Themis Ensemble Manager

Presented To: WoWoHa Seminar

June 2020

David Domyancic, James Corbett, Andrew Fillmore, Keith Healy, and Richard Klein

Lawrence Livermore National Laboratory

LLNL-PRES-811656

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

Abstract

We present a new ensemble of simulations generator named **Themis**. Themis leverages a simulation submission batch script to create an ensemble with minimal setup time. Themis can be used to generate simple parameter studies, which can be scaled to million member studies, or to generate complex design optimization workflows or machine learning workflows such that users can create **dynamic and adaptive optimization loops** using straightforward Python scripting. Themis has an easy-to-use command line interface for fast study generation, and a Python API for building complex workflows.

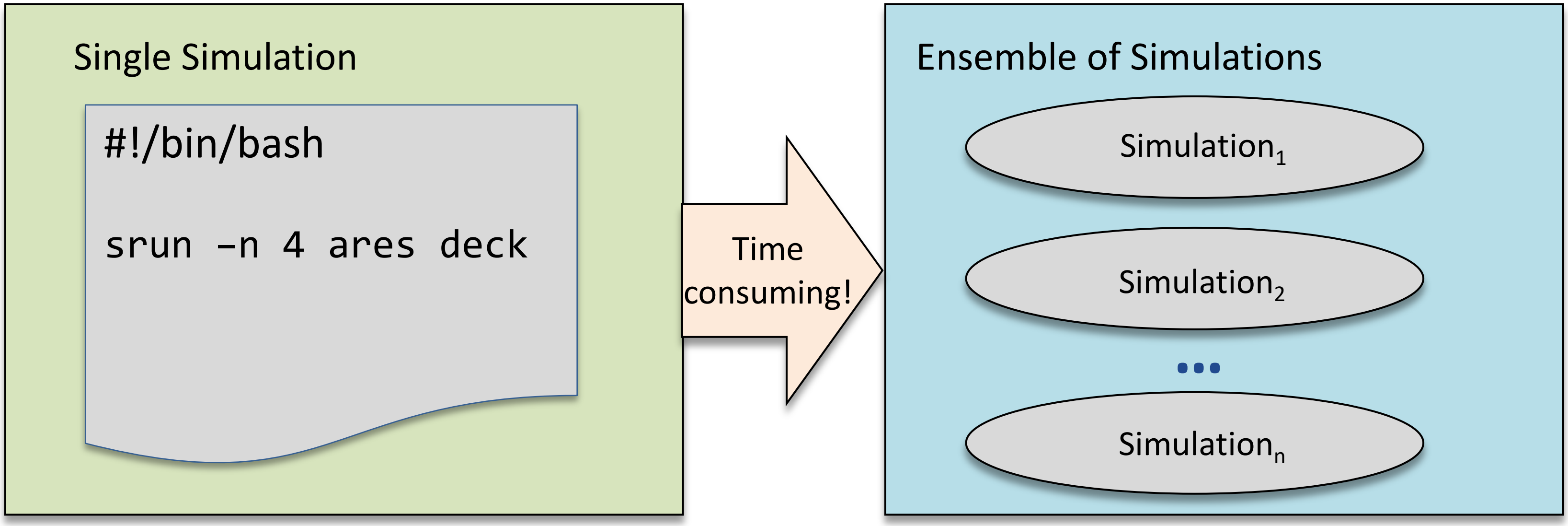
We will demonstrate how to evolve a batch submission script, which runs a single simulation, to a study using the Themis **command line interface**. Themis' CLI allows users to:

- generate studies
- dry-run studies
- report study status
- kill/restart of individual simulations
- harvest simulation outputs

We will also show how Themis's **Python API** can be used to build **a dynamic optimization workflow** incorporating ARES, VisIt, and Scikit Learn.

Our new capability is free-standing, with a Python interface, allowing it to be **incorporated into existing tools and workflows**. We will present our path forward to supporting massive ensembles on the El Capitan system to be sited in 2022 by discussing the results of scaling to a million member ensemble and our ongoing collaboration with FLUX, the next generation scheduler team in Livermore Computing.

Creating a set of simulations from a single simulation is a time consuming and painful process

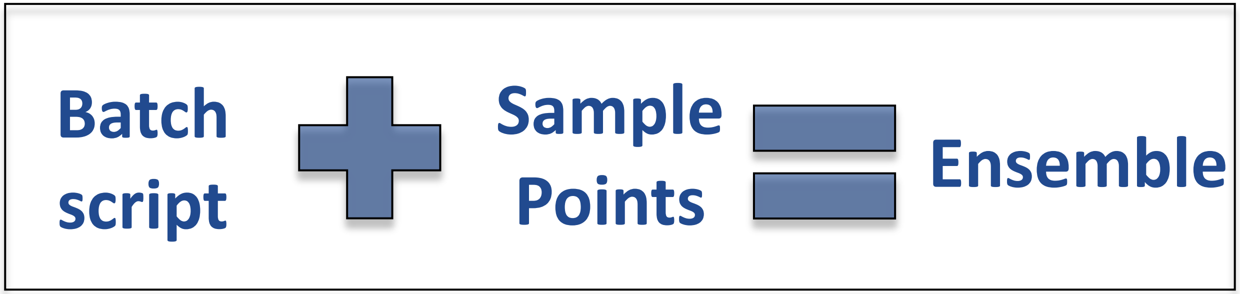


```
#!/bin/bash

srun –n 4 ares deck
```

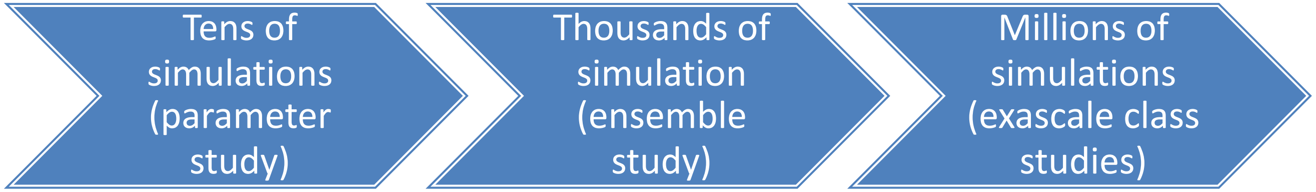
- These studies are critical to WCI mission
  - Parameter/Sensitivity Studies
  - Machine Learning based Studies
  - Design Optimization Studies
  - UQ Studies
  - Mesh Resolution Studies
  - Convergence Studies

Themis, a new scaleable ensemble generator, solves these problems

The Themis Ensemble Generator leverages the batch submission script to create studies at scale

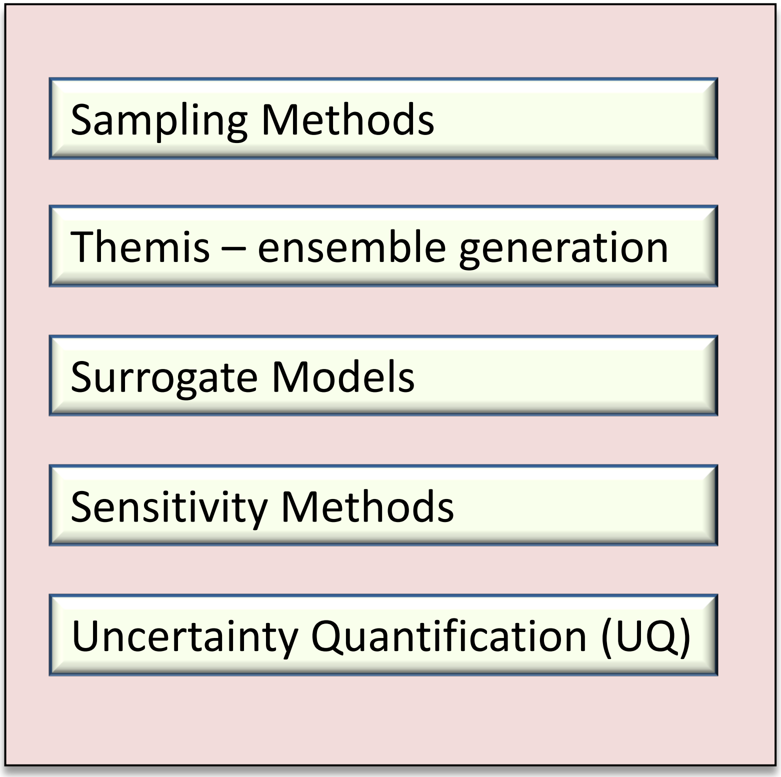


- Batch script can be user-developed or created by a tool or suite


Tens of simulations (parameter study)
Thousands of simulation (ensemble study)
Millions of simulations (exascale class studies)

- Generate studies using a CLI or a feature rich Python API
- Stores simulation-produced data (scalars, time histories, images)
- Easily integrates into existing and emerging workflows
- Generate simulations in a single batch job
- Auto-restart of studies

Themis is part of the UQ Pipeline project and its Components Strategy



Sampling Methods

Themis – ensemble generation

Surrogate Models

Sensitivity Methods

Uncertainty Quantification (UQ)

Goal:

Created a set flexible and resuable components

Each of which provides specific functionality or capability

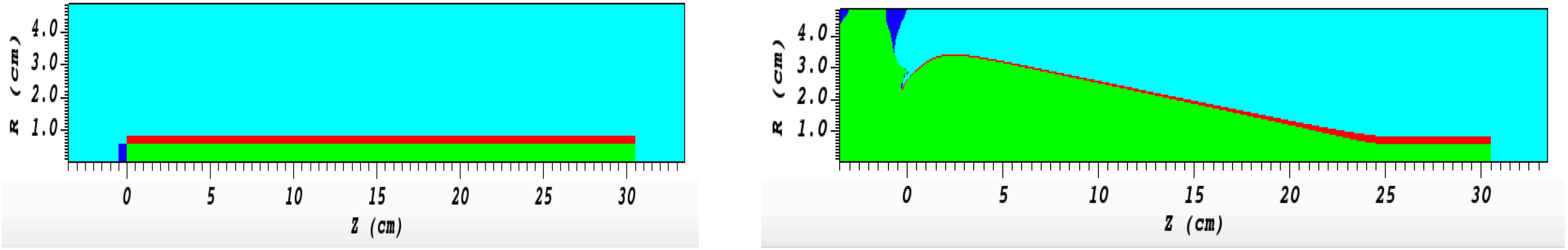
Easier to work with existing and emerging workflows

Components deployed at /collab/usr/gapps/uq/uqp/

Demo #1: Cylinder Test Simulation

Filled Boundary
Var: Materials

🟥 1 WALL
🟩 2 HE
🟦 3 HECB
▨ 4 AIR



- Consists of 2 files: an Ares input deck (ares_cyl.in) and a batch (runscript.csh) submission script
- Parameters varied
  - HE Model: Cheetah, Crest, I&G, Augmented I&G
  - HE Thickness (hethk)
  - Zones per cm (zpcm)

Study Creation

themis **create** batch samples.csv -b -f ares_cyl.in

themis **allocation** -N 5 -p pbatch -b wbronze -m cli_cyl

themis **dryrun** 0

themis **execute**

Themis Study Status

themis **progress** -v

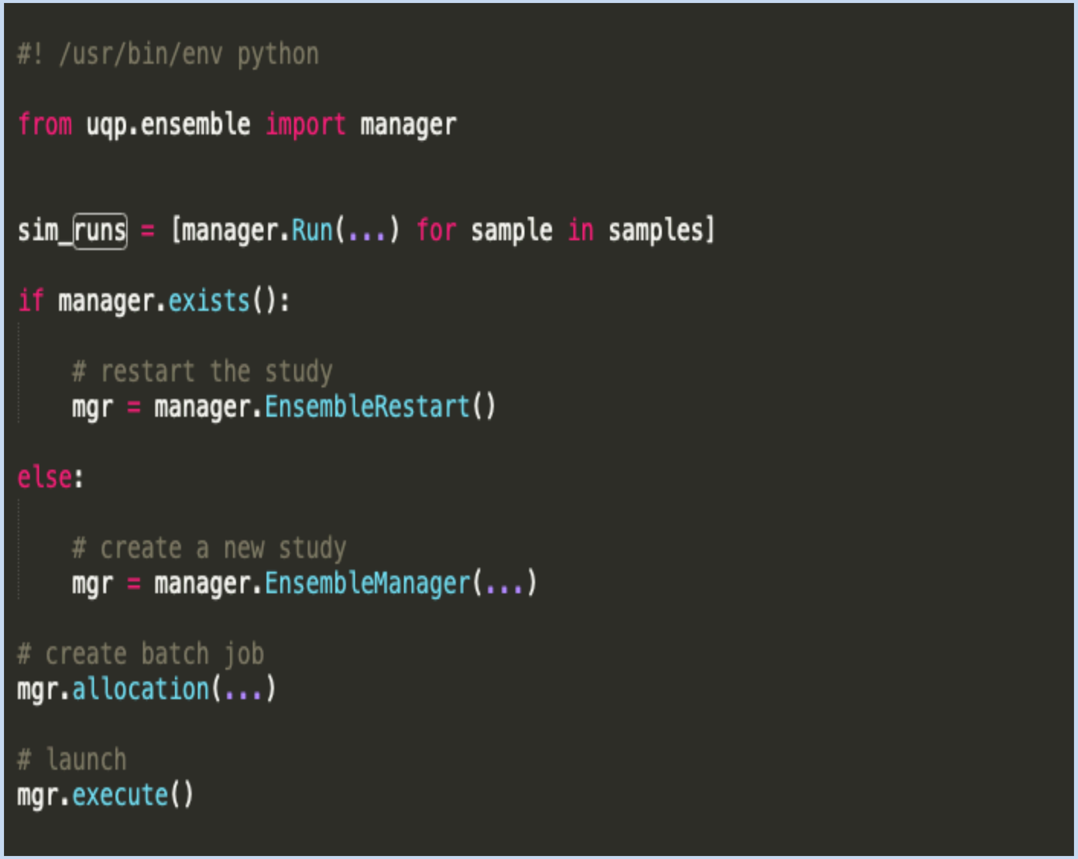
themis **display** 0 11 -s

themis **status** successful

Themis’ flexible Python API gives you full control over your studies

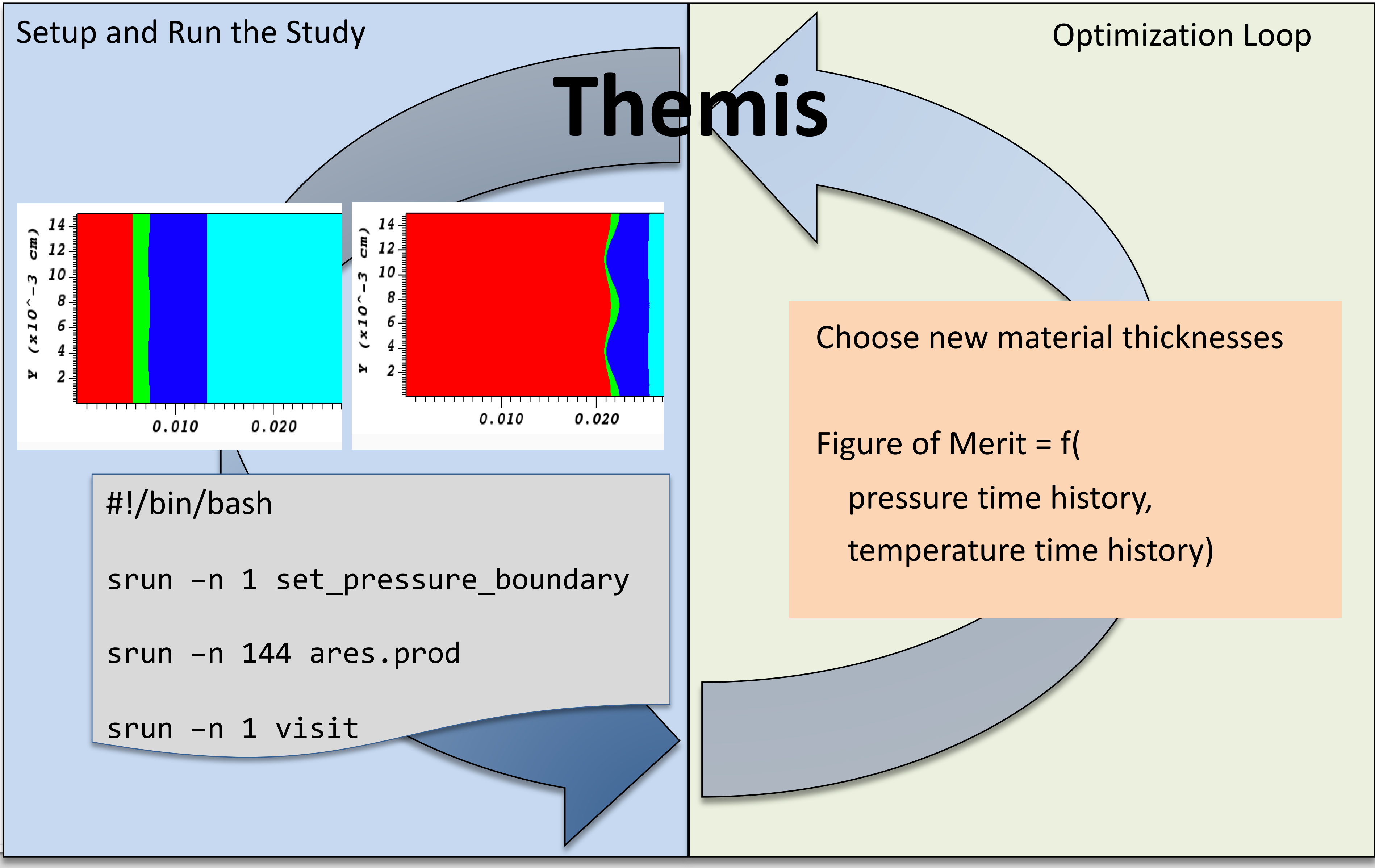
Setup and Run the Study

Themis



```
#! /usr/bin/env python

from uqp.ensemble import manager


sim_runs = [manager.Run(...) for sample in samples]

if manager.exists():

    # restart the study
    mgr = manager.EnsembleRestart()

else:

    # create a new study
    mgr = manager.EnsembleManager(...)

# create batch job
mgr.allocation(...)

# launch
mgr.execute()
```

Demo 2: Design Optimization Study



Setup and Run the Study

Optimization Loop

Themis

```
#!/bin/bash

srun -n 1 set_pressure_boundary

srun -n 144 ares.prod

srun -n 1 visit
```

Choose new material thicknesses

Figure of Merit = f(

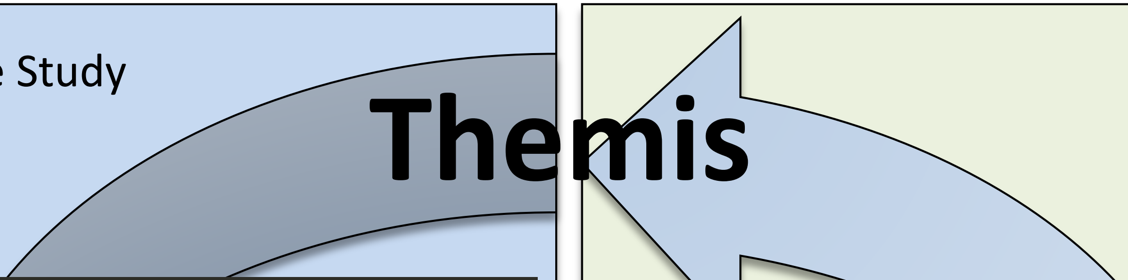
pressure time history,

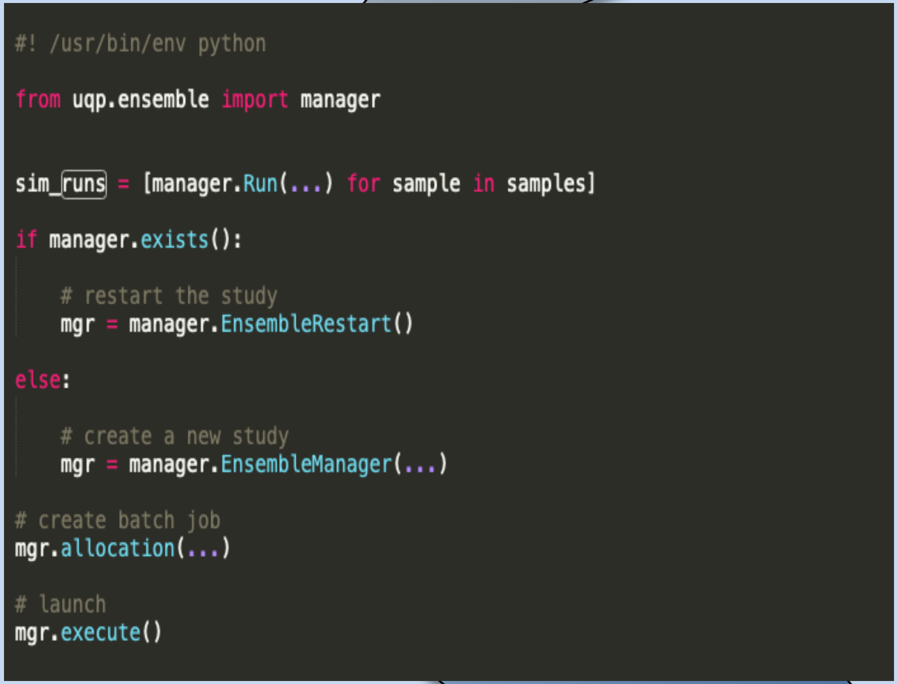
temperature time history)

Structure of the optimization loop

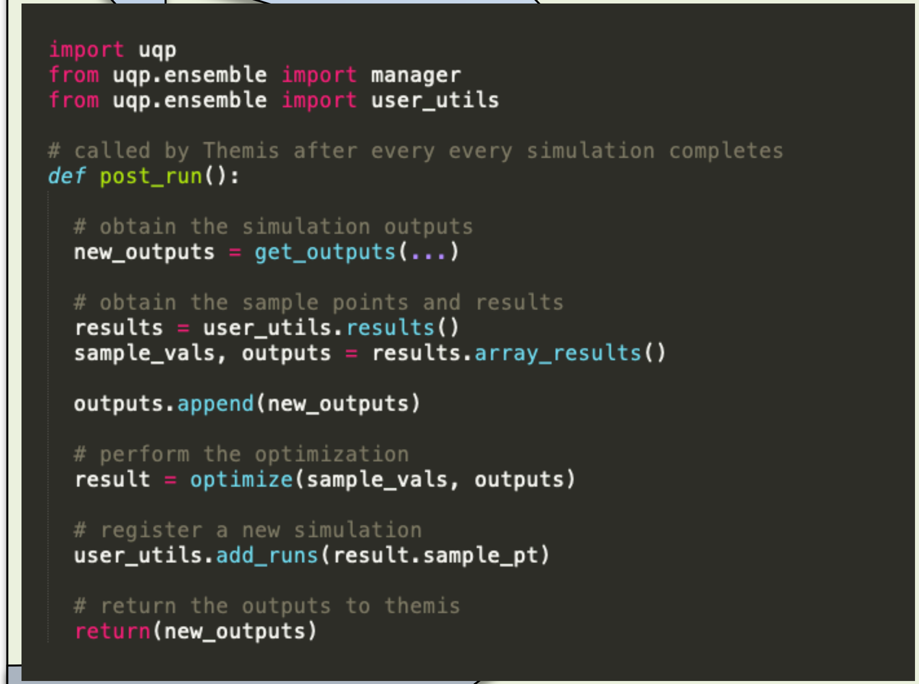
Setup and Run the Study

Optimization Loop





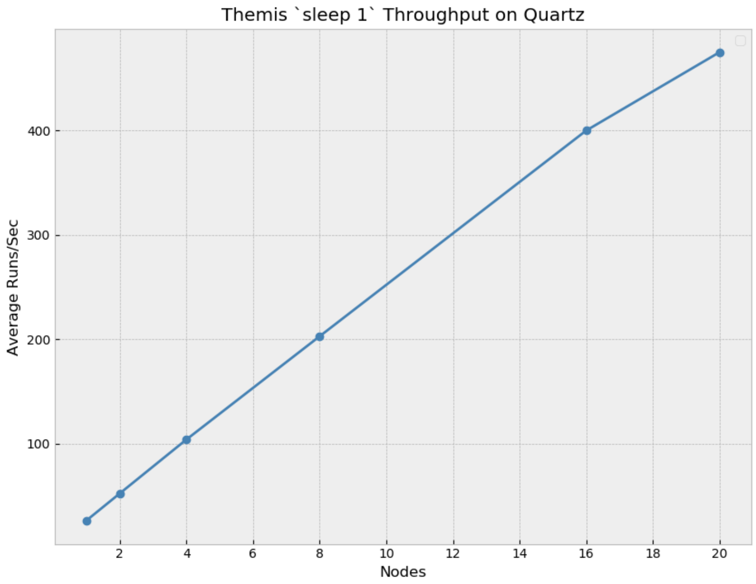
```
#! /usr/bin/env python

from uqp.ensemble import manager


sim_runs = [manager.Run(...) for sample in samples]

if manager.exists():

    # restart the study
    mgr = manager.EnsembleRestart()

else:

    # create a new study
    mgr = manager.EnsembleManager(...)

# create batch job
mgr.allocation(...)

# launch
mgr.execute()
```



```
import uqp
from uqp.ensemble import manager
from uqp.ensemble import user_utils

# called by Themis after every every simulation completes
def post_run():

  # obtain the simulation outputs
  new_outputs = get_outputs(...)

  # obtain the sample points and results
  results = user_utils.results()
  sample_vals, outputs = results.array_results()

  outputs.append(new_outputs)

  # perform the optimization
  result = optimize(sample_vals, outputs)

  # register a new simulation
  user_utils.add_runs(result.sample_pt)

  # return the outputs to themis
  return(new_outputs)
```

Themis efficiently scales a million member study

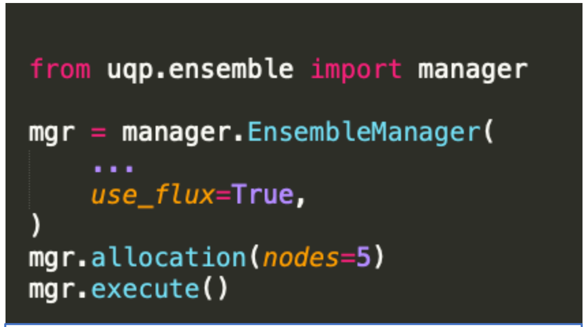
- Capable of executing hundreds of runs per second
- Files and directories are created on-demand and completed in parallel, so start-up time is low
- Efficiently uses compute resources within each allocation



```
#!/usr/bin/bash

srun –n 1 lagj

srun –n 144 ares.prod
```

Themis is Well-Positioned for Exascale

- Fully Integrated with Flux
  - Just set a flag and Themis handles the rest
  - Automatically leverages Flux's hierarchical scheduling abilities
- Generating studies quantifies the overhead of Themis working with FLUX
- Generating studies that also stress test FLUX
- 1M member ensemble of ‘sleep 0’ completed in 1 hour 15 minutes



```
from uqp.ensemble import manager

mgr = manager.EnsembleManager(
    ...
    use_flux=True,
)
mgr.allocation(nodes=5)
mgr.execute()
```

That's all there is to it---just fill in the "..." with your application and your inputs.

- Themis is scaleable and robust
  - Minimizes the time to create studies
  - Supports today’s multi-simulation studies
  - Positioned to address the next generation studies

Documentation and Availability

- Themis is deployed and available on all LC machines, as well as LANL's Trinitite.
- Themis is stable. It is fully documented, along with tutorials and examples, at https://lc.llnl.gov/uqp/docs/ and https://rzlc.llnl.gov/uqp/docs/.
- "Themis" MS Team

David Domyancic

domyancic1@llnl.gov

(925) 422-1847

James Corbett

corbett8@llnl.gov

(925) 424-4453

Andrew Fillmore

filllmore1@llnl.gov

(925) 423-7658

Lawrence Livermore
National Laboratory