



Scaling Neural Simulation-Based Inference at High Performance Computing for LHC analysis

Aishik Ghosh, Walter Hopkins, Xiangyang Ju, Nathan Kang

Feb 26, 2026, [Simulation Based Inference Blueprint](#)
CERN

Placing HEP-CCE in Context

[Take from Paolo's slides at IRIS-HEP retreat](#)

HEP-CCE Primary Mission

- Bring computational resources to bear on pressing HEP science problems
- Develop cross-cutting solutions leveraging ASCR and HEP expertise and resources



HEP-CCE Roles

- ASCR-HEP connection (liaison and cross-cutting technical R&D)
 - Future architectures — evolution of ASCR High Performance Computing (HPC) systems and HEP priorities for future systems
 - Long-term engagement with ASCR facilities — Leadership Computing Facility (LCF) allocations and ASCR/HEP “cross-infrastructure” opportunities
 - HEP requirements — What do experiments need from ASCR facilities? What is possible to do on ASCR HPC systems? How do these resources fit within the HEP computing ecosystem?
 - **(new) Act as liaison between HEP and new DOE AI initiatives (AmSC, AI-MP2)**
-

HEP-CCE Technical Areas

[Take from Paolo's slides at IRIS-HEP retreat](#)

Portable Applications and Workflows

Run HEP workflows across multiple HPCs

- Develop and deploy mini-apps and benchmarks representative of accelerated HEP workflows
- Run complex workflows on HPCs, and develop cross-cutting HEP workflow portability solutions

Storage OPTimization

Help address HEP storage challenges

- Map GPU-friendly data models to disk storage.
- Tools for mimicking and characterizing the I/O of HEP workflows.
- Store complete experimental data using RNTuples
- “Intelligent” compression algorithms

Scalable Machine Learning

AI/ML at scale on HPCs

- Tools and methods to design scalable models and train them in parallel
- Training workflows that scale up to 100s of nodes
- Best practices for workflows portability

Accelerating HEP Simulation

MC simulation on GPU

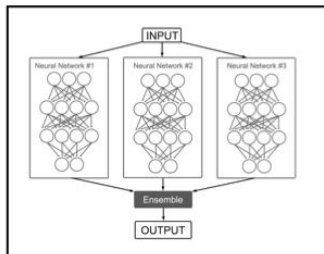
- Next-to-Leading Order event generators on GPUs with controlled precision
- Celeritas: GPU optimized, CPU reproducible, full-fidelity Monte Carlo detector simulation

SML activities

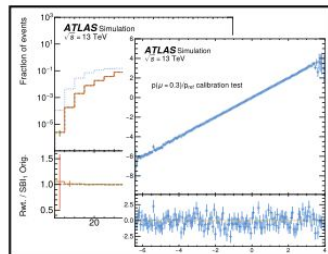
- **Prototyped Inference as a Service (IaaS)**
 - HL-LHC and DUNE are expected to use many more ML algorithms, especially in reconstruction.
 - Set up Triton Server at different HPC sites, starting from Perlmutter
 - Build a shared model registry that enables seamless deployment of Triton Server
 - **Developing a Neural Simulation-Based Inference (NSBI)-oriented, HPC-friendly package.**
 - Involves training an ensemble of many small models ($> 10^4$)
 - [ACAT poster](#), and [code repository](#).
 - Work with analysis groups to have physics result use the common framework
 - **Testing future HEP data formats (RNTuple) for ML training (Exa.TrkX)**
-

A typical NSBI workflow

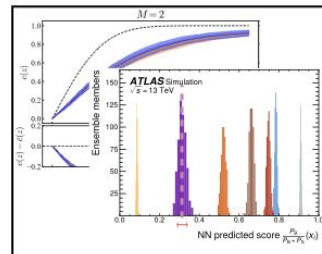
Train ensembles on HPC



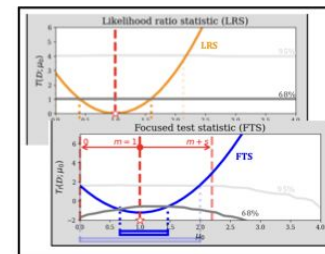
Validation of density estimates



Uncertainty quantification



Neyman construction with pseudo-experiments



Optimize

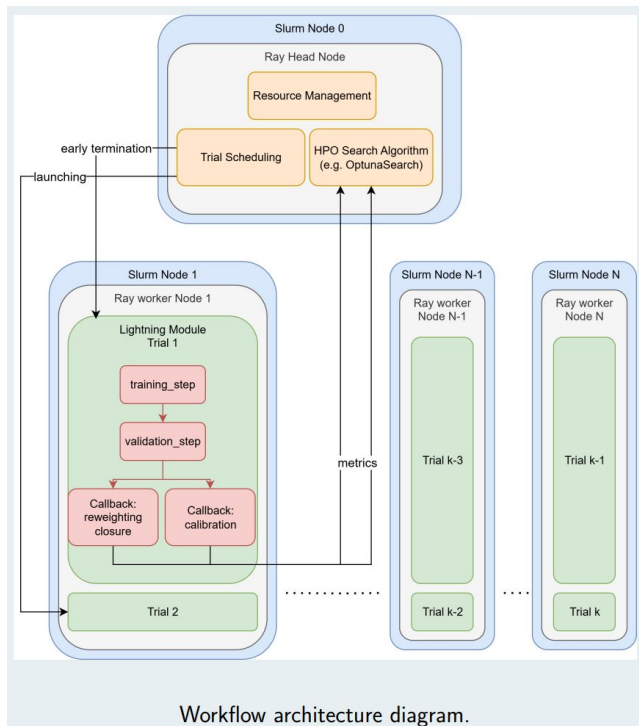
- The NSBI involves multi-step processes and recursive optimization.
- Executing the pipeline is computationally challenging and time consuming.

Computing challenges in NSBI

1. Training thousands of medium-sized neural networks ($\sim 1\text{M}$ parameters) for uncertainty quantification
 2. Validation of high-dimensional probabilities
 - a. Evaluate model performance with different metrics
 3. Hyperparameter optimizations (HPO) using downstream metrics
 - a. Currently it is an iterative feedback-loop
 4. Unbinned fitting with a large number of statistics
-

HPC workflow for distributed HPO

[See our ACAT poster for details.](#)



Ray for trial scheduling & resource management

- **Ray Tune's** scheduler launches trials via Ray Train workers
- Integrated with Slurm for scaling on HPC systems

PyTorch Lightning: Model training & evaluation

- Use callbacks to evaluate NSBI calibration and closure metrics using goodness-of-fit measures such as χ^2 and Wasserstein distances.
- These metrics are reported back to **Ray Tune**.

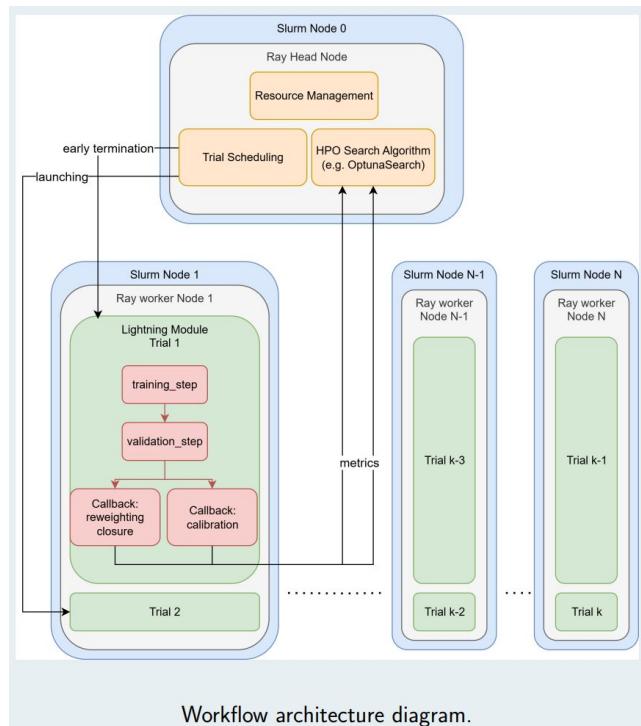
HPC workflow for distributed HPO

[See our ACAT poster for details.](#)

Current: First perform HPO using the NSBI calibration and closure metrics then run ensemble

Future: Run HPO and ensemble at the same time:

- Simultaneously train each model with the same hyperparameters N times
- Groupby the N runs to quantify the systematic uncertainties and feed this information to Ray Tune for HPO search



Workflow architecture diagram.

HPC workflow for distributed HPO

[See our ACAT poster for details.](#)

```
seed: 42
do_hpo_tune: true
stage: "fit"
do_compile: true
compile_kwargs: ...

hpo_tune:
  num_samples: 20
  scheduler:
    _target: ray.tune.schedulers.ASHAScheduler
    max_t: 50
    grace_period: 5
    reduction_factor: 2

  num_workers: 4

  search_space:
    n_layers: "randint:5,15"
    n_nodes: "qrandint:100,1000,4"
    learning_rate: "loguniform:1e-5,2e-4"

  do_ensemble: false
  monitor: ["val_loss", "avg_ws_distance", "calibration_ece", "calibration_mce"]

datamodule: ...

model:
  _target_: nsbi.models.car1.CARL
  n_features: 16
  n_layers: 4
  n_nodes: 100
  learning_rate: 0.001

logger: ...

trainer: ...

callbacks:
  model_checkpoint: ...
  model_checkpoint2: ...
  early_stopping: ...
  closure_metrics: ...
  calibration_metrics: ...
```

YAML-based configuration

- Modular design for each key component: data, ML model, training parameters, HPO search spaces, and so on.

We would like to make the package easily extendable to meet different analysis needs and integrable into other NSBI systems.

Our next step: HPC workflow for NSBI

- Add different ensembling approaches including the WIFI ensemble [arXiv:2506.00113](https://arxiv.org/abs/2506.00113).
- Integrate the uncertainty quantification in the HPO search
- Integrate more downstream tasks:
 - Toy experiments
 - Unbinned fitting for parameter estimation