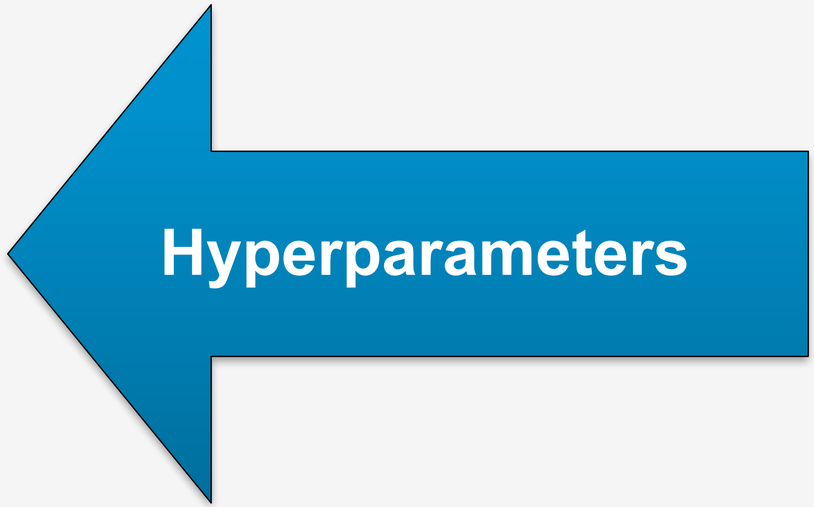
A Quick Look at Hyperparameter Tuning

https://github.com/argonne-lcf/ATPESC_2019/blob/master/HPS/

Misha Salim
Argonne Leadership Computing Facility
msalim@anl.gov

Hyperparameters may control model structure

```
def run_model(
    batch_size=128,
    epochs=1,
    nunits1=16,
    nunits2=16,
    dropout=0.1,
    activation='relu',
    lr=0.01,
    momentum=0.0,
  ):
  """
  Train MLP with 2 hidden layers on MNIST digit classification.
  Returns the minimization objective, which is (-1.0 * test_accuracy)
  """
  global x_train, x_test, y_train, y_test, num_classes
  model = Sequential()
  model.add(Dense(nunits1, activation=activation, input_shape=(784,)))
  model.add(Dropout(dropout))
  model.add(Dense(nunits2, activation=activation))
  model.add(Dropout(dropout))
  model.add(Dense(num_classes, activation='softmax'))
```



...or optimizer settings

```
model.compile(loss='categorical_crossentropy',
              optimizer=SGD(lr=lr, clipnorm=1., momentum=momentum),
              metrics=['accuracy'])

history = model.fit(x_train, y_train,
                    batch_size=batch_size,
                    epochs=epochs,
                    verbose=0,
                    validation_data=(x_test, y_test))
score = model.evaluate(x_test, y_test, verbose=0)
return -1.0*score[1] # minimize -1.0*test_accuracy
```

Scikit-Optimize: Bayesian Optimization

Treat hyperparameters like magic rules of thumb?

Or optimize them in an "outer loop" over hyperparameters?

https://scikit-optimize.github.io/

Defining a Search Space

Consider tuning:
batch size (64, 128, 256, or 512)
learning rate (0.001 to 1.0)

```
problem_dimensions = [
    (6, 9),       # log2 (batch_size)
    (-3.0, 0.0), # log10 (learning rate)
]
def to_dict(x):
  return dict(batch_size=2**x[0], lr=10.0**x[1])
```

Create an Optimizer

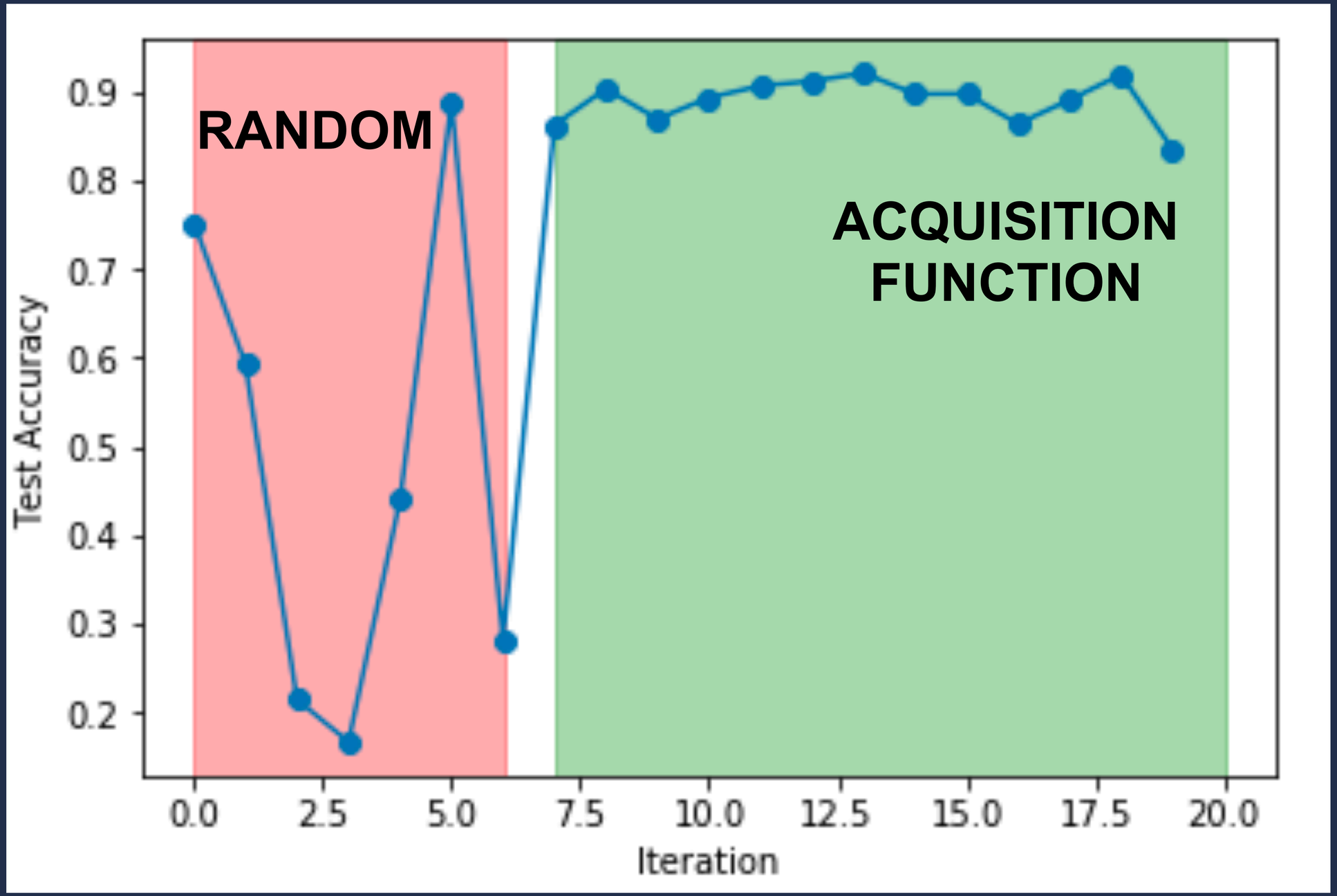
Use ExtraTreesRegressor to predict test_accuracy from hyperparameters

```
optimizer = skopt.Optimizer(
    problem_dimensions,
    "ET",
    acq_optimizer="sampling",
    n_initial_points=6,
)
```

Run Black-box Optimization Loop

1) Select new hyperparameters **x** from optimizer
2) Build, train, evaluate model with **x**
3) Update optimizer with objective

```
N = 20
for i in range(N):
    next_x = optimizer.ask()
    eprint(f"[{i+1}/{N}]", to_dict(next_x))
    objective = run_model(**to_dict(next_x))
    optimizer.tell(next_x, objective)
```


RANDOM
ACQUISITION FUNCTION
Test Accuracy
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Iteration

How do we parallelize this?

deephyper.readthedocs.io

Production-ready modules on Theta
Detailed tutorials on readthedocs

DeepHyper

Search docs

INSTALLATION

Local

Theta

Cooley

Analytics

GETTING STARTED

Create a new hyperparameter search problem

Create a problem directory

Create load_data.py

Create model_run.py

A hyperparameter search (HPS) problem can be defined using three files with a HPS problem directory:

```
hps_problem_directory/
      load_data.py
      model_run.py
      problem.py
```

We will illustrate the HPS problem definition using a regression example. We will use polynome function to generate training and test data and run a HPS to tune the hyperparameters of a simple neural network.

Create a problem directory

First, we will create a hps_problem_directory `polynome2`.

bash

```
mkdir polynome2
cd polynome2
```

Create load_data.py

DeepHyper runs with Balsam Workflows

balsam.readthedocs.io

Theta Ensemble Jobs

```
#!/bin/bash

myApp="/path/to/app --input="
```

Job scripts run on MOM (Broadwell) nodes

Compute (KNL) Nodes

- nid00001
- nid00002
- nid00003
- nid00004
- nid00005

Theta Ensemble Jobs

```
#!/bin/bash

myApp="/path/to/app --input="

aprun -n 64 -N 64 $myApp input1 >& run1.out &
sleep 1
```

Job scripts run on MOM (Broadwell) nodes

aprun

Compute (KNL) Nodes

- nid00001
- nid00002
- nid00003
- nid00004
- nid00005

Theta Ensemble Jobs

Job scripts run on MOM (Broadwell) nodes

```
#!/bin/bash

myApp="/path/to/app --input="

aprun -n 64 -N 64 $myApp input1 >& run1.out &
sleep 1

aprun -n 128 -N 64 $myApp input2 >& run2.out &
sleep 1
```

aprun

aprun

Compute (KNL) Nodes

nid00001

nid00002

nid00003

nid00004

nid00005

Theta Ensemble Jobs

Job scripts run on MOM (Broadwell) nodes

```
#!/bin/bash

myApp="/path/to/app --input="

aprun -n 64 -N 64 $myApp input1 >& run1.out &
sleep 1

aprun -n 128 -N 64 $myApp input2 >& run2.out &
sleep 1

aprun -n 128 -N 64 $myApp input3 >& run3.out &
wait
```

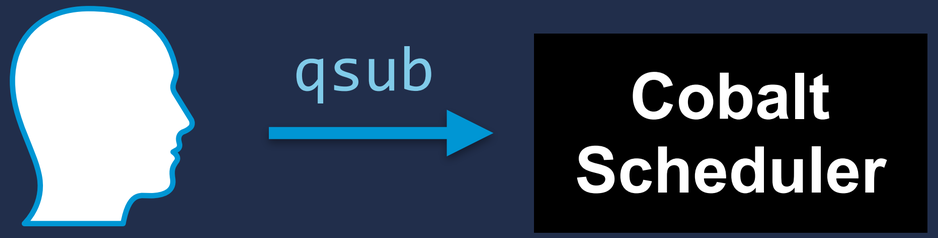
Compute (KNL) Nodes

alcf.anl.gov/user-guides/running-jobs-xc40#bundling-multiple-runs-into-a-script-job

What do we mean by workflow?

Sometimes a few scripts is enough

```
(100 runs) (1024 nodes) (12 hours) = 1.23 M node-hours
```



- Queue up to 20 script jobs
- Keep organized directory layout
- Compose shell commands with bash or Python scripting

What do we mean by workflow?

Sometimes a few scripts is enough

(100 runs) (1024 nodes) (12 hours) = 1.23 M node-hours

Large ensembles: start building more complex workflows

(9600 runs) (128 node) (1 hour) = 1.23 M node-hours

- Run jobs concurrently ***and*** one-after-another?
- Track which tasks are left to run?
- Handle timed-out runs?

What do we mean by workflow?

Sometimes a few scripts is enough

(100 runs) (1024 nodes) (12 hours) = 1.23 M node-hours

Large ensembles: start building more complex workflows

(9600 runs) (128 node) (1 hour) = 1.23 M node-hours

Human effort scales unfavorably with # of runs

(12,288,000 runs) (1 node) (6 minutes) = 1.23 M node-hours

What do we mean by workflow?

Max 20 queued jobs

Lacking job packing / MPMD execution

Cumbersome error & timeout handling

Human effort scales unfavorably with # of runs

```
(12,288,000 runs) (1 node) (6 minutes) = 1.23 M node-hours
```

You either build workflow tools or adopt existing ones

Balsam

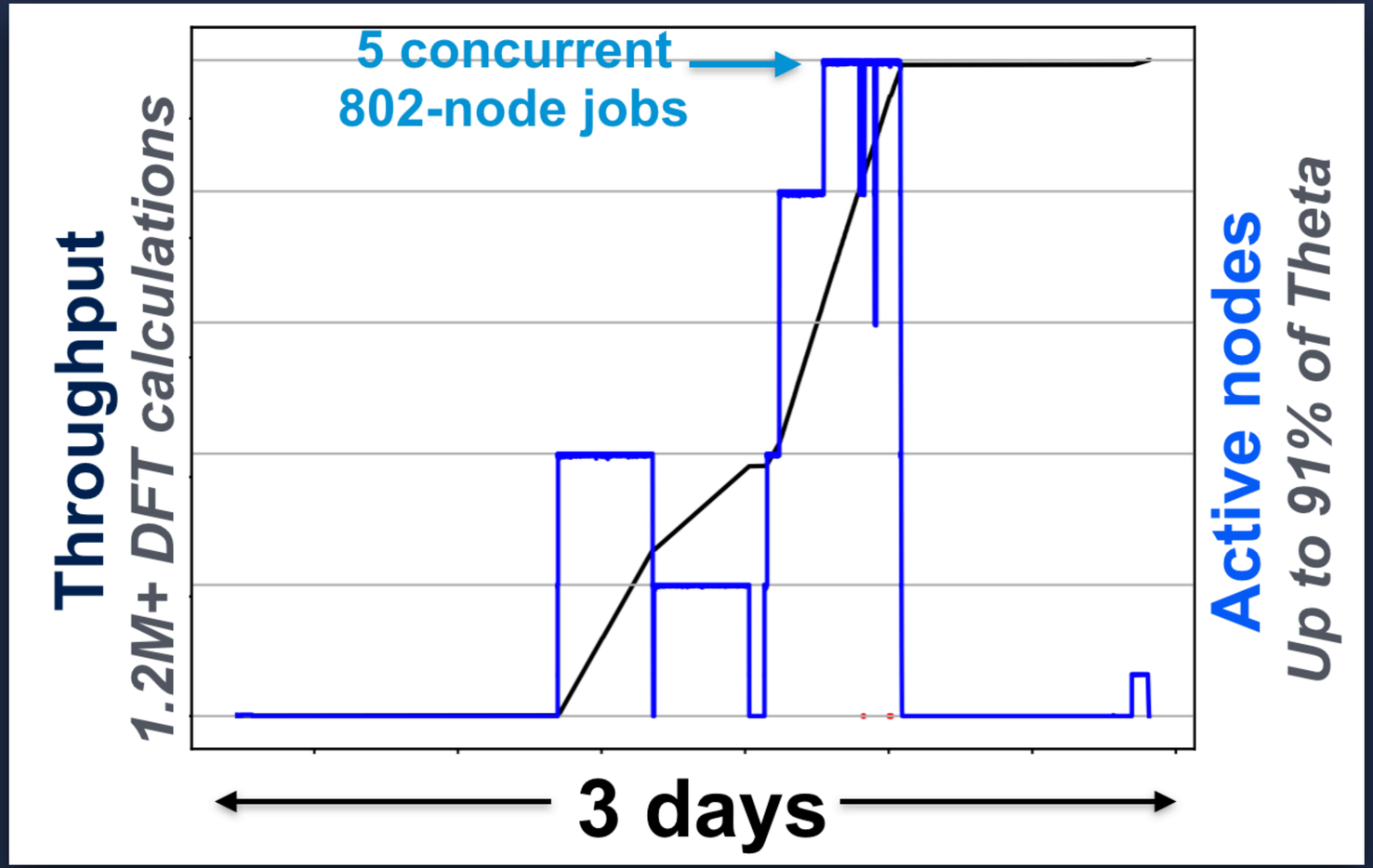
Automated scheduling and execution on ALCF Systems

- **You tell Balsam about your jobs (Python API or command line)**
- **Balsam automates the rest: scheduling & execution**
- **no modification of user applications**
- **strong fault tolerance at task level**
- **Workflow status and project statistics available at-a-glance**

Scaled to 91% of Theta, 1.2M+ tasks

Constructing and Navigating Polymorphic Landscapes of Molecular Crystals (PI: Alexandre Tkatchenko)

- User dumped a few million DFT runs into Balsam
- These ran over several jobs, totaling 7M+ core hours
- Up to 5 **simultaneous** Cobalt jobs running tasks from one centralized service



Release in production @ ALCF


```
module load balsam
```

Use Balsam for DeepHyper & other high-throughput workloads

- https://balsam.readthedocs.io
- https://deephyper.readthedocs.io