October 10-12, 2023

ALCF Hands-on HPC Workshop

Integrating AI/ML and Simulation

ALCF Hands-On HPC Workshop, October 10 - 12, 2023

Riccardo Balin
Postdoctoral Appointee, ALCF
October 11, 2023

Riccardo Balin

- Postdoctoral appointee, ALCF
- Interests and expertise:
  - Computational fluid dynamics and turbulence modeling
  - Simulations and modeling of complex turbulent flows for aerodynamic applications
  - Scientific ML and applications to CFD for closure and surrogate modeling, and flow state compression
  - Coupling simulations and AI/ML for scalable online learning workflows on HPC clusters

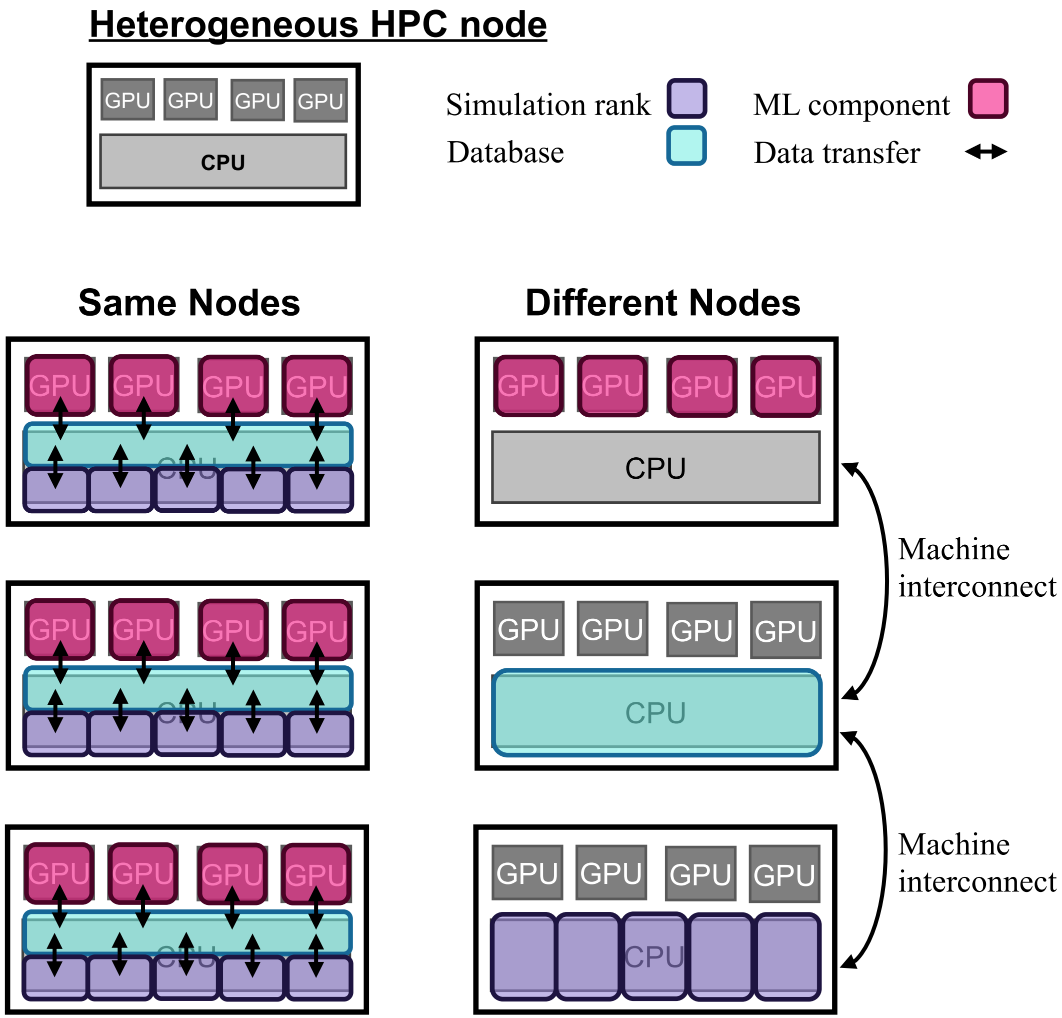
Why Couple HPC Simulations and AI/ML?

- Substitute inaccurate or expensive components of simulation with ML models
  - Closure or surrogate modeling
- Control simulation with ML
  - Select numerical scheme or input parameters
- Avoid IO bottleneck and disk storage issues
  - Online training through data streaming and in-memory storage
- Fine tune models online
  - Access training data not available during pre-training offline
- Active learning
  - Continuous improvement of ML model training as simulation progresses

How to Couple HPC Simulations and AI/ML?

Physical Proximity

- Components share the same node
- Components use different nodes on the same system
- Components are run on separate specialized systems



Childs et al., "A terminology for in situ visualization and analysis systems", Intl. Journal of High Performance Computing Applications, 2020

Balin et al., arXiv:2306.12900, 2023.

How to Couple HPC Simulations and AI/ML?

Data Access

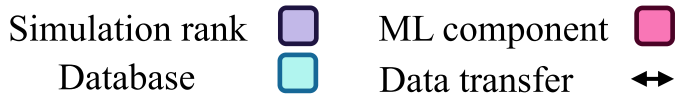
- Direct: components share same memory space (may allow for zero-copy data transfer)
- Indirect: components use distinct logical memory (requires data copy and may require data transfer)
- Either way, requires frequent memory synchronization

Childs et al., “A terminology for in situ visualization and analysis systems”, Intl. Journal of High Performance Computing Applications, 2020

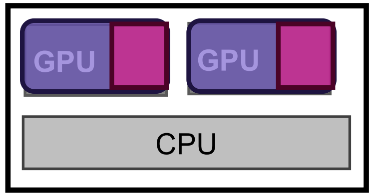
How to Couple HPC Simulations and AI/ML?

Execution Management

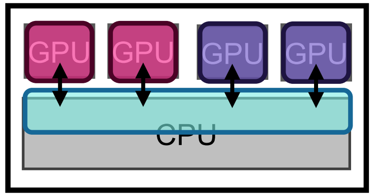
- Time division (tight coupling)
  - Components run on same compute resources (may even use same processes)
  - Staggered in time, execution of one component halts the other
  - May allow for direct memory access and no data copy/transfer
  - Idle time of individual components may be significant
- Space division (loose coupling)
  - Components run on separate compute resources
  - Concurrent in time, both components run simultaneously
  - Minimal idle time of components for fast data copy/transfer
  - Usually requires indirect memory access with data copy/transfer



Time Division: Same Compute Resource



Space Division: Same Node



Childs et al., "A terminology for in situ visualization and analysis systems", Intl. Journal of High Performance Computing Applications, 2020

Software for Coupling Simulations and AI/ML

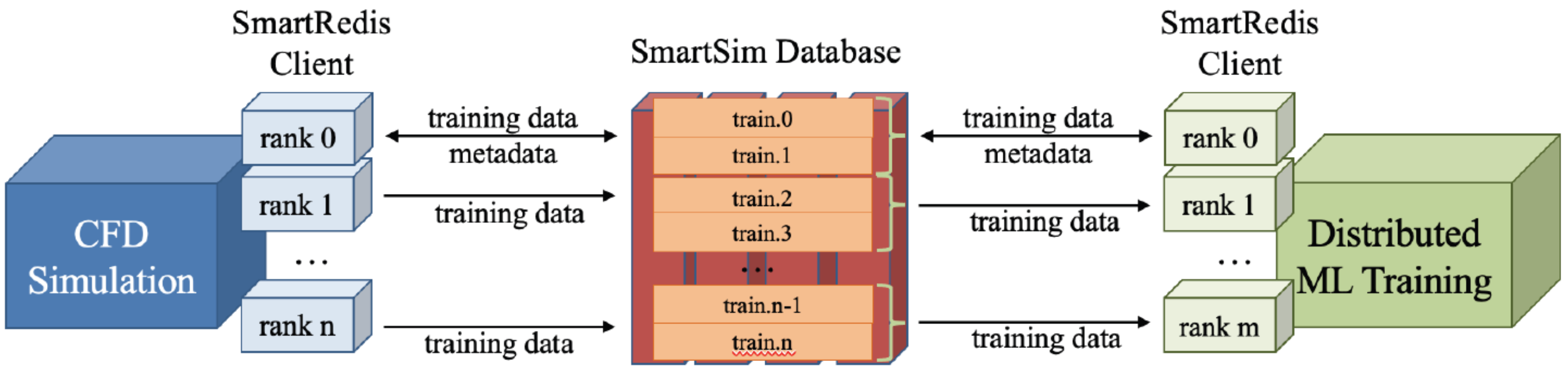
- Tight coupling
  - Python and ML frameworks embedding into simulation code
    - PythonFOAM, TensorFlowFOAM and nekRS (by Romit Maulik, Saumil Patel, Bethany Lusch at ALCF)
  - Linking to LibTorch or ONNX Runtime libraries for ML inferencing from C, C++ and Fortran
    - Aurora will support LibTorch and Intel's OpenVINO inference library
  - Usually more performant and preferred for inferencing (ML model deployment within simulation)
- Loose or no coupling
  - SmartSim / SmartRedis
    - Workflow manager and client libraries for in-situ workflows by sharing data across a database
  - ADIOS2
    - Same I/O API to transport data across different media (file, wide-area-network, in-memory staging, etc.), favoring asynchronous streaming
  - Dragon
    - Run-time library for managing dynamic processes, memory, and data at scale through high-performance communication
  - Usually preferred for training thanks to concurrency and greater flexibility of workflow

Coupling Simulations and AI/ML with SmartSim

- Open source tool developed by HPE designed to facilitate the integration of traditional HPC simulation applications with machine learning workflows
- Infrastructure library (IL)
  - Python API to start, stop and monitor HPC applications from Python (workflow driver)
  - Interfaces with the scheduler launch jobs (PBSPro on Polaris and Cobalt on Theta/ThetaGPU)
  - Deploys a distributed in-memory database called the Orchestrator
- SmartRedis client library
  - Provides clients that connect to the Orchestrator from Fortran, C, C++, Python code
  - The client API library enables data transfer to/from database and ability to load and run JIT-traced Python and ML runtimes acting on stored data

Coupling Simulations and AI/ML with SmartSim

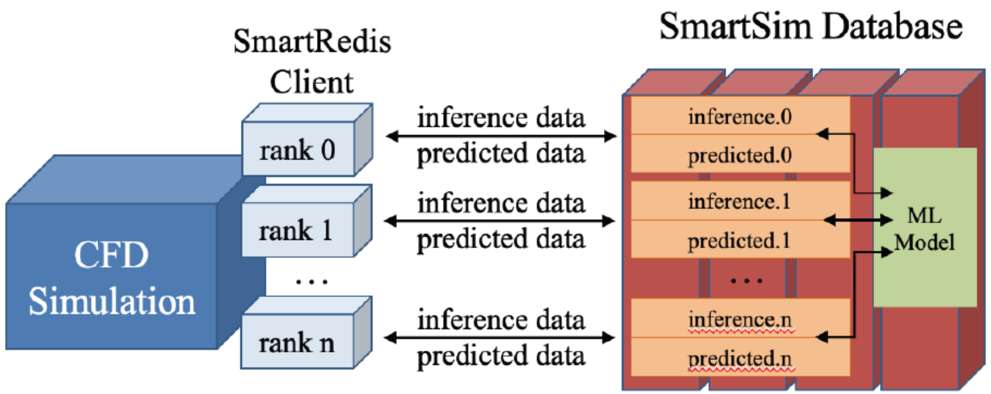
- Online training of ML models from ongoing simulation
  - Data flows from the data producer (e.g., a numerical simulation) through the SmartSim database to the data consumer (e.g., a distributed training program)
  - Simulation and the ML training run simultaneously
  - Training data stored in-memory within database for duration of job, no I/O bottleneck and disk storage issues
  - Fully decoupled – components run independently, without blocking and on separate resources
  - Can connect multiple components through the database



Balin et al., arXiv:2306.12900, 2023.

Coupling Simulations and AI/ML with SmartSim

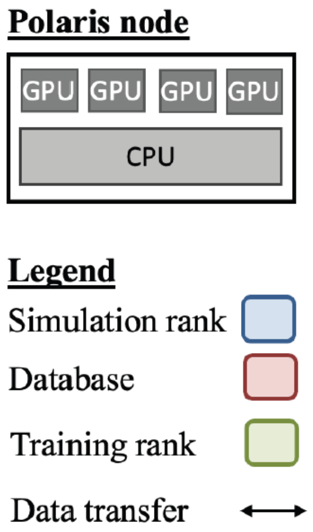
- Online inference of ML models
  - Simulation sends/retrieves model inputs and outputs and evaluates ML models on data within database
  - Compatible with TensorFlow, TensorFlow Lite, Torch, and ONNX Runtime backends for model evaluations
  - Supports model evaluation on CPU and GPU
  - Loosely coupled – components run on separate resources but inference blocks simulation progress

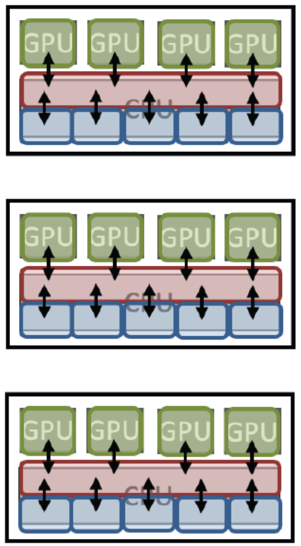


Balin et al., arXiv:2306.12900, 2023.

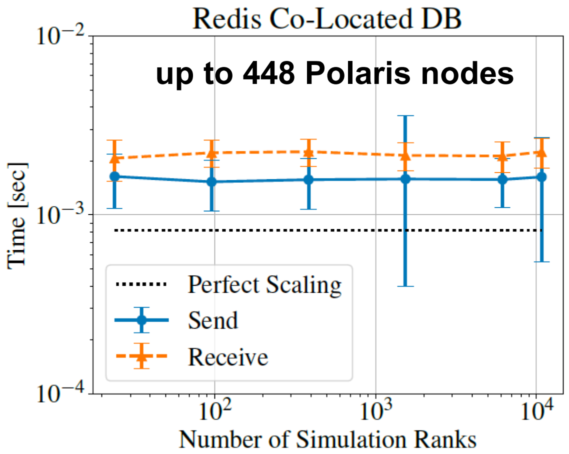
Coupling Simulations and AI/ML with SmartSim

- Scalable colocated deployment
  - Database, simulation and ML component share resources on each node
  - Distinct database is deployed on each node
  - Highly scalable – constant overhead from data transfer to/from database!
  - Good use of node compute resources
  - Training/inference data is distributed across the various databases, accessing off-node data is non-trivial

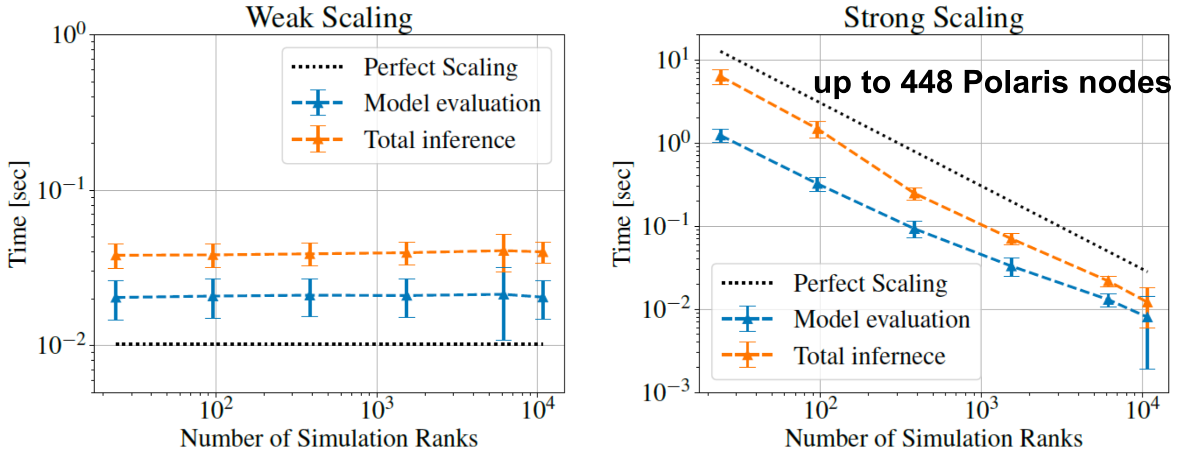




Send/Retrieve Weak Scaling Test



Inference Scaling Tests



Balin et al., arXiv:2306.12900, 2023.

Demo: Online Training and Inference with nekRS

Feel free to follow along by pulling material from workshop repo

```
git clone https://github.com/argonne-lcf/ALCF_Hands_on_HPC_Workshop.git
```

OR if already cloned the repo

```
git pull origin master
```

Then, switch to demo directory and submit interactive job

```
cd couplingSimulationML/NekRS-ML
./subInteractive.sh
```

Demo: Online Training and Inference with nekRS

Environment Setup

- Conda environment with the SmartSim modules for Polaris is available in the workshop project folder
- Demos use this environment
- Activate with

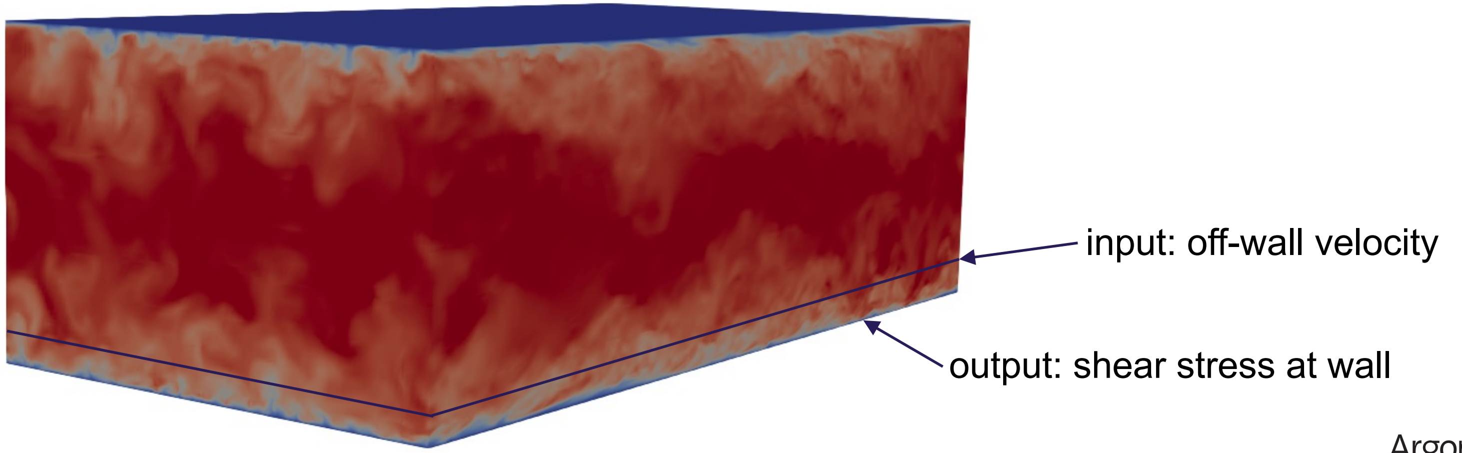
```
module load conda/2022-09-08
conda activate /eagle/projects/fallwkshp23/SmartSim/ssim
```

- Note:
  - This conda env does not contain all the modules available with the base env from the conda/2022-09-08 module, but many of the essential ones for distributed training
  - New env and instructions based on latest conda module is coming soon, along with more instructions on Polarid documentation
  - More information on this env and building it are found [at the workshop repo]

Demo: Online Training and Inference with nekRS

- Goals:
  - Use SmartSim/SmartRedis to train an ML model from ongoing CFD simulation
  - Call model from CFD code for inference
  - Make use of GPU on Polaris nodes
- Using an in-house fork of nekRS, called nekRS-ML
  - nekRS is a popular, efficient and scalable code tested on Polaris and Aurora
  - nekRS-ML is ALCF sandbox for various approaches of integrating code with ML
- Performing wall-modeling
  - Estimate the wall-shear stress of a turbulent channel flow from the velocity at a location above the wall

Turbulent Channel Flow at $Re_\tau = 550$



Demo: Online Training and Inference with nekRS

Training example

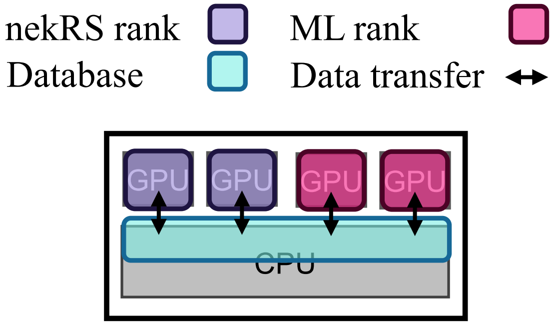
- From interactive session

```
cd train_example
source env.sh
./run.sh
```

- OR submit batch script

```
cd train_example
qsub submit.sh
```

- Both nekRS and training run in parallel and on GPU of same node
- Database deployed on CPU



- Training data sent to database every 10 time steps

nekrs.out

```
copying solution to nek

Sending field with key x.0.10
Done

Sending time step number ...
Done
```

train_model.out

```
Getting new training data from DB ...
Added time step 30 to training data

Epoch 3 --------------------------------
[0]: Grabbing tensors with key ['x.0.30', 'x.1.20', 'x.1.10’]
[1]: Grabbing tensors with key ['x.0.10', 'x.0.20', 'x.1.30']
```

Demo: Online Training and Inference with nekRS

Inference example

- From interactive session

```
cd inference_example
source env.sh
./run.sh
```

- OR submit batch script

```
cd inference_example
qsub submit.sh
```

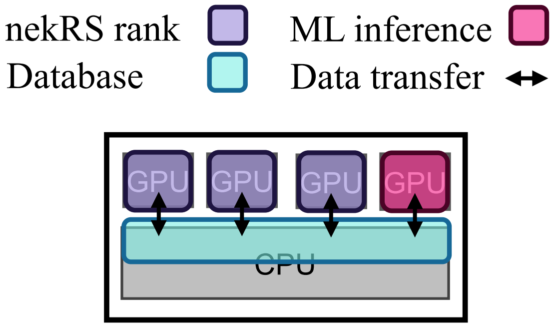
nekrs.out

```
Sending field with key x.0
Done

Running ML model ...
Done

Retrieving field with key y.0
Done
```

- nekRS runs in parallel on 3 GPU
- Inference performed on 4th GPU through database
- Database deployed on CPU



- Inference performed every 10 time steps

Demo: Online Training and Inference with nekRS

- More details on training and inference examples available at workshop repo
  - SmartSim driver script managing workflow and deploying components
  - How to scale out to multiple nodes

Thank you, any questions?

Please direct any additional questions to support@alcf.anl.gov