CFDML – Data Analytics and ML for Exascale CFD

Riccardo Balin
Postdoctoral Appointee
ALCF
rbalin@anl.gov

ALCF 2022 Simulation, Data and Learning Workshop

10/04/2022

Overview

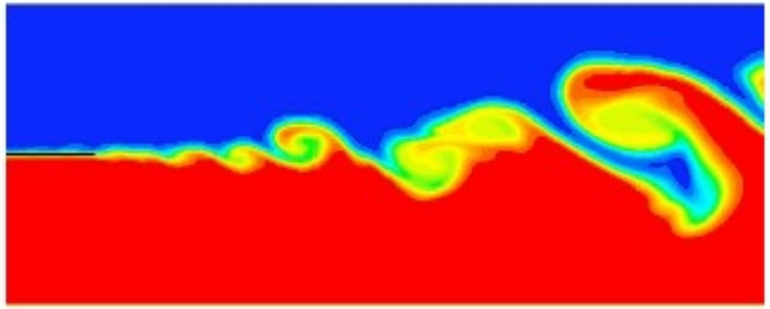
- Introduction and science goals
- Online ML with SmartSim
- Performance on Polaris and Theta
- Conclusions and future work

Overview

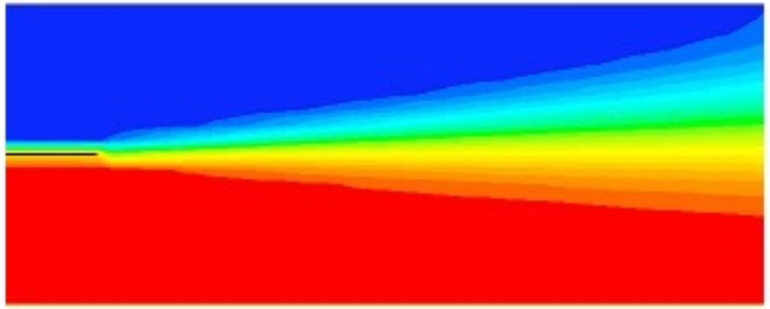
- Introduction and science goals
- Online ML with SmartSim
- Performance on Polaris and Theta
- Conclusions and future work

Introduction

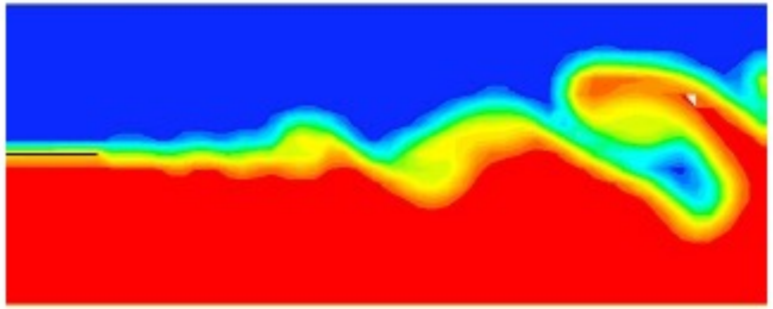
- There are 4 main modeling approaches to computations of turbulent flow



Direct numerical simulation (DNS)
- Solve unsteady Navier-Stokes (NS) equations directly
- Resolve all spatial and temporal turbulent scales, no modeling
- Most accurate
- Most computational expense



Reynolds-averaged NS (RANS)
- Solve for the steady mean flow directly
- Model all spatial and temporal turbulent scales
- Inaccurate for complex flows
- Least computational expense



Large eddy simulation (LES)
- Solve unsteady filtered NS equations
- Resolve largest spatial scales and model smallest (sub-grid) scales
- Modest accuracy
- Modest computational expense

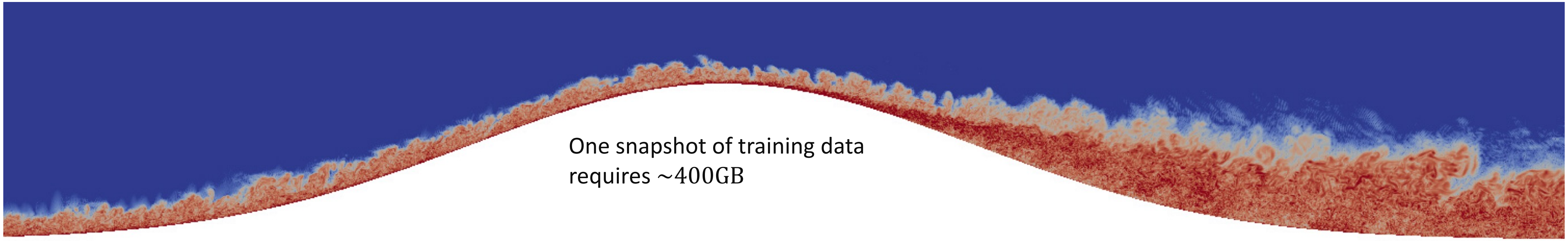
Hybrid RANS/LES and Wall-Modeled LES (WMLES)
- Solve unsteady RANS and/or LES equations
- Model all turbulent scales of the near-wall flow

Developing closure models for LES using ML approaches

Introduction

- Neural net (NN) closure models for RANS are trained on mean flow data
  - Training data easily stored on disk, even when considering multiple flows
  - Offline learning is sufficient and preferred
  - Data production and training performed separately
- NN models for LES and WMLES closures must be trained on instantaneous flow data
  - For petascale and exascale simulations, expensive multi-terabyte databases are needed to store training data
  - Online (in situ) learning offers attractive solution to avoid I/O and storage bottleneck
  - Data production and training performed concurrently

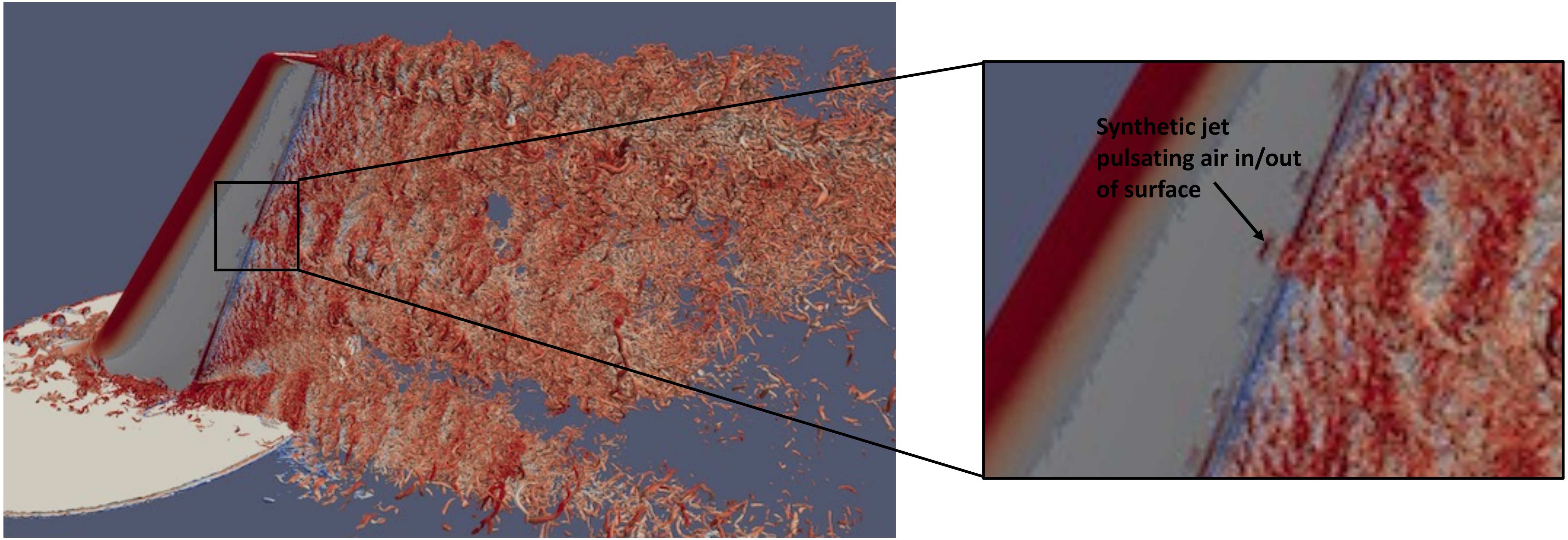
DNS of a Turbulent Boundary Layer Over a Bump at $Re = 2\text{M}$



Science Goals

- Flows over bumps are insightful, but what are we really after?
  - Full CFD simulations of complex aeronautical and aerospace systems
  - NASA vision 2030: towards aircraft certification by simulation

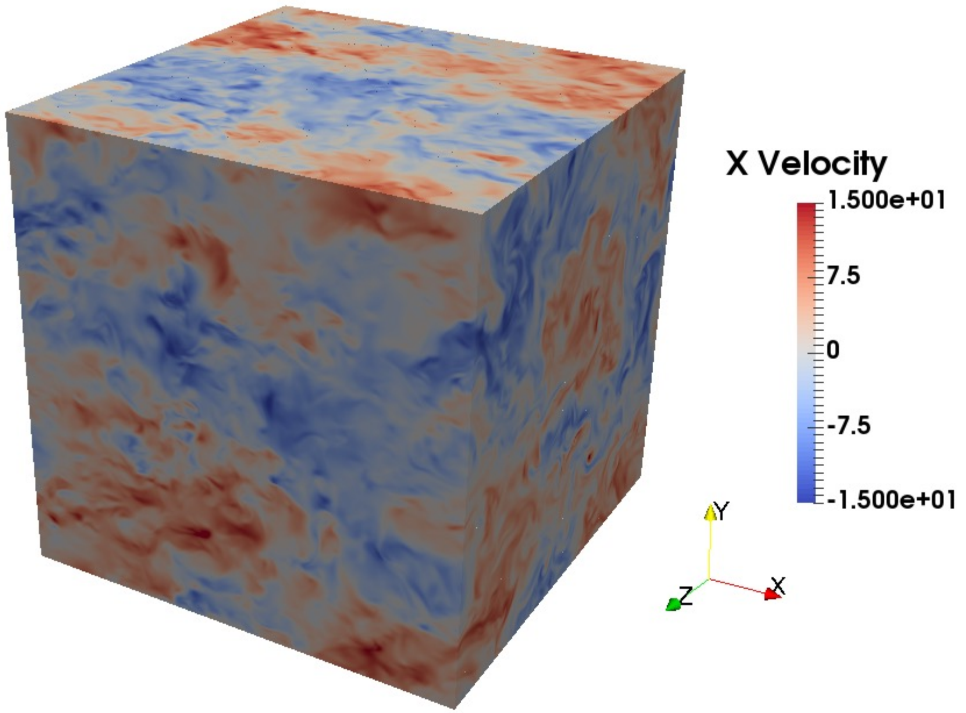
Hybrid RANS/LES of Flow Around Aircraft Vertical Tail with Active Flow Control



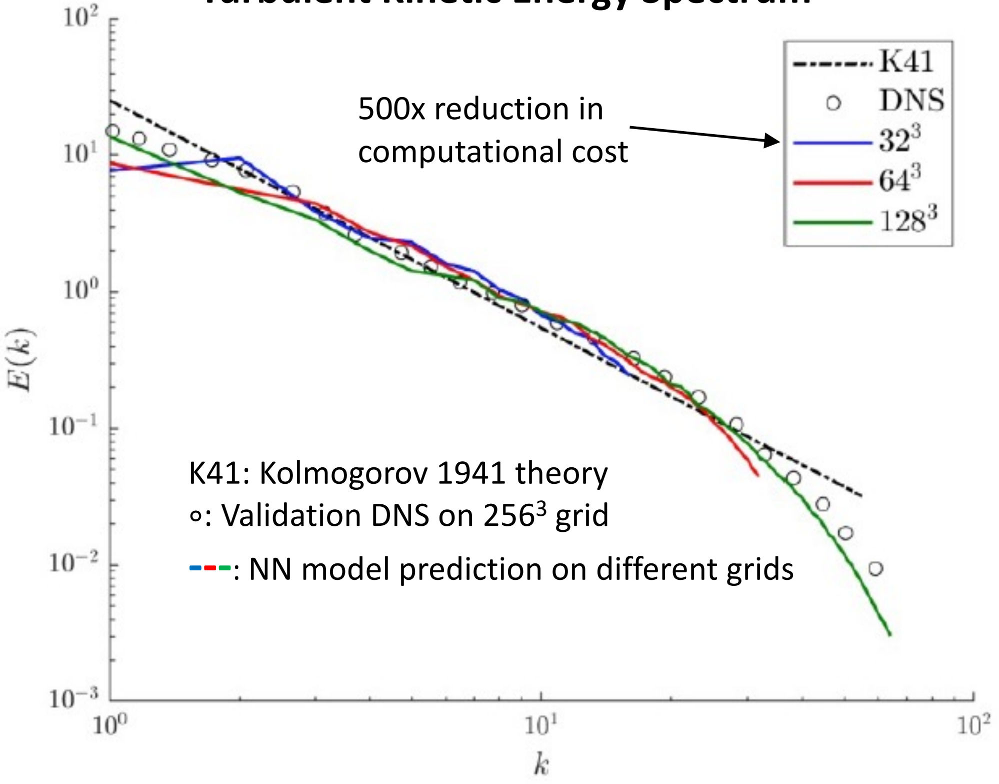
Science Goals

- NN model for the unclosed term in LES equations
- Goal is to predict local and instantaneous sub-grid stress (SGS) tensor, $\tau_{ij} = \widetilde{u_i u_j} - \tilde{u}_i \tilde{u}_j$
- Model is predictive on homogeneous isotropic turbulence, but needs to be extended to wall-bounded flows

Forced Homogeneous Isotropic Turbulence



Turbulent Kinetic Energy Spectrum



Overview

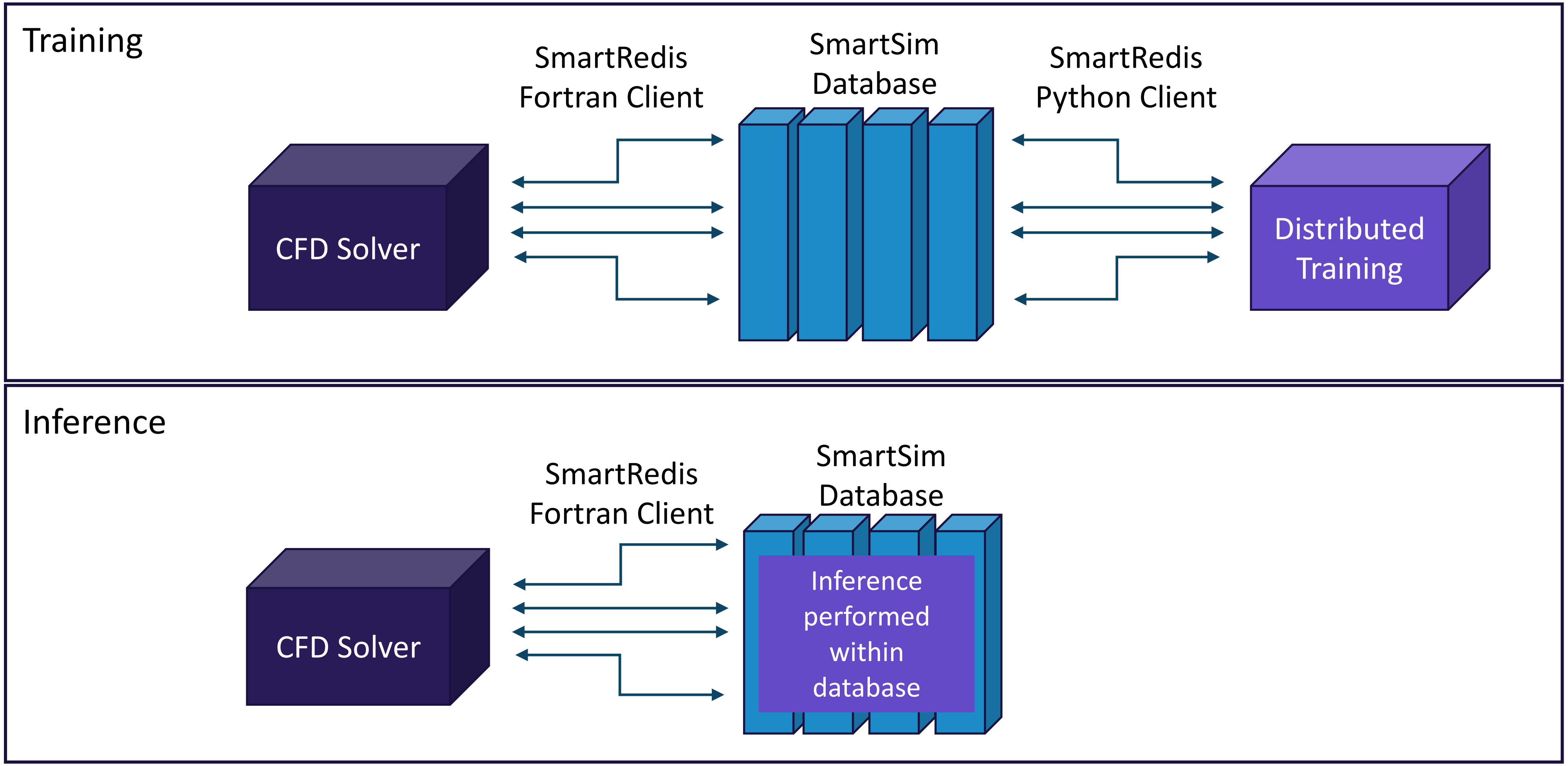
- Introduction and science goals
- Online ML with SmartSim
- Performance on Polaris and Theta
- Conclusions and future work

Online ML with SmartSim

- We use SmartSim and SmartRedis to build workflows for online ML capabilities
- SmartSim/SmartRedis are open-source tools developed by HPE
- Learn more about SmartSim:
  - Integrating AI and Simulations session, Thursday 1:00 - 1:30 pm CST
  - Tutorial and hands-on exercises on Polaris

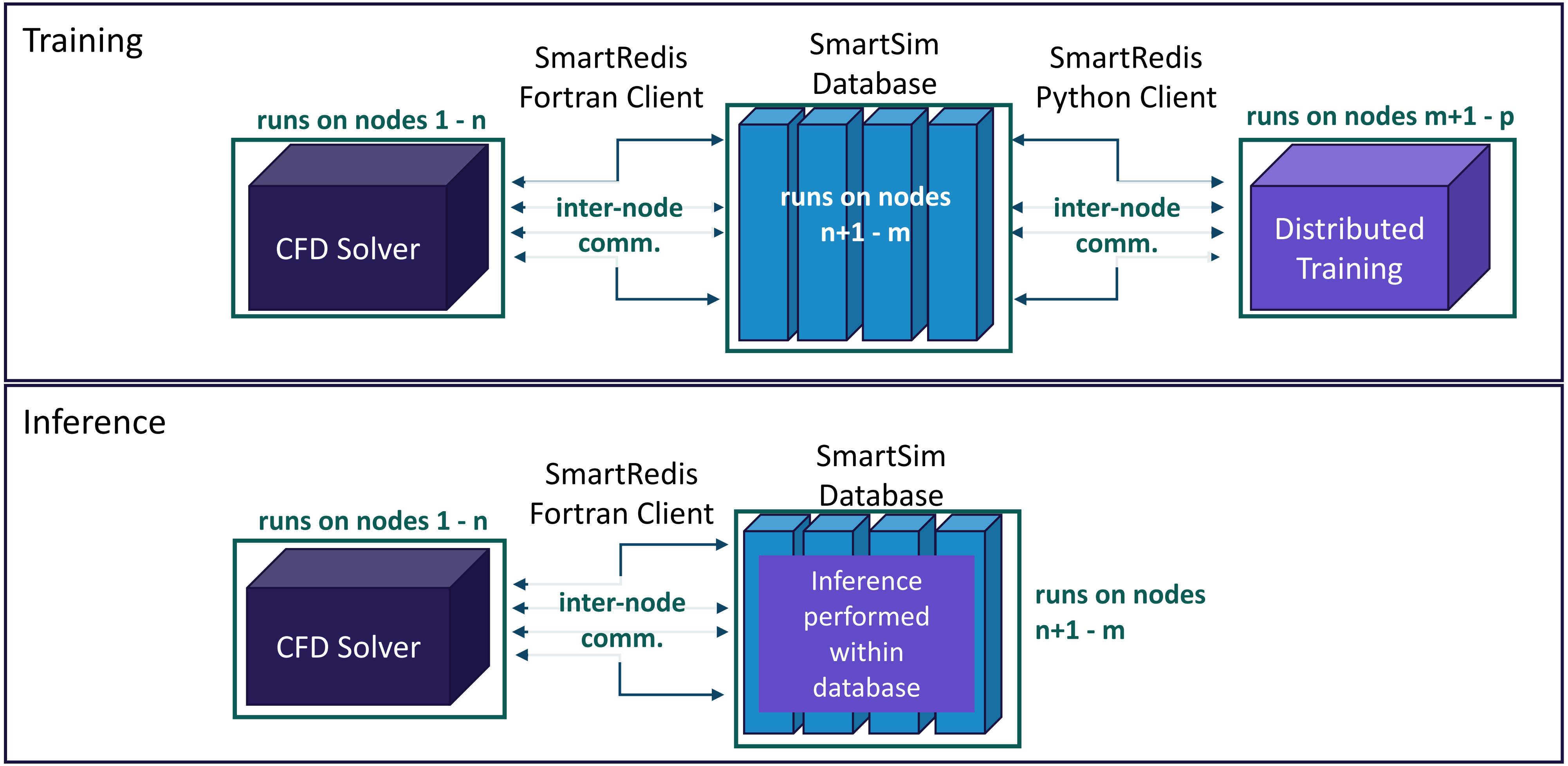
Online ML with SmartSim

- We developed workflows for online training and inference



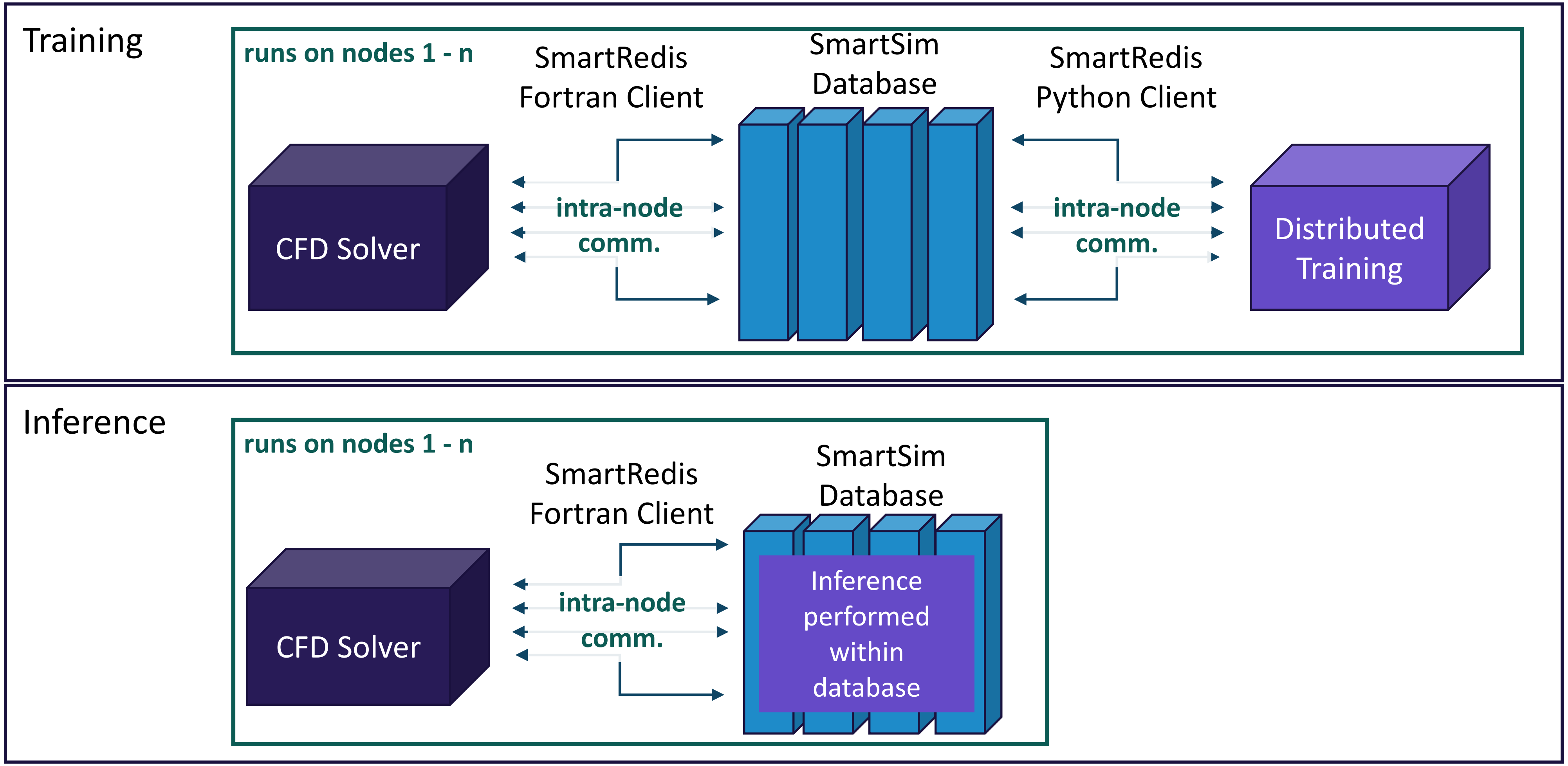
Online ML with SmartSim

- We developed workflows for online training and inference
- Two implementations: **clustered** and co-located



Online ML with SmartSim

- We developed workflows for online training and inference
- Two implementations: clustered and **co-located**



Online Training and Inference with SmartSim

Clustered Implementation

- Single database sharded across a cluster of nodes
- SmartSim Orchestrator, PHASTA and distributed training run on distinct set of nodes
- Pros: all data contained in single database visible by all applications, most flexibility of workflow
- Cons: reduced data transfer performance as PHASTA and distributed training scale out

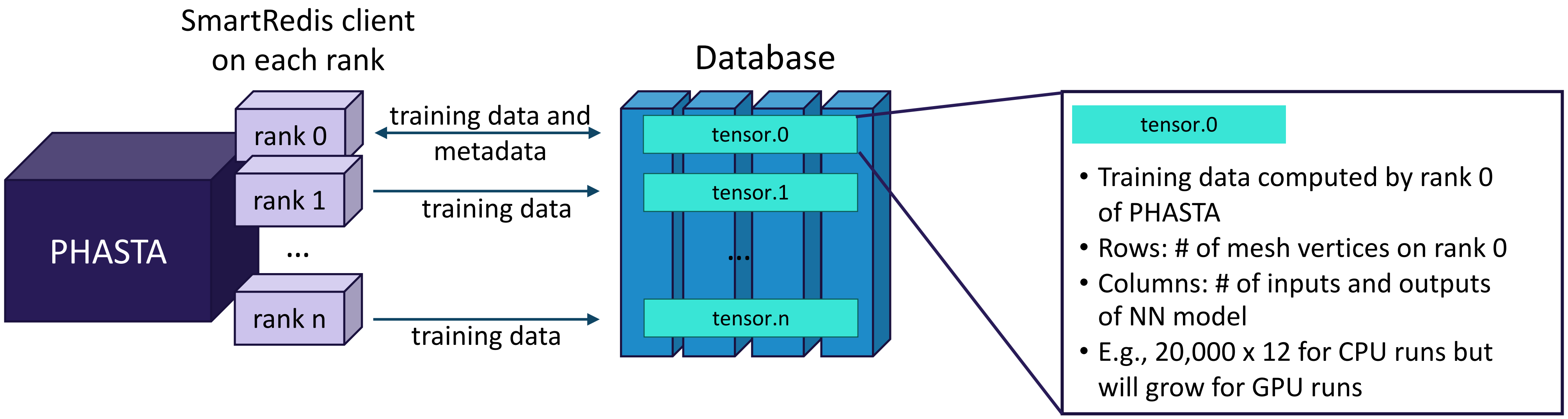
Co-Located Implementation

- Distinct databases launched on each node
- SmartSim Orchestrator, PHASTA and distributed training share resources on each node
- Pros: most efficient data transfer performance at scale
- Cons: data distributed across many Orchestrators, accessing off-node data is non-trivial

Online Training with SmartSim

Data Production for Online Training

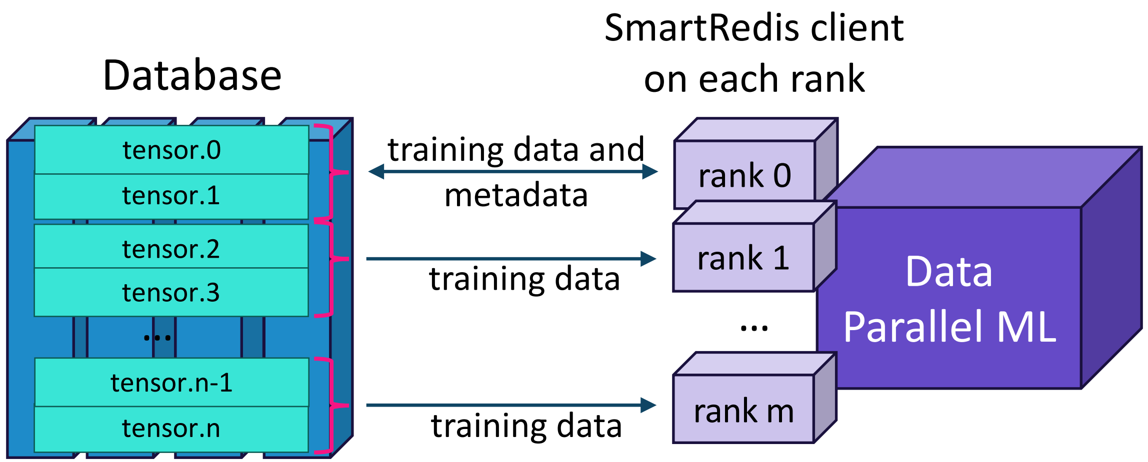
- PHASTA flow solver (mostly Fortran) or other CFD code
- During training:
  - We use domain decomposition, so each PHASTA rank works on a partition of entire domain
  - Solution states processed online and in parallel to compute model inputs and outputs
  - Each rank sends training data to database with unique key for its domain partition
  - Database will contain (num. ranks x num. time steps) distinct tensors
- Computation and transfer of training data done at specified frequency (e.g., every 50-100 time steps)



Online Training with SmartSim

Data Consumption for Online Training

- Distributed ML algorithm for training of turbulence model closures
- Runs at scale concurrently with PHASTA simulation
- During training:
  - Each epoch, set of randomly chosen tensors is retrieved by each ML rank from database
  - Each ML rank performs mini-batch stochastic gradient descent with synchronized optimizer steps
  - When new data placed in database by PHASTA, training continues with latest training data
  - Tensors in database can be overwritten (train on one snapshot at a time) or accumulated (train on multiple snapshots)



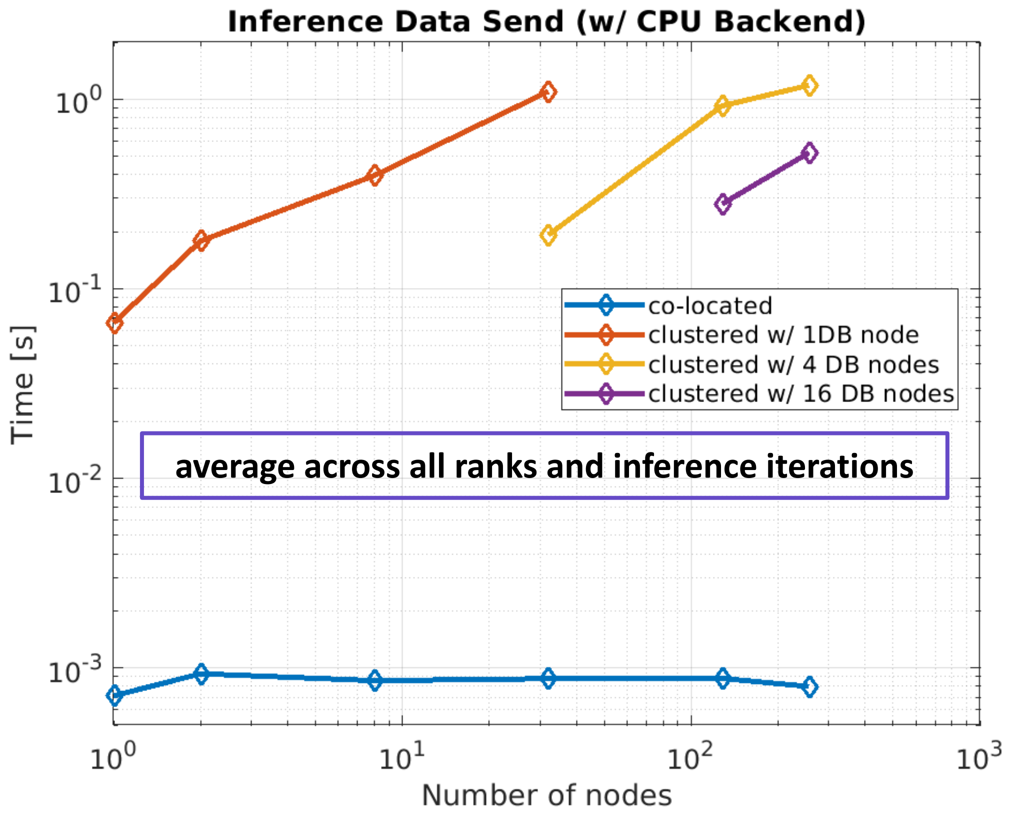
Overview

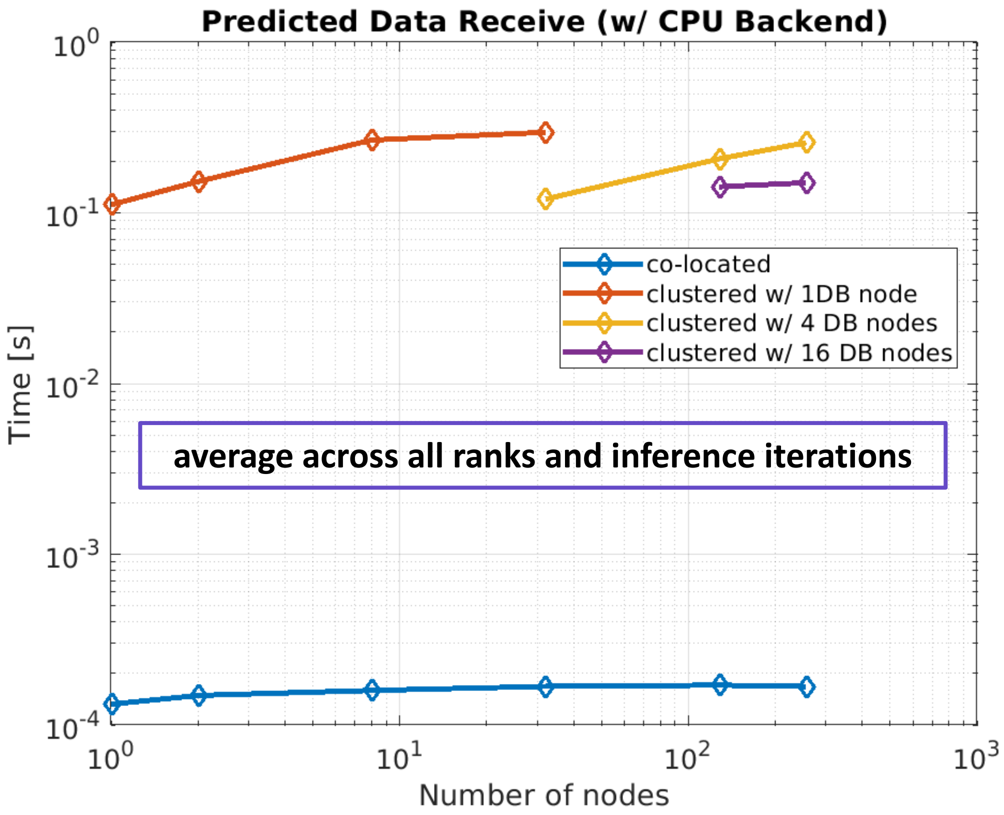
- Introduction and science goals
- Online ML with SmartSim
- Performance on Polaris and Theta
- Conclusions and future work

Scaling Performance on Polaris

Online Inference

- Data transfer overhead with co-located and clustered approaches

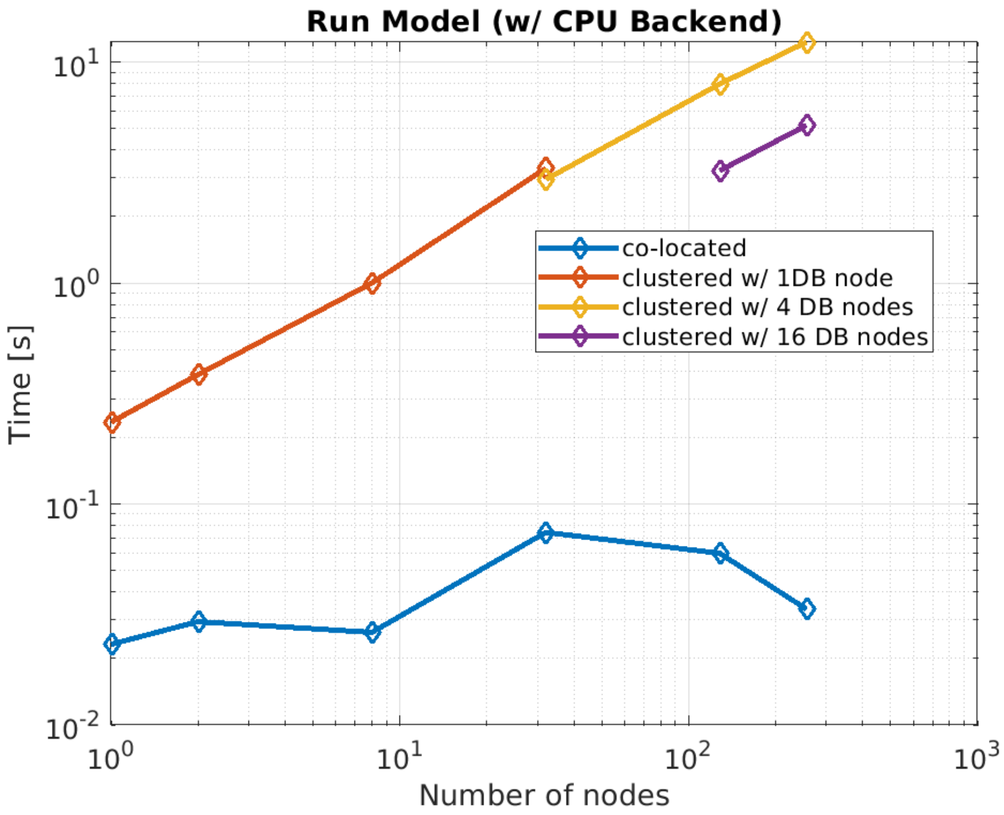




Scaling Performance on Polaris

Online Inference

- Model evaluation overhead with co-located and clustered approaches
- Model ran on CPU

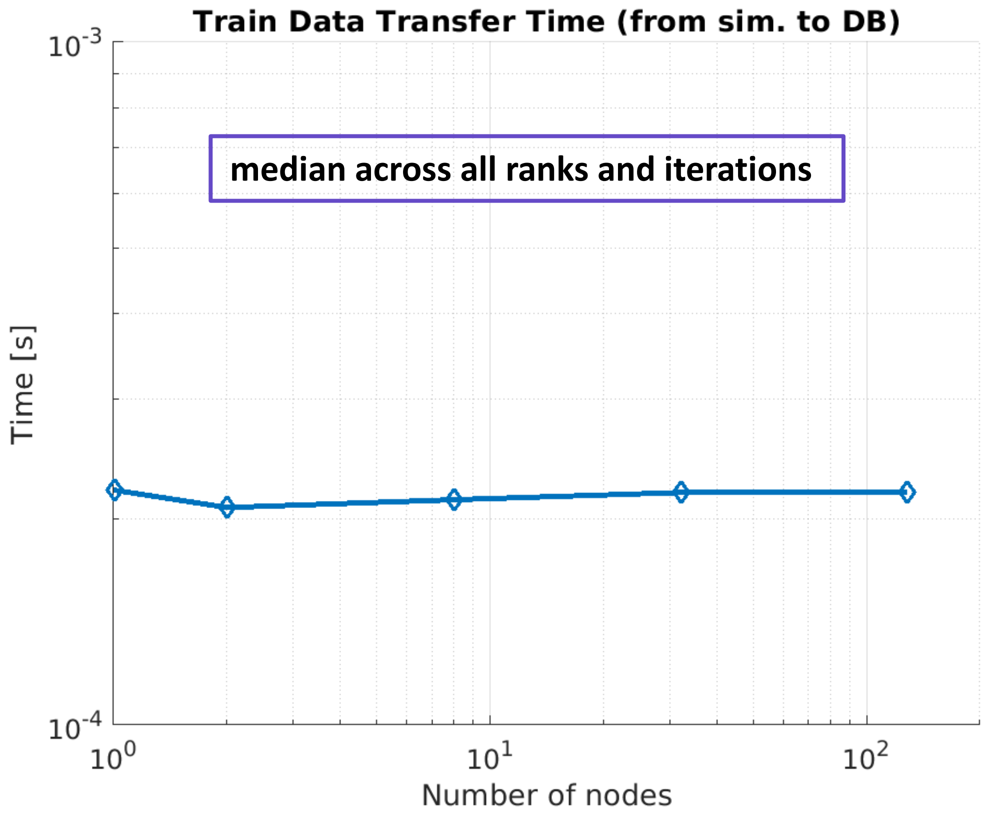


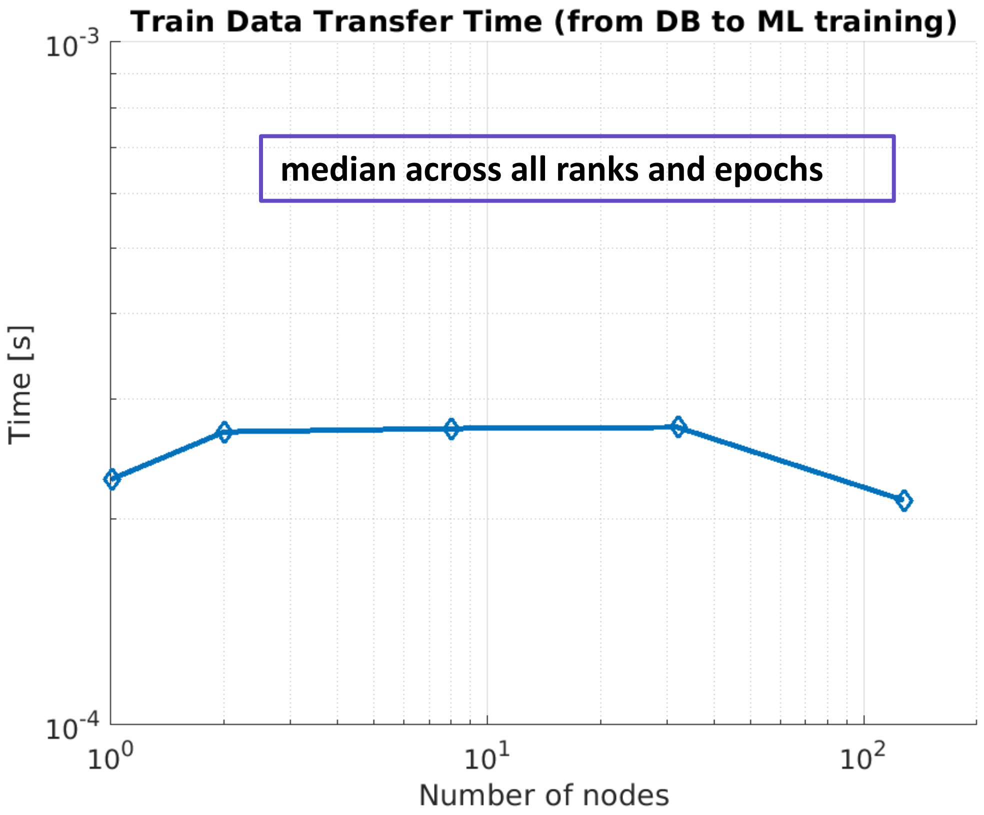
average across all ranks and inference iterations

Scaling Performance on Polaris

Online Training

- Training data transfer overhead with co-located approach
- Sent from simulation to DB and received by ML training from DB

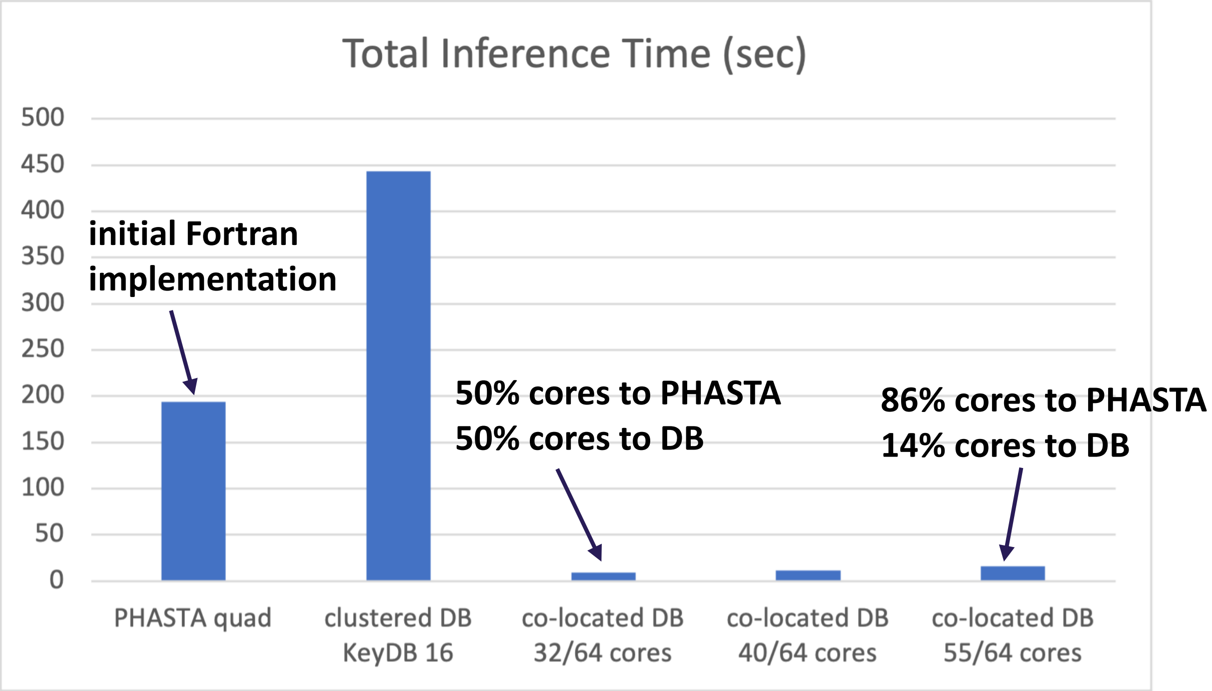




Performance on Theta

Online Inference with PHASTA

- Co-located approach for online inference using 110-220 nodes
- Significant speedup relative to SmartSim clustered implementation and initial PHASTA native implementation in Fortran



Note:

- Database can consume small fraction of CPU cores with small performance drop
- Total inference time with co-located database reduced to 1% of solver time step

Overview

- Introduction and science goals
- Online ML with SmartSim
- Performance on Polaris and Theta
- Conclusions and future work

Conclusions

- Developed a software infrastructure to combine CFD with online (in situ) ML at scale
- Capable of performing online training and inference of NN models efficiently
- Co-located implementation shows close to constant data transfer cost with increasing scale
- Currently using software to develop turbulence closure models for large eddy simulation

Ongoing and Future Work

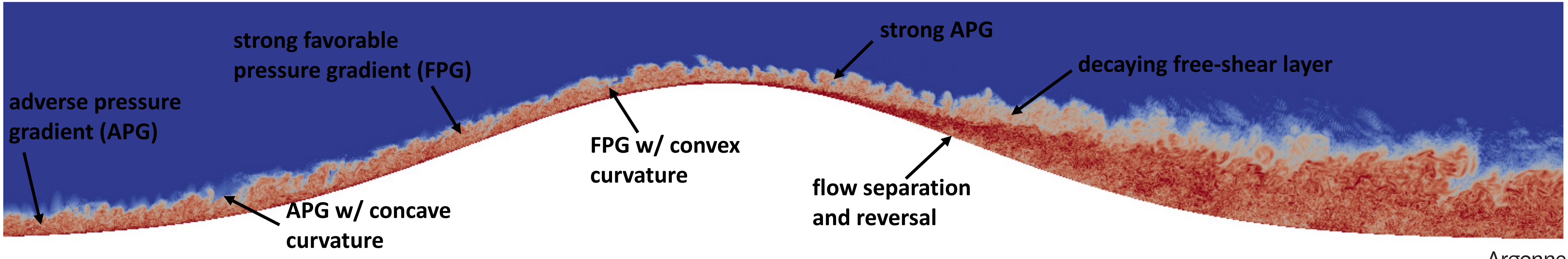
- Incorporating GPU enabled PHASTA into workflow
- Expanding to other applications of ML for turbulence modeling
- Adding user interactivity to online workflow
  - User-dedicated node to run scripts/notebooks evaluating model during training
  - Adding online visualization with Paraview
- Extending to other CFD solvers (Nek5000, NekRS, OpenFOAM)

Questions?

Introduction and Science Goals

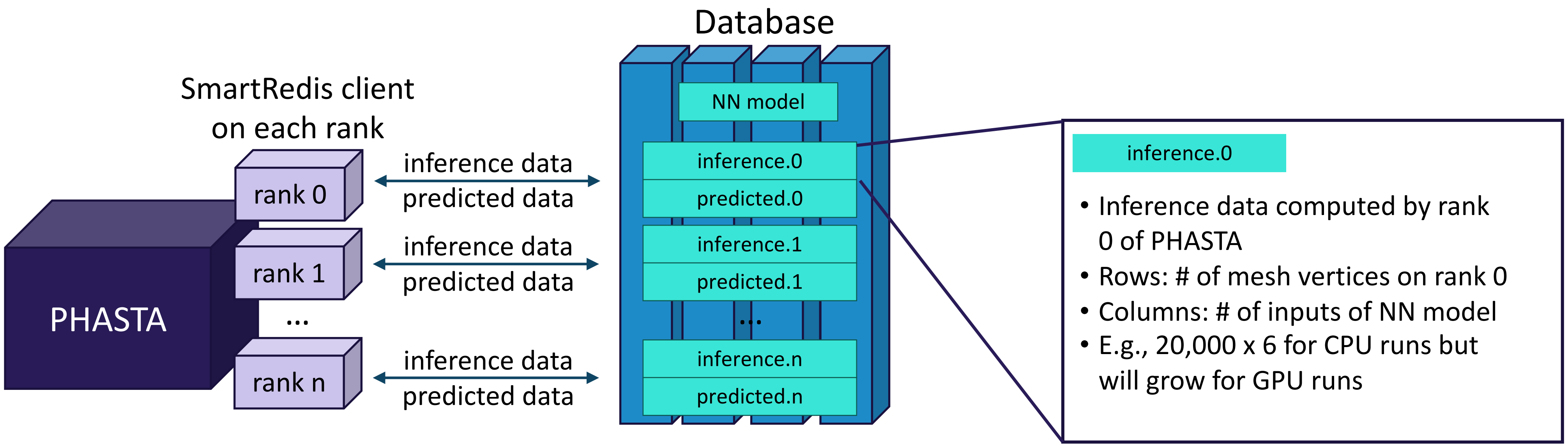
- Why do we need to train on the entire flow domain?
  - A simple bump geometry introduces many physical disturbances that change the turbulence physics
  - A predictive model must be accurate for all these physical effects
- Why do we need to train on multiple snapshots?
  - Single snapshot likely does not contain all time-dependent phenomena that model should handle during inference
  - Some flows of interest are not statistically stationary and evolve over time (e.g., internal combustion engines, active flow control applications, off-design conditions in turbine engines, etc.)

DNS of a Turbulent Boundary Layer Over a Bump



Online Inference with SmartSim

- Evaluation of the NN closure model during simulation
  - Each PHASTA rank sends inference data to database with unique key for its domain partition
  - PHASTA uses SmartRedis API to evaluate model on inference data (using PyTorch jit traced model)
  - Each rank retrieves predictions from database
  - Predictions are used to close equations, build linear system, and advance integration to next time step



Argonne NATIONAL LABORATORY