EESP Workshop SC26 - Chicago, 15 November 2026

OBSERVE: An Observable, Sustainable, and Fault-Tolerant Runtime for Hyperparameter Optimization on Production HPC Systems

Marco De Carlo[1], Gabriele Padovani[2], Gabriele Accarino[3], Valentine Anantharaj[4], Paola Nassisi[1], Sandro Fiore[2]

[1]CMCC Foundation - Euro-Mediterranean Center on Climate Change, Lecce, Italy
[2]Department of Information Engineering and Computer Science, University of Trento, Trento, Italy
[3]Department of Earth and Environmental Engineering, Columbia University, New York, NY, U.S.A.
[4]Oak Ridge National Laboratory, Oak Ridge, TN, U.S.A.

cmcc Centro Euro-Mediterraneo sui Cambiamenti Climatici

Columbia University

Università di Trento

Oak Ridge National Laboratory

EESP Workshop 2026

SC26 hpc unites | Chicago

What OBSERVE gives you on a production HPC system

Before diving into the use cases, here is what the framework produces concretely, across two models and two HPC clusters

~10× less energy
with identical accuracy

~70%
of campaign energy avoidable through 3 runtime signals that tell when to stop

7 KB
Checkpoint weight after 80 iterations, so fault tolerance comes almost for free

LSF → Slurm
Portable state, campaign stopped on JUNO (NVIDIA) and resumed to Frontier (AMD)

HPO today

Training a model means choosing dozens of parameters - Getting them right makes the difference between a good model and a great one

This iterative process is called Hyperparameter Optimization

Run-to-budget

Campaigns stop when the budget runs out, not when they stop improving

Energy-blind

Optimizes accuracy only — energy cost completely ignored

Ephemeral

Campaign die at wall-time limits

Practical implications

- Costly campaigns with no clear criteria for when to stop
- GPU-hours spent after convergence
- Sustainability ignored
- Lost work and manual restarts at every interruption

Existing tools treat HPO as an isolated, fire-and-forget step — disconnected from operational needs

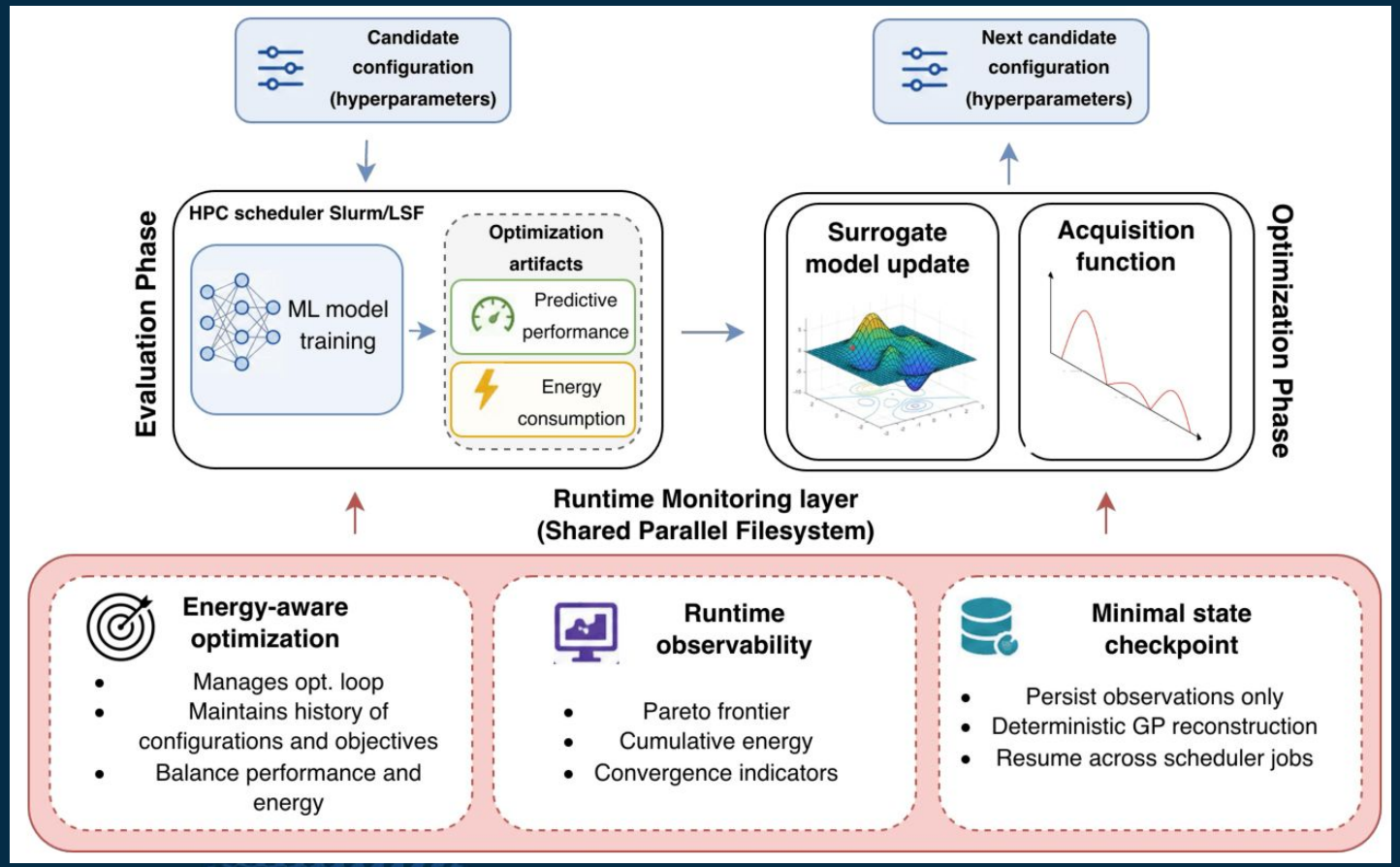
OBSERVE

One job, and the whole optimization loop runs inside its allocation: no external orchestrator, no service outside the job

The optimizer looks for the trade-off between performance and energy

Minimal-state checkpoint, nothing depends on a library version or on the machine

Information written to the file system at each iteration, accessible without relying on any external service



Evaluation setup: two workloads, two machines

ClimateBench

- Climate-model emulation (regression)
- **Objectives:**
  - validation NRMSE ↓
  - energy ↓

MLPerf DeepCAM

- Atmospheric semantic segmentation
- **Objectives:**
  - validation IoU ↑
  - energy ↓

JUNO - CMCC

- Petascale - Intel Xeon + NVIDIA A100 40GB
- Scheduler: IBM Spectrum LSF

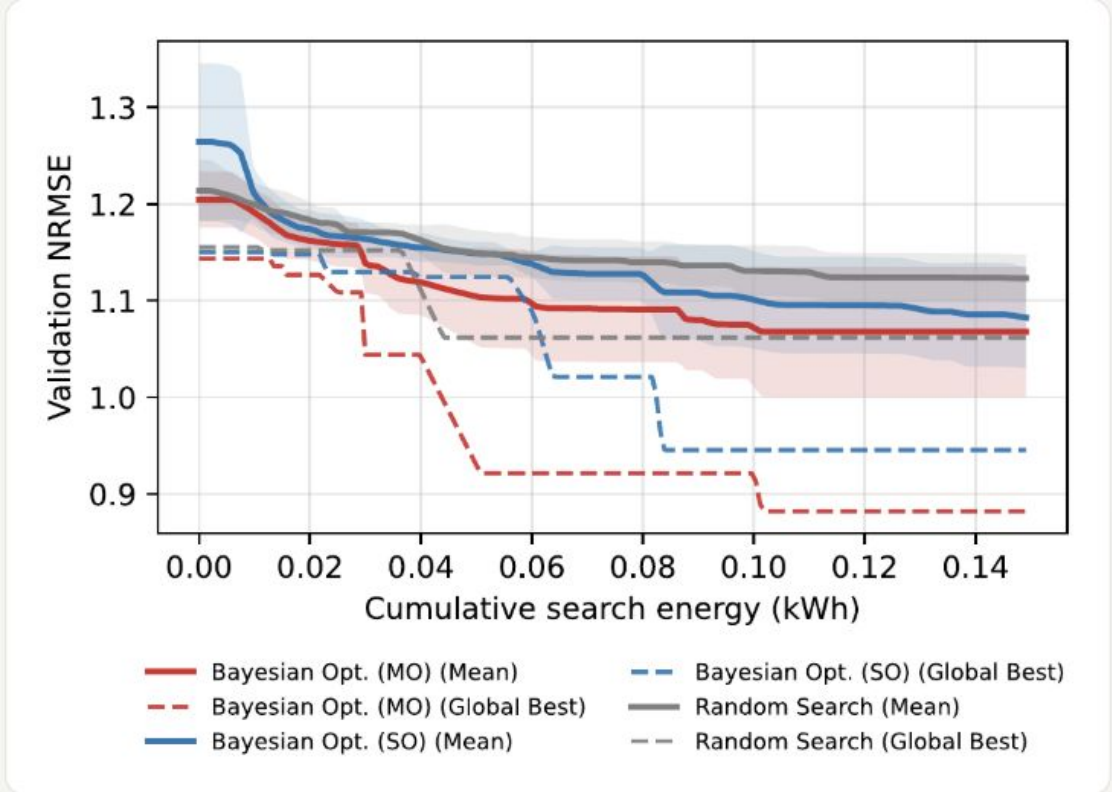
Frontier - OLCF

- Exascale - AMD EPYC + AMD Instinct MI250X
- Scheduler: Slurm

7 hyperparameters - energy through CodeCarbon - up to 64 GPUs

Energy as an objective costs no accuracy

Does adding energy, and all this machinery, cost us optimization quality?

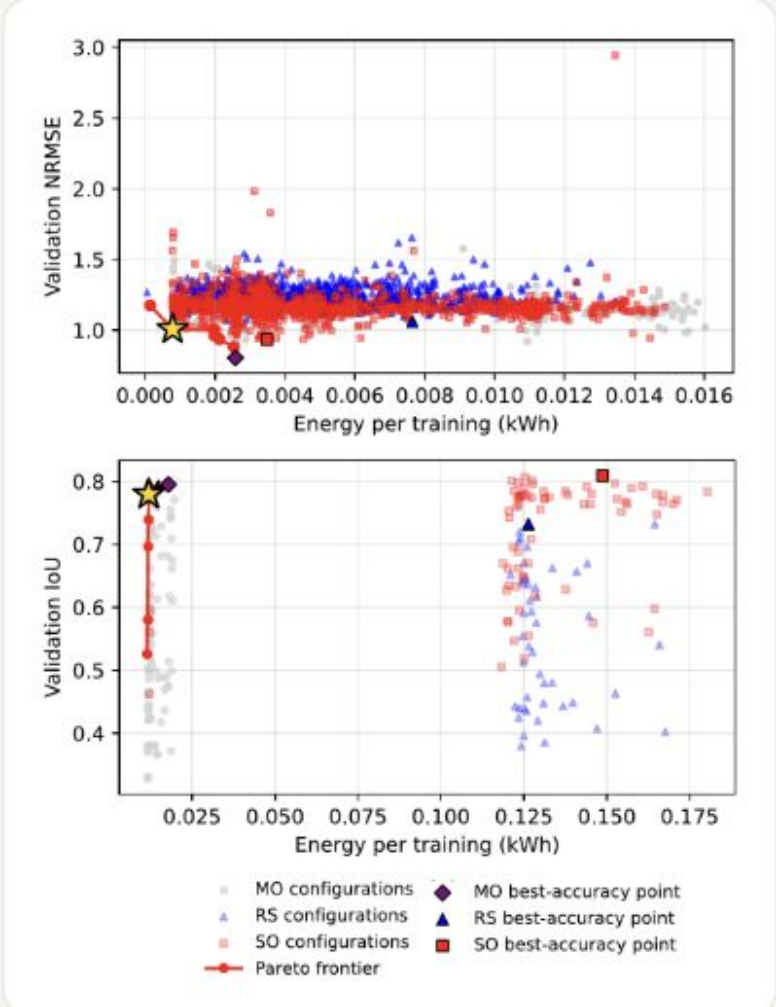


Method	Best NRMSE ↓
Random Search	1.1230 ± 0.0346
BO (SO)	1.0843 ± 0.0743
BO (MO)	1.0677 ± 0.0938

- BO converges faster and lower than Random Search
- The multi-objective run, that cares about energy, is even faster in the early stages of the search in this case

ClimateBench, best validation error against the cumulative energy spent by the search over 10 seeds (Fig. 2, Table IV)

Pareto fronts: same quality, far less energy



> Because energy is an objective, you get a front

A set of operating points, not one optimum

Each point is a configuration we trained. Multi-objective BO returns the whole energy-quality front; the knee point is a practical compromise

DeepCAM: the gap is largest

The effect is strongest, the single objective optimum spends about an order of magnitude more energy per training for almost no gain in IoU

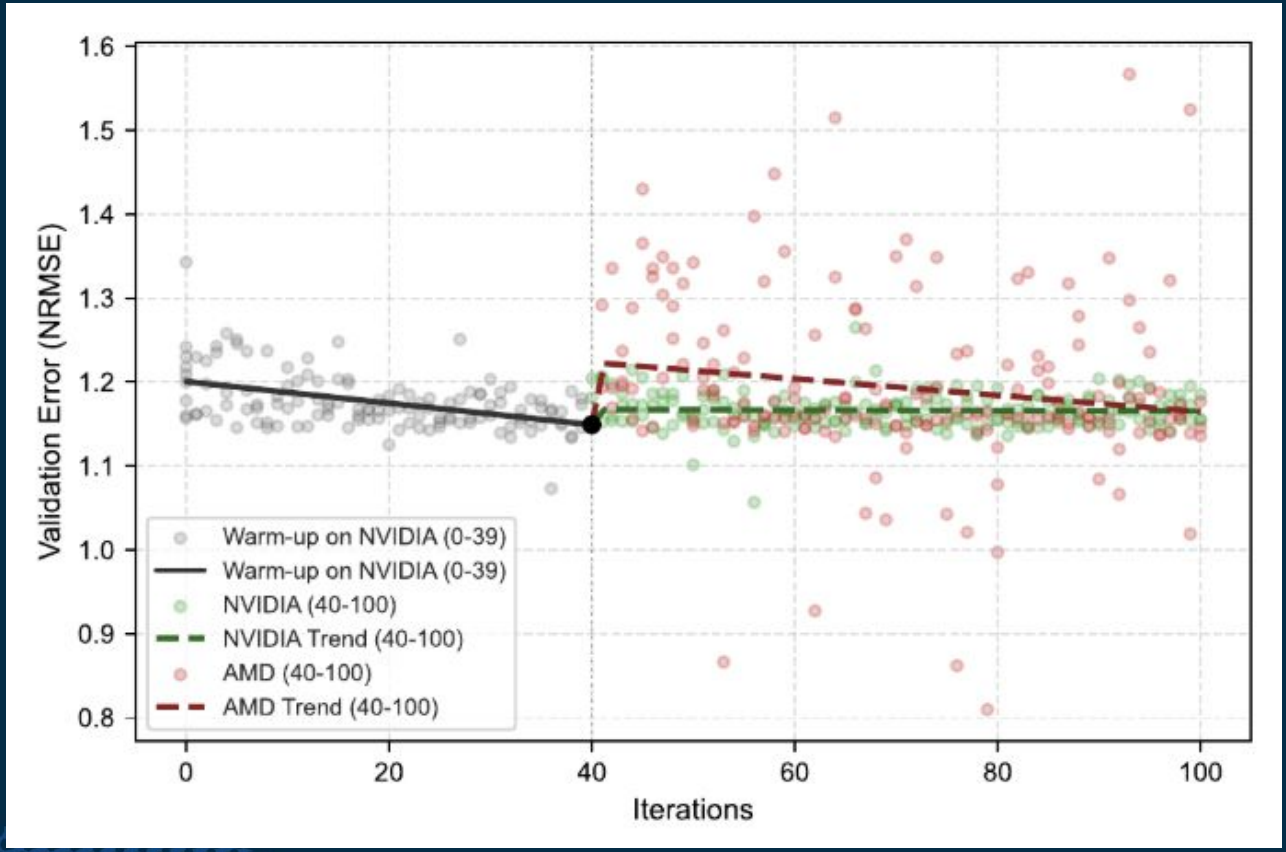
The front is a runtime artifact

Stored at every iteration, so its evolution can be watched while the campaign runs.

ClimateBench (top) and DeepCAM (bottom); stars mark the knee point (Fig. 3)

Resume across jobs and across machines

Fault tolerance: the next job reads the file, refits the surrogate and continues



- Checkpointed after 40 evaluations
- Checkpoint size: 4.4 KB (Optuna: 128–196 KB + external DB)
- Save time: 10 ms
- Restoring time: 2.64 s

Unchanged core logic; only scheduler scripts and CodeCarbon settings adapted

The trajectory just continue on NVIDIA, a small transient shift on AMD

Same convergence: the surrogate is refitted on the new observations and absorbs that shift

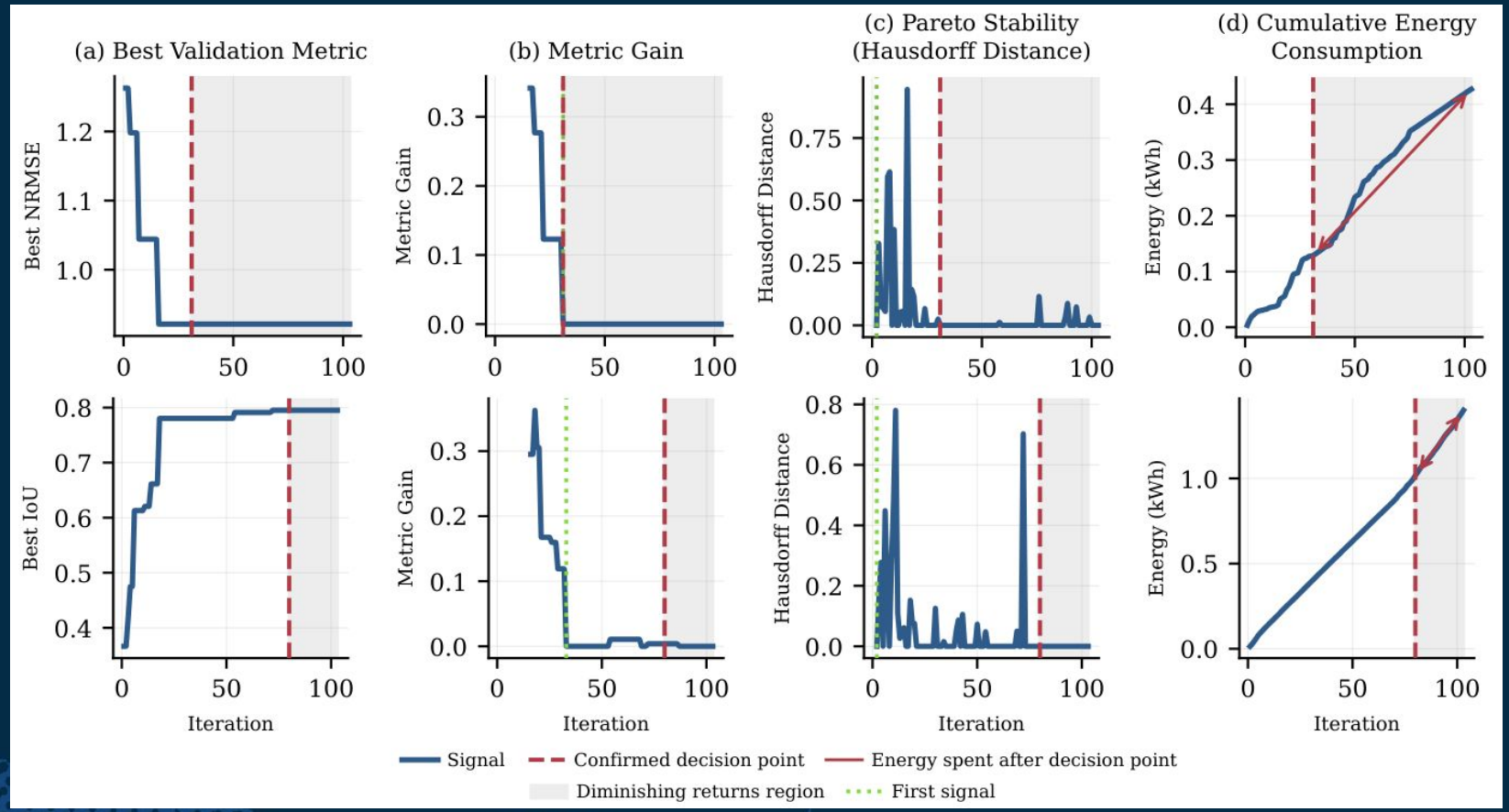
ClimateBench, stopped after 40 evaluations, JUNO → Frontier (Fig. 4, Tables V and VIII)

Runtime indicators show when to stop paying

Same pattern for both workloads: gain and front stability flatten early, while energy keeps climbing

ClimateBench (Watson-Parris et al., 2022)

DeepCAM (Farrell et al., 2021)



3 signals

What we expose while the search runs: best results so far, how much is still improving, and how stable the trade-offs are

ClimateBench - 61.7% saved

Energy avoidable by stopping at iteration 31; only +0.0014 NRMSE left

DeepCAM - 27.3% saved

Of campaign energy spent after the decision point at iteration 80, for no improvement

Decision point: best objective, Metric Gain and Hausdorff distance all stable over 15 iterations (Fig. 6, Table VI)

Conclusions

✔ Conclusions

Energy as a BO objective

keeps search quality and exposes cheaper Pareto-optimal configurations

Checkpointing

An observation-only checkpoint (KB, ms) survives wall-time limits and moves across HPC systems

Early stopping signals

Filesystem-based runtime indicators let users stop when more evaluations no longer pay off

⚠ Future Work

Adaptive policies

That act on the indicators to allocate budget

More informative objectives

Carbon-intensity-aware objectives

Acquisition function

Integrated scheduler- and resource-aware HPO with an energy-aware acquisition function

Thank you!

De Carlo, M., Padovani, G., Accarino, G., Nassisi, P., & Fiore, S. (2026). MLOps+: Enhancing the MLOps lifecycle with traceable, explainable, and energy-aware Bayesian optimization. Future Generation Computer Systems, 189, 108883. doi:10.1016/j.future.2026.108883

cmcc Centro Euro-Mediterraneo sui Cambiamenti Climatici

Columbia University

Università di Trento

Oak Ridge National Laboratory

EESP Workshop 2026

SC26 hpc unites | Chicago

Code available on request, public release coming soon - marco.decarlo@cmcc.it