Simulations and AI

© 2026 Hewlett Packard Enterprise Development LP

Where might machine learning play a role with Simulation

Completely replace a simulation

- AI model learns to produce output by observing simulation inputs/outputs

Use simulation as one input to a machine learning model

- For example, use an ML model to account for location-specific history (weather)

Replace modeling of physical processes or parameterised models with machine learning, Examples..

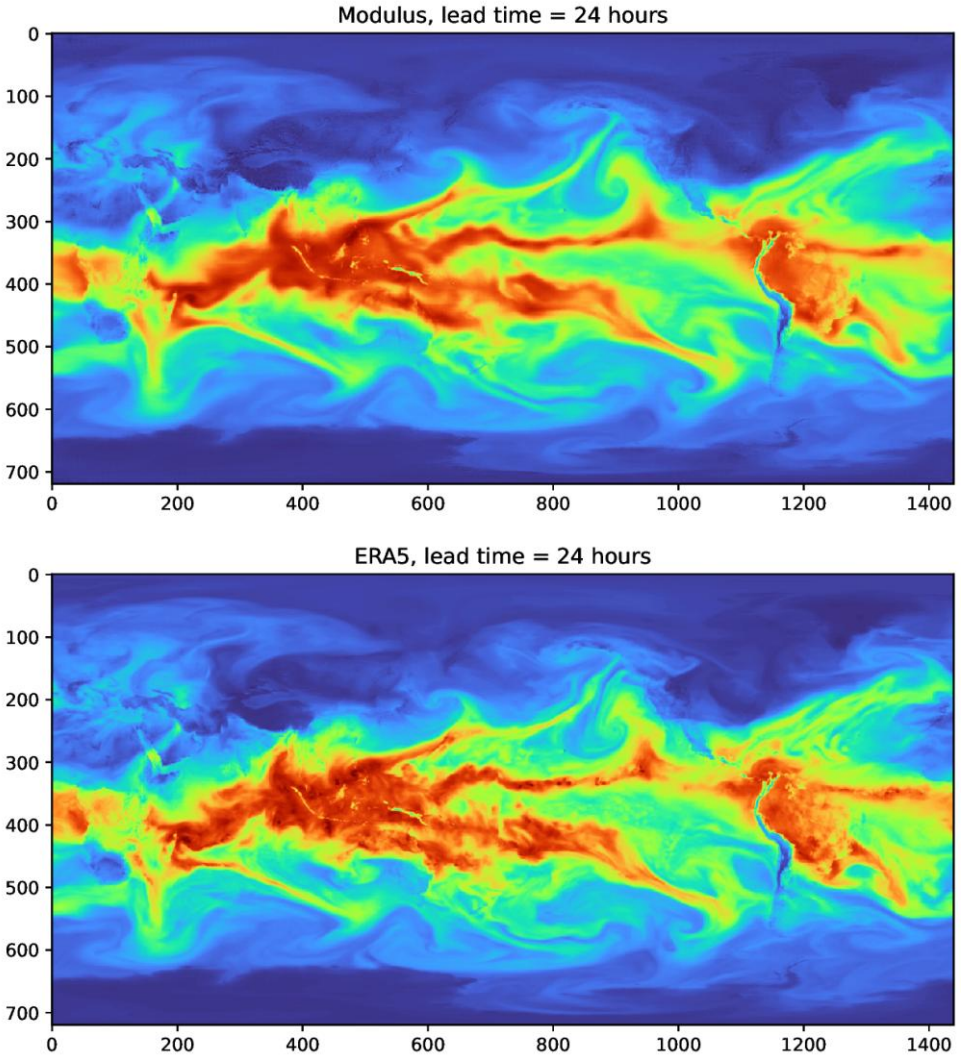
- Reduce search space for Drug discovery
- A turbulence model in an ocean simulation
- Particle physics: particle tracks
- Atomic potentials (computational chemistry)

We will concentrate on the case where AI is **coupled** with simulation

© 2026 Hewlett Packard Enterprise Development LP

Why HPC and AI instead of HPC vs. AI?

- **Can AI replace numerical-based approaches?**
  - No, observable data is too sparse and/or expensive to collect
- **Benefits of AI models**
  - Can be run more quickly than traditional numerical models
  - Simple to run, does not need complicated software infrastructure and HPC resources
  - Skillful models are lower-order representations of 'true' simulation
    - Useful for exploring parameter space/uncertainties
- **Downsides of AI models**
  - How do you add process complexity?
  - Can they extrapolate beyond the data they have been trained on?
  - *Rely on the training data for improvement*
- **Challenges to combining HPC&AI**
  - Numerical: How can you characterize the stability and accuracy of an ML model in that context
  - Technical:
    - How do you connect Fortran/C/C++ codebases to ML packages?
    - How do you appropriately balance high-value/cost GPU resources in predominantly CPU-based code?



https://docs.nvidia.com/deeplearning/modulus/modulus-sym/user_guide/neural_operators/fourcastnet.html#introduction

© 2026 Hewlett Packard Enterprise Development LP

Combining AI/ML with scientific simulation

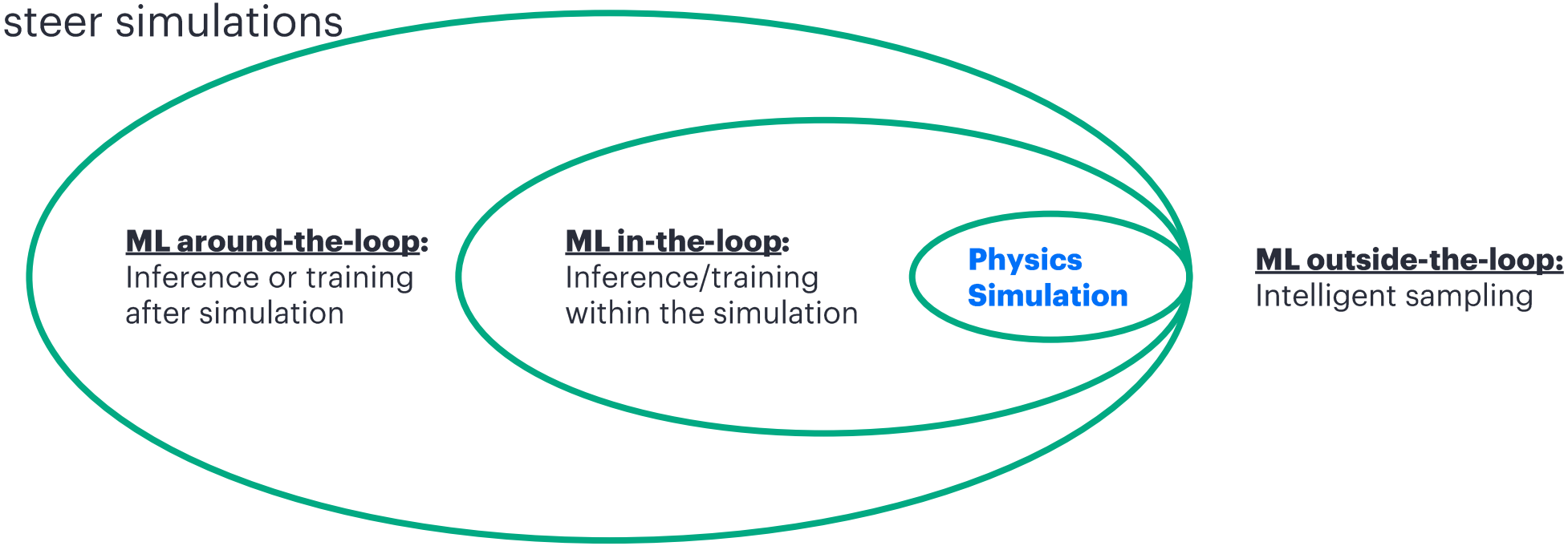
Combining AI software and traditional HPC applications at different levels of a workflow unlocks innovative solutions

ML around-the-loop

- Automatic parameter tuning
- New data assimilation techniques
- Learn to steer simulations

ML in-the-loop

- Embedding machine-learning predictions within numerical solvers
- On-the-fly analysis and visualization



© 2026 Hewlett Packard Enterprise Development LP

Challenges and approaches

Machine learning frameworks are invariably accessed via Python

HPC simulation is most likely written in C/C++/Fortran

We could implement ML in our simulation language but…

- A lot of work for something likely already done and likely more efficiently than you will
- We don’t get access to tools to train models
- Hard to integrate a model externally developed

Some approaches:

Use language-interoperability to interface between simulation and Machine Learning

Couple ML components to our simulation (sockets, messaging transports, files)

Use a framework designed to provide such interoperability (via network transport or APIs)

- Fortran Keras Bridge
- SmartSim

© 2026 Hewlett Packard Enterprise Development LP

Language Interoperability

Interoperability by calling conventions

- Fortran and Python
  - f2py and fmodpy or forpy can help build wrappers to call Fortran from Python
  - ISO C bindings on the Fortran side interfaced to ctypes/Cython on the python side
  - Really helpful if what you are interfacing to has direct support for the Numpy C API
- C++/C and Python
  - Cython, pybind11, SWIG

Interoperability at Framework Level (Fortran)

- Directly call Tensorflow or Torch APIs from Fortran using ISO C interoperability
  In both cases you may have to save model in a special format

Alternatively

- Communicate workflow components via filesystem or network

© 2026 Hewlett Packard Enterprise Development LP

What is SmartSim?

© 2026 Hewlett Packard Enterprise Development LP

A paradigm shift

Old Numerical Workflows

Input data

- Set of initial conditions
- File representing geometry
- Hard-coded values

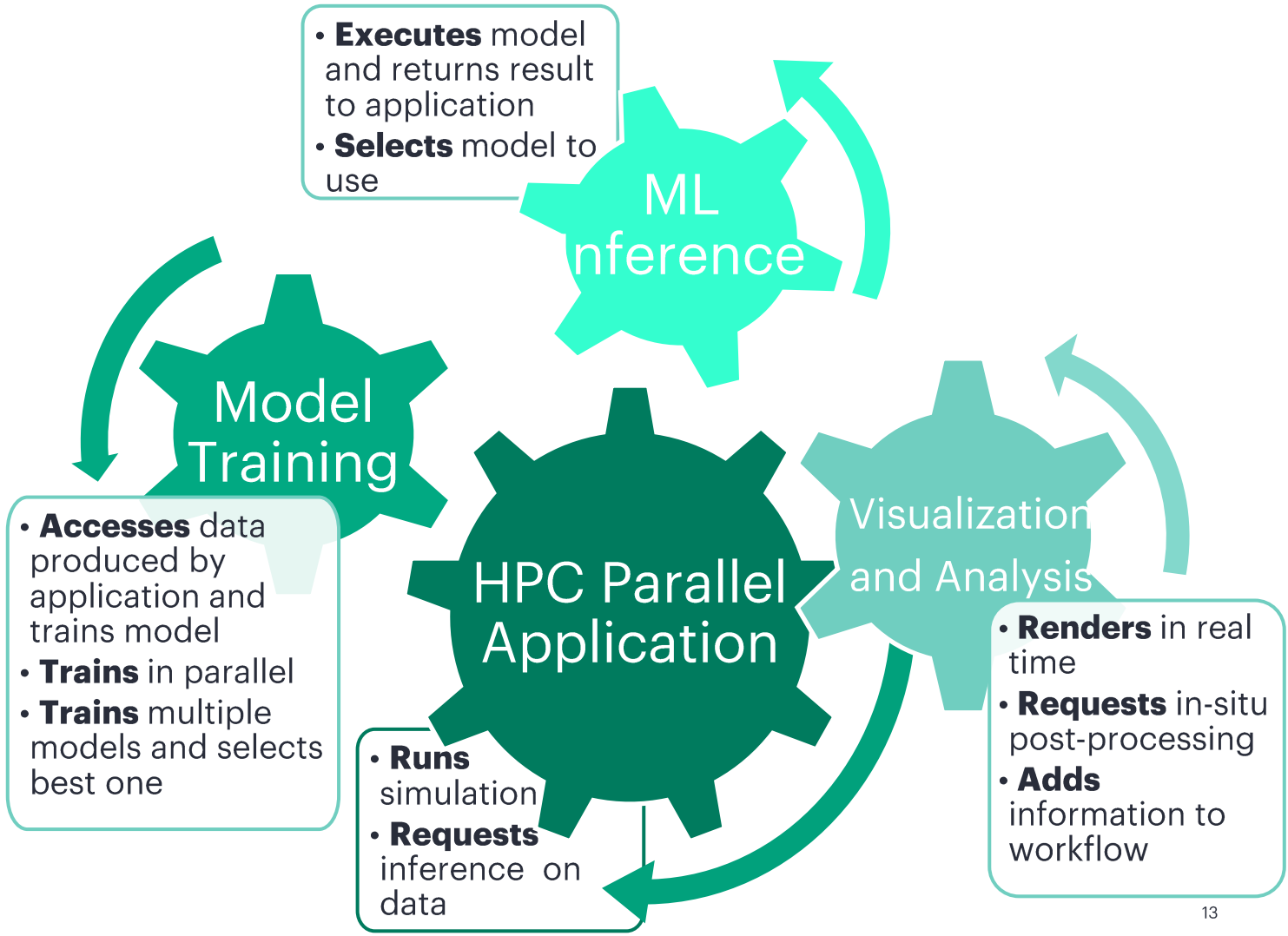
Monolithic Application

- HPC native
- Parallel C/C++/Fortran
- Contains all needed logic

Output data

- Stored on filesystem
- Visualized or analyzed

New AI-Enhanced Numerical Workflows



© 2026 Hewlett Packard Enterprise Development LP

About SmartSim

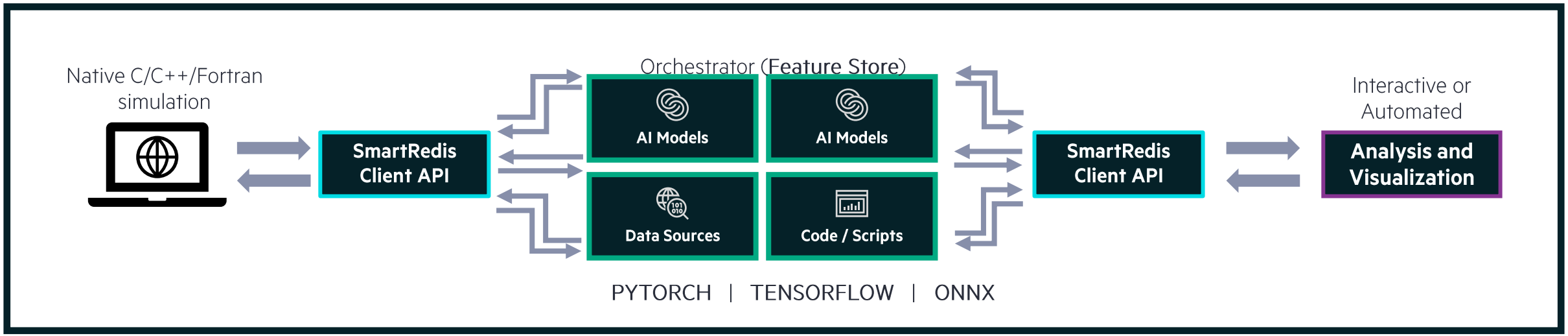
SmartSim is an open-source library

- **bridging** the divide between traditional numerical simulation and data science
- providing a **loose-coupling** philosophy for combining HPC & AI

SmartSim enables simulations to be used as engines within a system, producing data, consumed by other services to create new applications

- Use Machine Learning (ML) models in existing Fortran/C/C++ simulations
- Exchange data between C, C++, Fortran, and Python applications
- Train ML models and make predictions using TensorFlow, PyTorch, and ONNX
- Analyze data streamed from HPC applications while they are running

All of these can be done *without touching the filesystem*



© 2026 Hewlett Packard Enterprise Development LP

Opportunities enabled by SmartSim

SmartSim allows users to

- **Setup**, configure, **run**, and monitor **simulations from** a **Python** script or notebook
- **Embed machine-learning** and advanced **data-analysis** techniques into C/C++/Fortran **simulations**
  - Combine rapidly-evolving data science software stack with slower-moving simulation development
- **Exchange data at-scale** between various entities **without** using the **filesystem**

High performance **database-focused** architecture

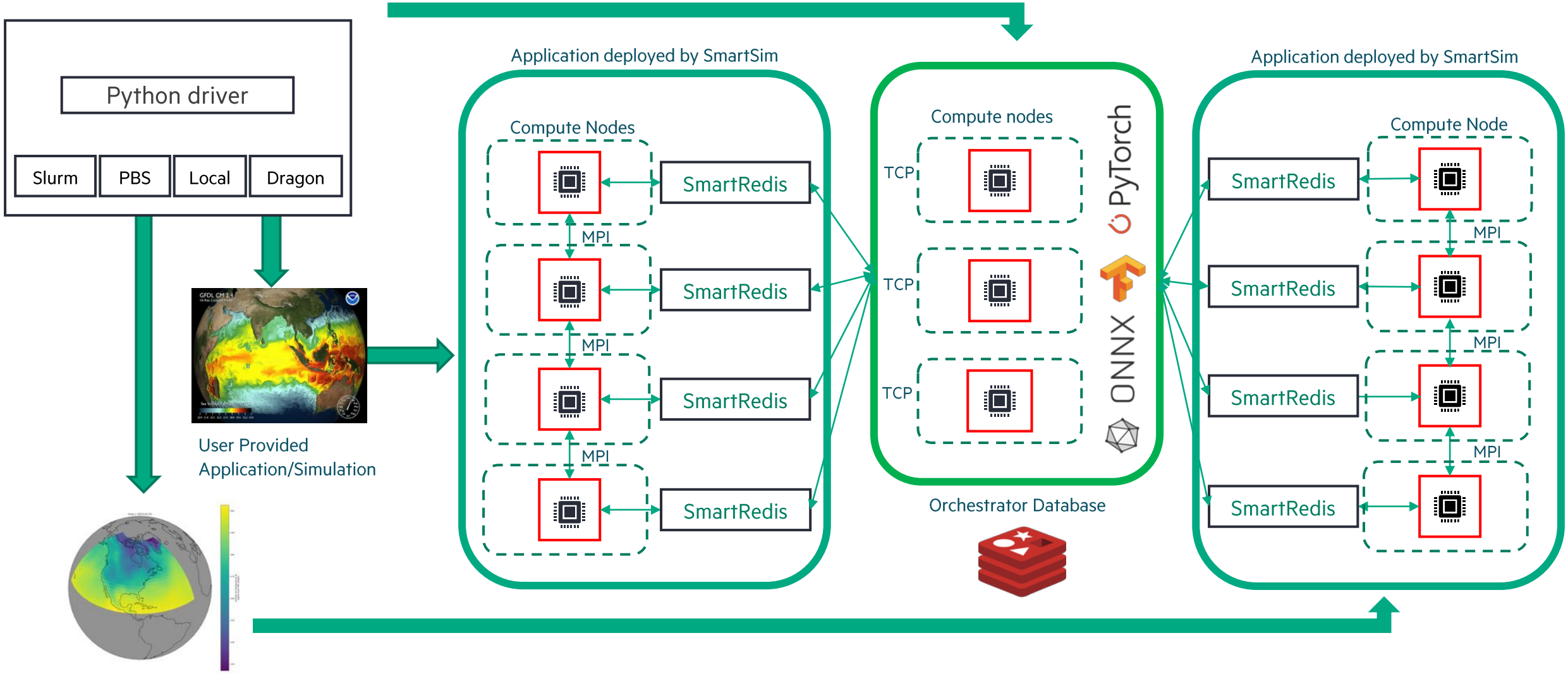
- **Scale** easily to **user needs** (memory, GPU, etc.) by adding additional shards
- **Efficiently use GPU hardware** by concentrating computations (better utilization)
- **Reduce storage** requirements by enabling **data-in-motion**
  - Analysis, visualization, training, and prediction as the simulation is running

Create **AI-enabled**, numerical simulation **ecosystems**

- Modest development effort to build new classes of applications from existing simulations
- Automatic shape design, parameter estimation, reinforcement learning, data assimilation

© 2026 Hewlett Packard Enterprise Development LP

Typical deployment



© 2026 Hewlett Packard Enterprise Development LP

SmartSim practical usage example

© 2026 Hewlett Packard Enterprise Development LP

SmartSim helloworld example

```
from smartsim import Experiment
exp = Experiment(name="hello-world", launcher="slurm")

run_settings = exp.create_run_settings(exe="echo", exe_args="hello!", run_command="srun")


run_settings.set_nodes(1)
run_settings.set_cpus_per_task(1)
run_settings.set_tasks(2)
run_settings.set_tasks_per_node(2)



 # run the model
M1 = exp.create_model("tutorial-model", run_settings=run_settings)
exp.start(M1, block=True, summary=True)
```

```
salloc -N 1 python helloworld.py
```

Interactive launch to use compute nodes

© 2026 Hewlett Packard Enterprise Development LP

SmartSim helloworld batch launch example

```
from smartsim import Experiment
exp = Experiment(name="hello-world", launcher="slurm")
run_settings = exp.create_run_settings(exe="echo", exe_args="hello!", run_command="srun")

run_settings.set_nodes(1)
run_settings.set_cpus_per_task(1)
run_settings.set_tasks(2)
run_settings.set_tasks_per_node(2)


# batch settings
sbatch_settings = exp.create_batch_settings(nodes=1, time="00:10:00", queue="test", account=[XXX])

 # run the model
M1 = exp.create_model("tutorial-model", run_settings=run_settings, batch_settings=sbatch_settings)
exp.start(M1, block=True, summary=True)
```

```
python helloworld.py
```

SmartSim itself launches batch jobs

© 2026 Hewlett Packard Enterprise Development LP

SmartSim Orchestrator

The orchestrator is an in-memory database that uses Redis and RedisAI to provide a distributed database and access to ML models.

Stores the tensors, the machine learning model and scripts.

Holds and executes ML models written in python on CPU or GPU.

```
from smartsim import Experiment

exp = Experiment("test", launcher="slurm")
db = exp.create_database(db_nodes=1,
                         db_port=6780,
                         batch=False,
                         interface="hsn0")
exp.start(db)

print(f"Orchestrator launched on nodes: {db.hosts}")

exp.stop(db)
```

© 2026 Hewlett Packard Enterprise Development LP

MOM6 example

Ocean turbulence modeling doing inference every timestep to predict the eddy kinetic energy (EKE).

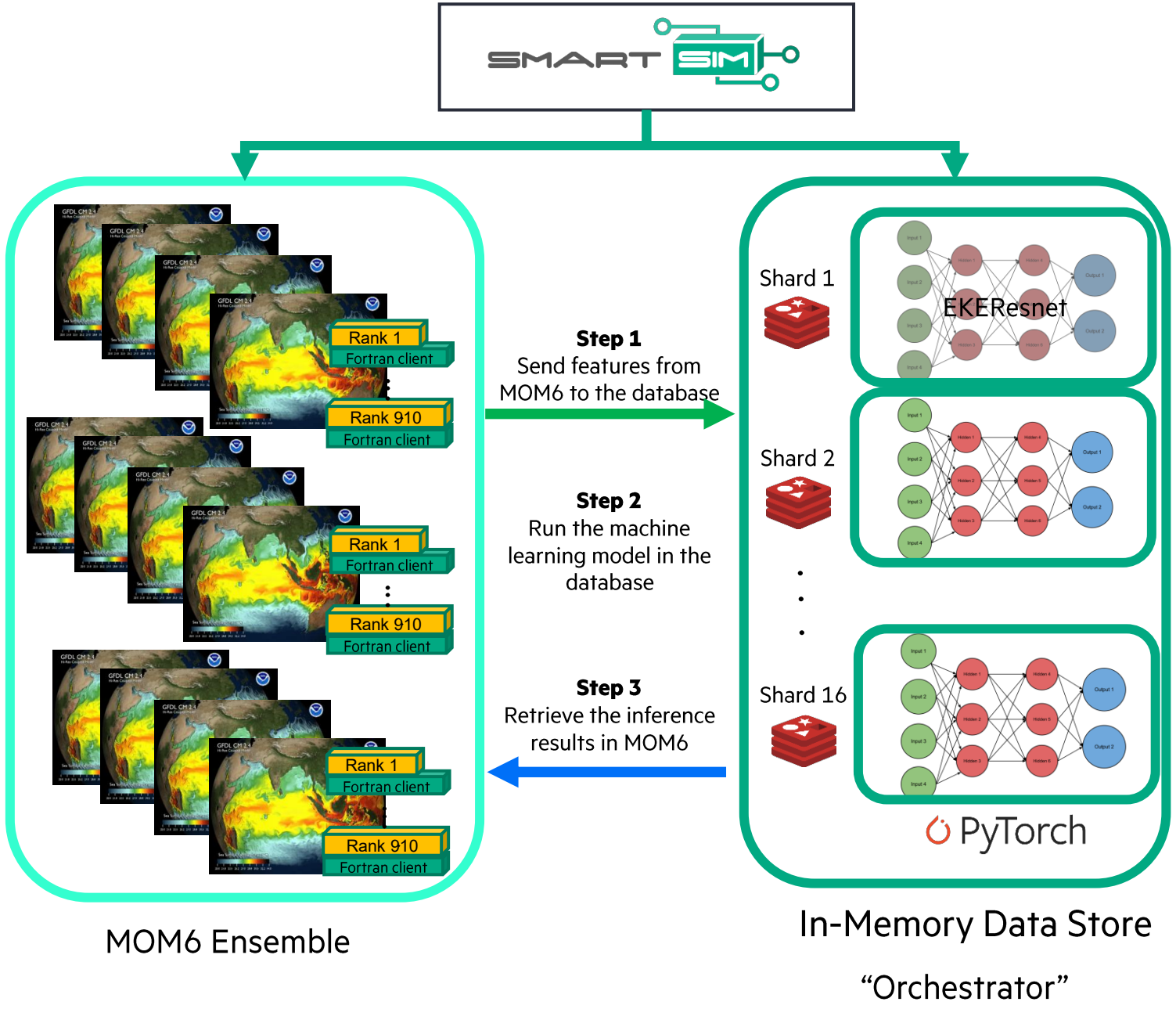
MOM6 application runs on CPUs.

Database can be run on CPU or GPU (using relevant pytorch).

To use:

— Modify the application code to use SmartRedis

— Create a SmartSim python driver script which will then launch the application and the database using the job scheduler.



© 2026 Hewlett Packard Enterprise Development LP

MOM6 application modifications

Add in calls to SmartRedis into the application (~10 lines of code)

— Start the Client, put and unpack tensors, and run the ML model.

```
client = smartredis_CS%client

..

sr_return_code = client%put_tensor("features"//CS%key_suffix, CS%features_array, shape(CS%features_array))
model_out(1) = "EKE"//CS%key_suffix
model_in(1) = "features"//CS%key_suffix
sr_return_code = smartredis_CS%client%run_model(CS%model_key, model_in, model_out)

..

sr_return_code = client%unpack_tensor( model_out(1), CS%MEKE_vec, shape(CS%MEKE_vec) )

..
```

© 2026 Hewlett Packard Enterprise Development LP

Driver script – create ensemble

The driver_OM4_025.py script is used to launch the orchestrator and the MOM6 experiment ensemble.

Key settings:

- Experiment: Top level object that provides factory methods to create workflow objects.
- Batch settings: Can be used if application is to be launched non-interactively (queues and walltime).
- Run settings: Describes system-specific resources (nodes, tasks) and the application executable.

```
ensemble_batch_settings = experiment.create_batch_settings(
      nodes     = ensemble_size*nodes_per_member,
      queue.    = ensemble_queue,
      time      = walltime,
      batch_args = mom6_batch_args
)


mom6_run_settings = experiment.create_run_settings(mom6_exe_path)
mom6_run_settings.set_tasks_per_node(tasks_per_node)
mom6_run_settings.set_tasks(nodes_per_member*tasks_per_node)


mom_ensemble = experiment.create_ensemble(
   "MOM",
   batch_settings = ensemble_batch_settings,
   run_settings   = mom6_run_settings,
   replicas       = ensemble_size
)
```

© 2026 Hewlett Packard Enterprise Development LP

Driver script – launch database

- Sets up the orchestrator which stores the tensors and executes the ML models.
- Can create the database with the specified port, network interface, no. of nodes.
- Can set batch and run settings (if needed).
- Experiment is then generated.

```
orchestrator = exp.create_database(
  port = orchestrator_port,
  interface = orchestrator_interface,
  db_nodes = orchestrator_nodes,
  time = walltime,
  threads_per_queue=8,
  batch=True)


orchestrator.set_cpus(8)
orchestrator.set_batch_arg("partition", "mult")
orchestrator.set_batch_arg("cpus-per-task", "8")
orchestrator.set_batch_arg("exclusive",None)
```

© 2026 Hewlett Packard Enterprise Development LP

MOM6 job script (generated by SmartSim)

```
#!/bin/bash


#SBATCH --output=/path/to/exp/AI-EKE-MOM6/MOM/MOM.out
#SBATCH --error=/path/to/exp/AI-EKE-MOM6/MOM/MOM.err
#SBATCH --job-name=MOM-DA6XOTVLPEEM
#SBATCH --partition=test
#SBATCH --tasks=64
#SBATCH --cpus-per-task=1
#SBATCH --gres=tmp:200G
#SBATCH --mem=200G
#SBATCH --nodes=1
#SBATCH --time=00:15:00

cd /path/to/exp/AI-EKE-MOM6/MOM/MOM_0 ; srun --output /path/to/exp/AI-EKE-MOM6/MOM/M0M_0/MOM_0.out --error /path/to/exp/AI-EKE-
MOM6/MOM/MOM_0/MOM_0.err --job-name MOM_0-DA6XOTVLRJJ9 --export
ALL,SSDB=172.18.18.59:6780,SR_DB_TYPE=Standalone,SSKEYIN=MOM_0,SSKEYOUT=MOM_0 --mpi=pmix --ntasks-per-node=64 --ntasks=64
/path/to/MOM6/MOM6
```

SSDB: Address of redis database and port

SSKEYIN: the prefix attached to tensors, datasets, models, etc. retrieved from the database

SSKEYOUT: the prefix that is attached to tensors, datasets, etc. sent from client to database

© 2026 Hewlett Packard Enterprise Development LP

Orchestrator job script (generated by SmartSim)

```
#!/bin/bash


#SBATCH --output=/path/to/exp/AI-EKE-MOM6/orchestrator/orchestrator.out
#SBATCH --error=/path/to/exp/AI-EKE-MOM6/orchestrator/orchestrator.err
#SBATCH --job-name=orchestrator-DA6XOOS6DNQ6
#SBATCH --nodes=1
#SBATCH --time=00:15:00
#SBATCH --cpus-per-task=64
#SBATCH --partition=multi
#SBATCH --exclusive
source /path/to/pyenv/bin/activate

cd /path/to/exp/AI-EKE-MOM6/orchestrator ; srun --output /path/to/exp/AI-EKE-MOM6/orchestrator /orchestrator_0/orchestrator_0.out --error /path/to/exp/AI-EKE-MOM6/orchestrator /orchestrator_0/orchestrator_0.err --job-name orchestrator_0-DA6XOOS6FHZS --ntasks=1 --ntasks-per-node=1 --cpus-per-task=64 --mpi=pmix /path/to/python -m smartsim._core.entrypoints.redis +orc-exe /path/to/smartsim/_core/bin/redis-server +conf-file=/path/to/smartsim/_core/config/redis.conf +rai-module --loadmodule /path/to/smartsim/_core/lib/redisai.so THREADS_PER_QUEUE 64 +name=orchestrator_0 +port=6780 +ifname=ib0
```

Uses the Smartsim python environment

-m: Redis python module
+orc-exe: Execute redis server
+conf-file: redis configuration file (redis.conf)
+rai-module: Redis AI module library

THREADS_PER_QUEUE: threads for the DB
+name: orchestrator name
+port: specified port no
+ifname: the network interface DB should listen on

© 2026 Hewlett Packard Enterprise Development LP

Driver script launch

The driver script launches batch jobs and manages execution of the components.

The key settings are printed out.

The driver continually polls the jobs to check their status.

```
python driver_OM4_025.py clustered
13:15:25 login2.hpc.ac.uk SmartSim[3034710:MainThread] INFO

=== Launch Summary ===
Experiment: AI-EKE-MOM6
Experiment Path: /nobackup/hpcday1/MOM6/NCAR_ML_EKE/driver/AI-EKE-MOM6
Launcher: slurm
Database Status: launching

=== Ensembles ===
MOM
Members: 1
Batch Launch: Batch Command: sbatch
Batch arguments:
partition = test
tasks = 64
cpus-per-task = 1
gres = tmp:200G
mem = 200G
nodes = 1
time = 00:15:00

=== Database ===
Shards: 1
Port: 6780
Network: ['ib0']
Batch Launch: Batch Command: sbatch
Batch arguments:
nodes = 1
time = 00:15:00
cpus-per-task = 64
partition = multi
exclusive = None

13:15:26 login2.hpc.ac.uk SmartSim[3034710:MainThread] INFO Orchestrator launched as a batch
13:15:26 login2.hpc.ac.uk SmartSim[3034710:MainThread] INFO While queued, SmartSim will wait for Orchestrator to run
13:15:26 login2.hpc.ac.uk SmartSim[3034710:MainThread] INFO CTRL+C interrupt to abort and cancel launch
13:15:42 login2.hpc.ac.uk SmartSim[3034710:MainThread] INFO MOM(12919350): SmartSimStatus.STATUS_NEW
13:15:47 login2.hpc.ac.uk SmartSim[3034710:MainThread] INFO MOM(12919350): SmartSimStatus.STATUS_RUNNING
13:15:52 login2.hpc.ac.uk SmartSim[3034710:MainThread] INFO MOM(12919350): SmartSimStatus.STATUS_RUNNING
```

```
  JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
12919349     multi orchestr    user1  R       0:28      1 cn001
12919350      test MOM-DA6X    user1  R       0:17      1 cn002
```

© 2026 Hewlett Packard Enterprise Development LP

Thank You

© 2026 Hewlett Packard Enterprise Development LP
© 2026 Hewlett Packard Enterprise Development LP