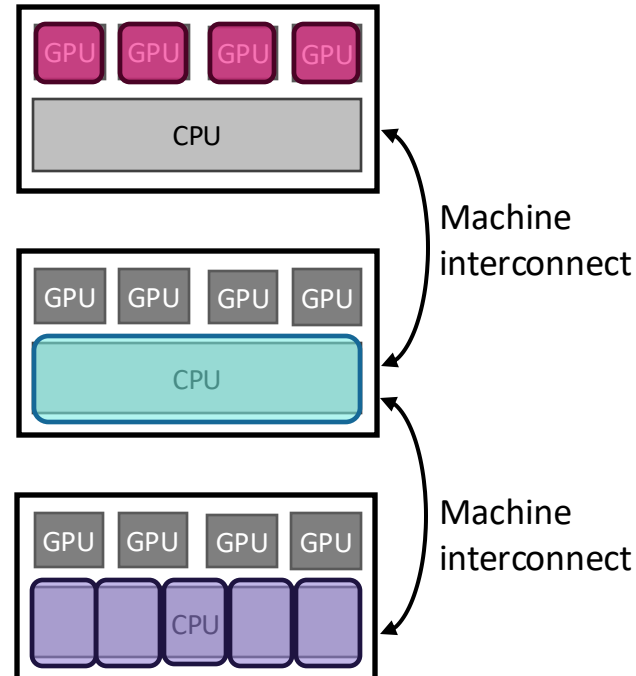
- ❑ Colocation: components share the same nodes
- ❑ Node-level clustering: components use different nodes on the same system
- ❑ Multi-system: components are run on separate specialized systems



Colocated Deployment



Node Clustered Deployment



System Clustered Deployment



How to Couple HPC Simulations and AI/ML?

Data Access

- ❑ Coupling simulation and ML requires frequent data sharing/transfer between components
- ❑ Direct: components share same memory space (may allow for zero-copy data transfer)
- ❑ Indirect: components use distinct logical memory (requires data copy and may require data transfer)

Data Staging or Streaming

- ❑ Staging: data is staged in memory or on disk (can reduce idle time but increases number of transfers)
- ❑ Streaming: data is streamed directly between components (can increase idle time due to synchronization)

How to Couple HPC Simulations and AI/ML?

- ❑ Coupling simulation and AI/ML can be a complex space to navigate
- ❑ Implementation choices can vary significantly depending on ML task and application needs
- ❑ This session will cover some common tools and approaches supported at ALCF

Exercise 1: ML-in-the-Loop for Molecular Design

Science Problem: identify high value molecules (i.e. molecules with high ionization energy) among a large search space of potential candidates

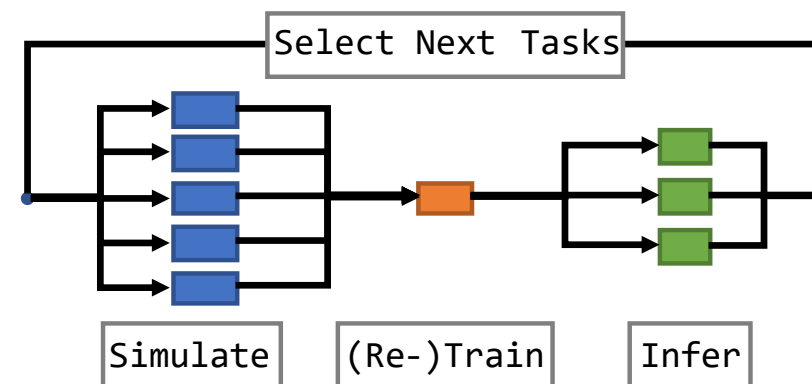
Challenge: The simulation is too computationally expensive to run for every candidate molecule

Approach: Create an active learning workflow that couples simulation with machine learning to simulate only high value candidates

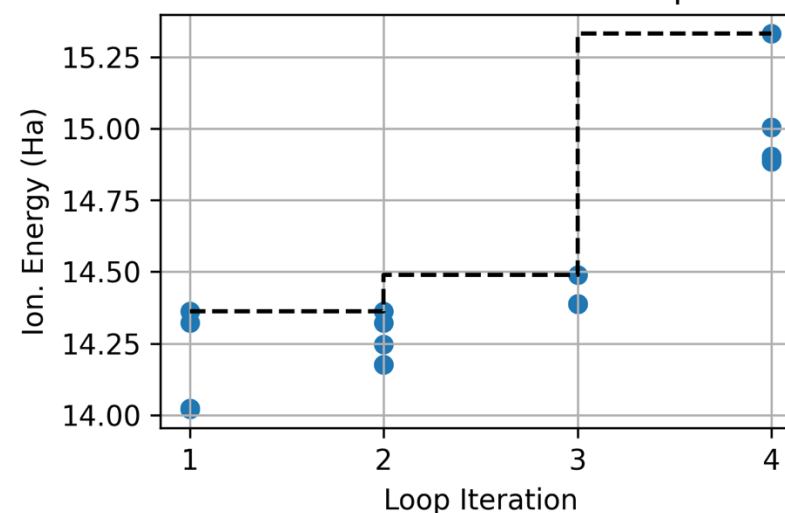
Tools:

- ❑ **Parsl** is used for task launching and integration
- ❑ Use RDkit and scikit-learn to train a k-nearest neighbor (knn) model
- ❑ Simulations done with MD package xTB

Workflow Pattern: ML Components steer Simulations



Best Predicted Molecules over Loop Iterations



ML-in-the-Loop Workflow for Molecular Design

Hands-On Time

Observations from ML-in-the-Loop Exercise

- ❑ AI/ML methods can significantly speed up traditional compute-heavy simulation tasks
 - From 2_training_and_inference.py, screening through ~130,000 compounds with knn model is significantly faster than simulating even just a few compounds

```
Submitted 16 simulations to start training ...  
...  
Training data collected in 9.65 seconds!  
...  
  
Starting training and inference ...  
Training and inference completed in 5.13 seconds!  
...
```

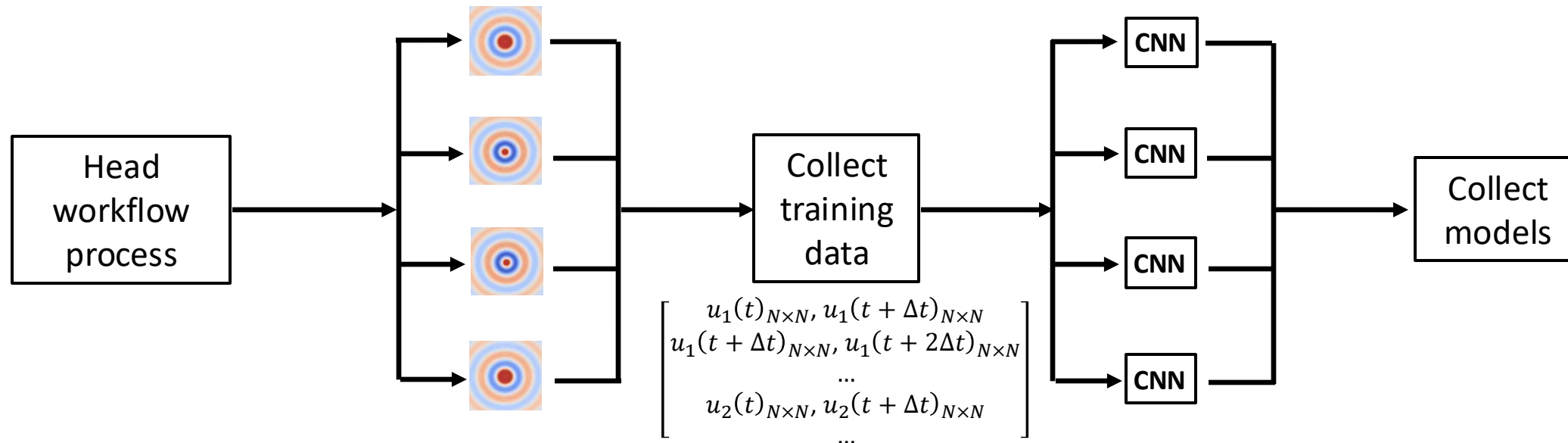
- ❑ Active learning can help refine AI/ML models by intelligently selecting training data and fine-tune models towards a specific task
- ❑ Parsl allows us to seamlessly automate and scale sim+AI workflows by deploying tasks in parallel on multiple Polaris nodes and handling required data dependencies

Exercise 2: Scaling a Producer-Consumer Workflow

- ❑ ML-in-the-loop exercise relied on small amounts of data to train the model and perform inference
- ❑ What happens when we scale up the size of the data?
- ❑ Let's consider a simple producer-consumer workflow to perform training of an ML surrogate

Data Producer

Ensemble of toy simulations to advance 2D wave equation on $N \times N$ grid and produce training data $(u(t)_{N \times N}, u(t + \Delta t)_{N \times N})$



Data Consumer

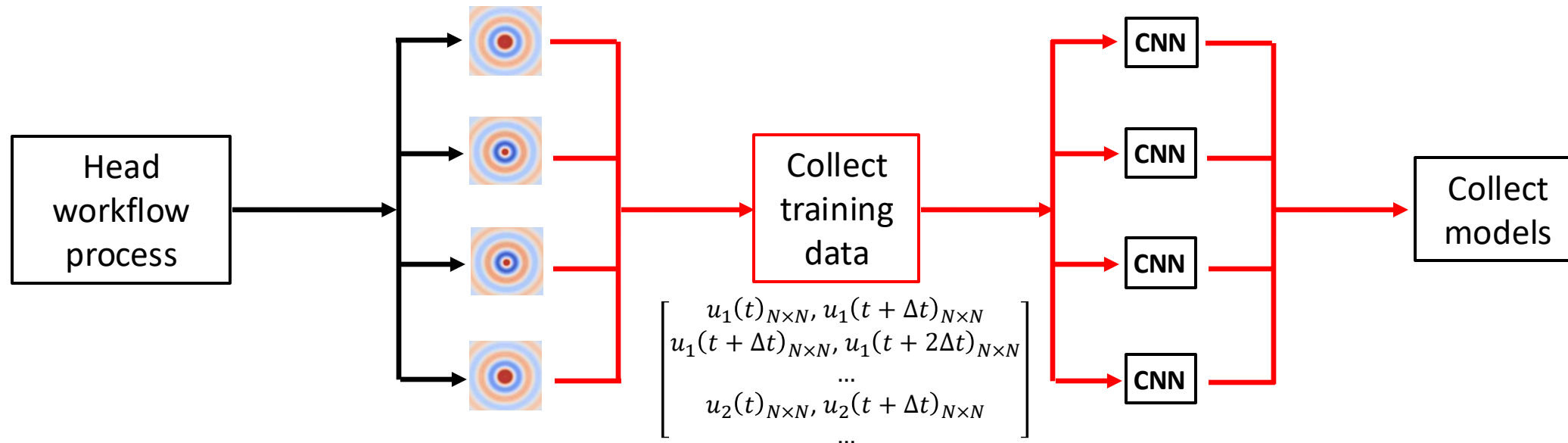
Ensemble of simple CNNs to train auto-regressive model, $u(t + \Delta t) = \text{CNN}(u(t))$, spanning various hyperparameters

Exercise 2: Scaling a Producer-Consumer Workflow

- ❑ As workflow and data scale up, **Parsl implementation runs into transfer speed and memory bottlenecks**
 - Collecting all training data on a single node can lead to out-of-memory issues
 - Multi-node concurrent future implementation relies on TCP transfer (inefficient for large data)

Data Producer

Ensemble of toy simulations to advance 2D wave equation on $N \times N$ grid and produce training data $(u(t)_{N \times N}, u(t + \Delta t)_{N \times N})$



Data Consumer

Ensemble of simple CNNs to train auto-regressive model, $u(t + \Delta t) = \text{CNN}(u(t))$, spanning various hyperparameters

Exercise 2: Scaling a Producer-Consumer Workflow

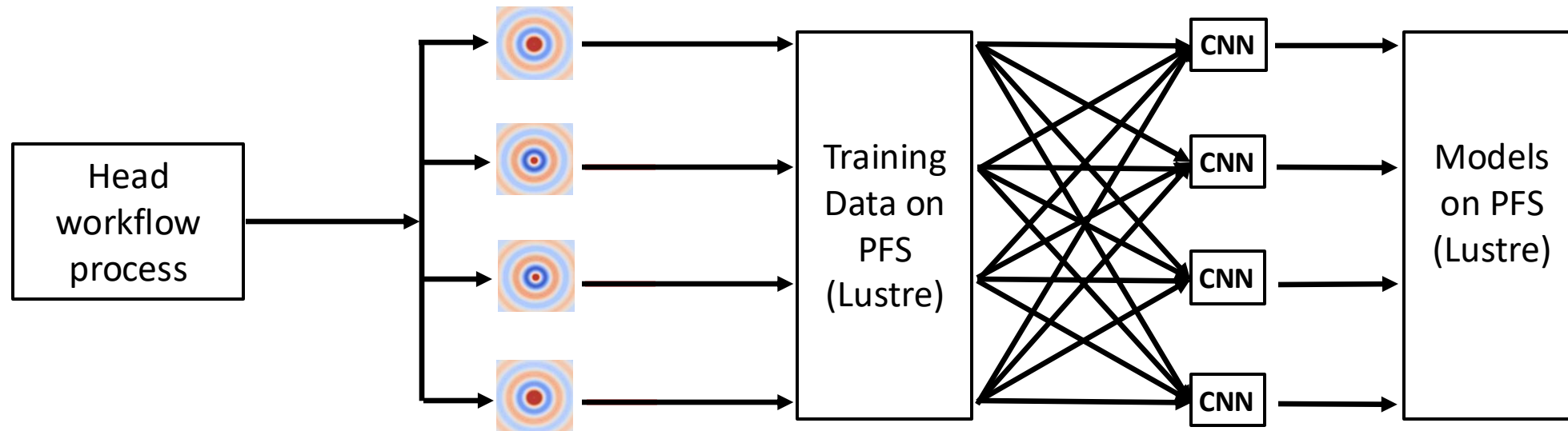
- ❑ Immediate solution would be to store data on the parallel file system (PFS)
 - Each simulation writes its own data to the file system (1 file per simulation, total of N_{sim} files)
 - Each training instance reads all data from simulations (N_{CNN} processes read N_{sim} files)

Data Producer

Ensemble of toy simulations to advance 2D wave equation on $N \times N$ grid and produce training data $(u(t)_{N \times N}, u(t + \Delta t)_{N \times N})$

Data Consumer

Ensemble of simple CNNs to train auto-regressive model, $u(t + \Delta t) = \text{CNN}(u(t))$, spanning various hyperparameters



Exercise 2: Scaling a Producer-Consumer Workflow

❑ At large scale, the file system can become a bottleneck too

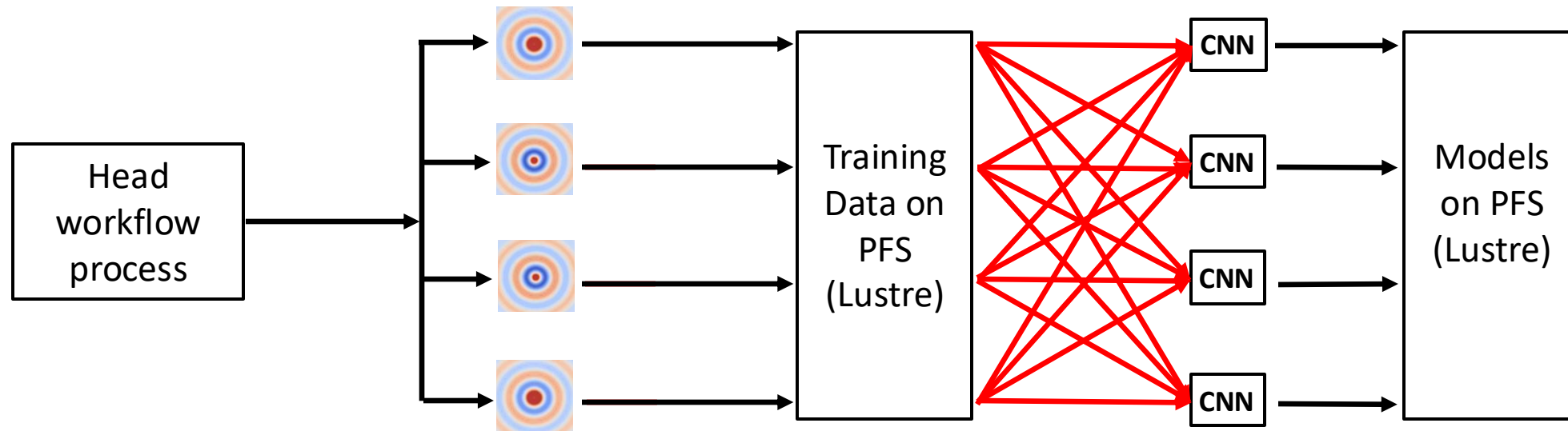
- Metadata server contention from too many concurrent file open/close operations
- Limited parallel IO bandwidth (~650 GB/s for Eagle on Polaris)

Data Producer

Ensemble of toy simulations to advance 2D wave equation on $N \times N$ grid and produce training data $(u(t)_{N \times N}, u(t + \Delta t)_{N \times N})$

Data Consumer

Ensemble of simple CNNs to train auto-regressive model, $u(t + \Delta t) = \text{CNN}(u(t))$, spanning various hyperparameters



Exercise 2: Scaling a Producer-Consumer Workflow

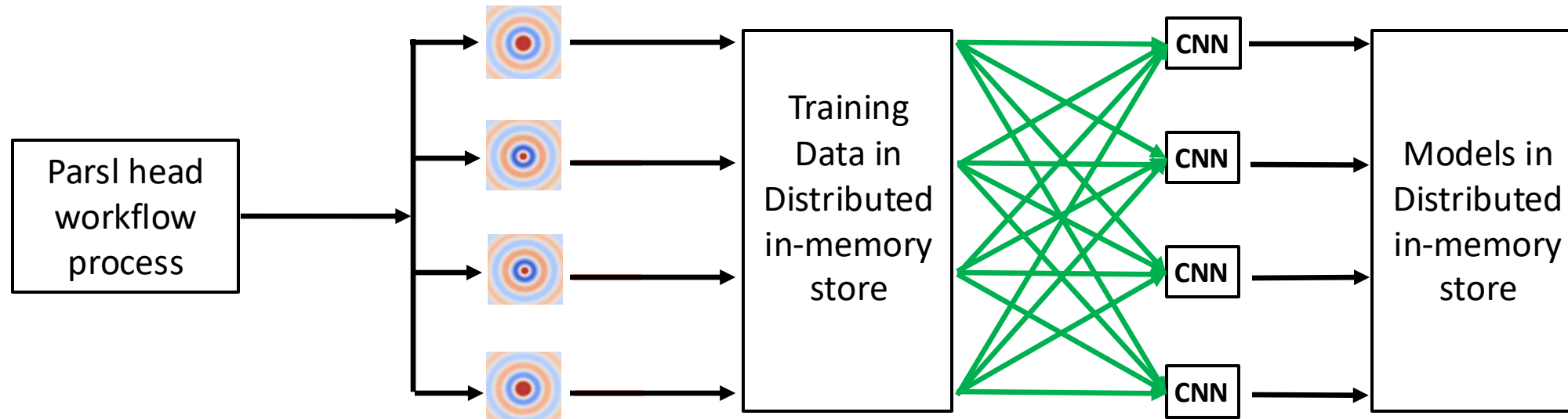
- ❑ File system bottlenecks can be alleviated by leveraging distributed in-memory key-value data stores
 - E.g., DragonHPC Distributed Dictionary or sharded databases, such as Redis
 - Enable fast IO storing data on nodes' DRAM memory in distributed fashion
 - Data transfer across system interconnect (TCP or more efficient RDMA/MPI)
 - In some cases, enable full collocation of data and compute avoiding any inter-node transfer (VERY fast and scalable!)

Data Producer

Ensemble of toy simulations to advance 2D wave equation on $N \times N$ grid and produce training data $(u(t)_{N \times N}, u(t + \Delta t)_{N \times N})$

Data Consumer

Ensemble of simple CNNs to train auto-regressive model, $u(t + \Delta t) = \text{CNN}(u(t))$, spanning various hyperparameters



Scaling a Producer-Consumer Workflow

Hands-On Time

Observations from Producer-Consumer Hands On

- ❑ On a single node, the parallel file system offers the best performance
- ❑ What happens as the workflow scales up in size, both data size and number of nodes? The homework will help you explore this question.

In-depth webinar available:

**Methods, Tools, and Best Practices for Coupling
Simulation and AI on ALCF Systems**

This research used resources of the Argonne Leadership Computing Facility (ALCF), which is a DOE Office of Science User Facility supported under Contract DE-AC02-06CH11357.