CANDLE Tutorial: Workflow Technologies

ECP Annual Meeting, Houston, TX
February 4, 2020

Justin M Wozniak
Data Science & Learning
Argonne National Laboratory
woz@anl.gov

Talk Outline

- Computing environment
- Application cases
- Workflow technologies
- Performance results
- Next steps

Deep learning on supercomputers

- Steep learning curve with myriad technologies
  - Workflow manager (Swift/T, EMEWS) ; Scheduler ; Scripting

  - Deep learning (Keras, TensorFlow, Horovod)

  - Optimization algorithms (R, Python)

  - MPI implementation (MVAPICH, Open MPI) – or other communication

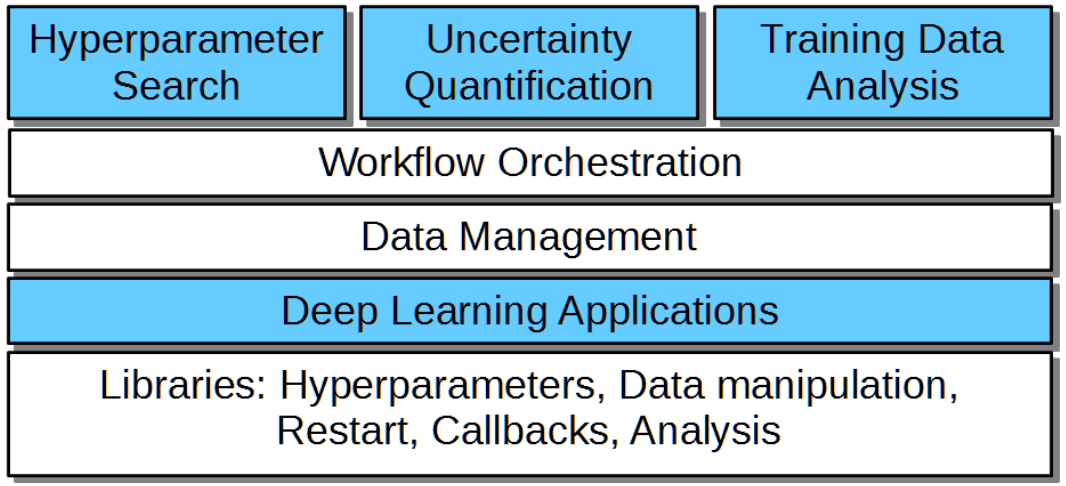
etc. …

CANDLE/Supervisor Goals

- Develop an exascale deep learning environment for cancer
- Building on open source deep learning frameworks
- Optimization for CORAL and exascale platforms
- Support all three Pilot project needs for deep learning
- Collaborate with DOE computing centers, HPC vendors and ECP co-design and software technology projects
- Mission statement: Enable the most challenging deep learning problems in cancer research to run on the most capable supercomputers in the DOE
- Provide a path forward for machine learning applications at the largest scale…

CANDLE/Supervisor overview

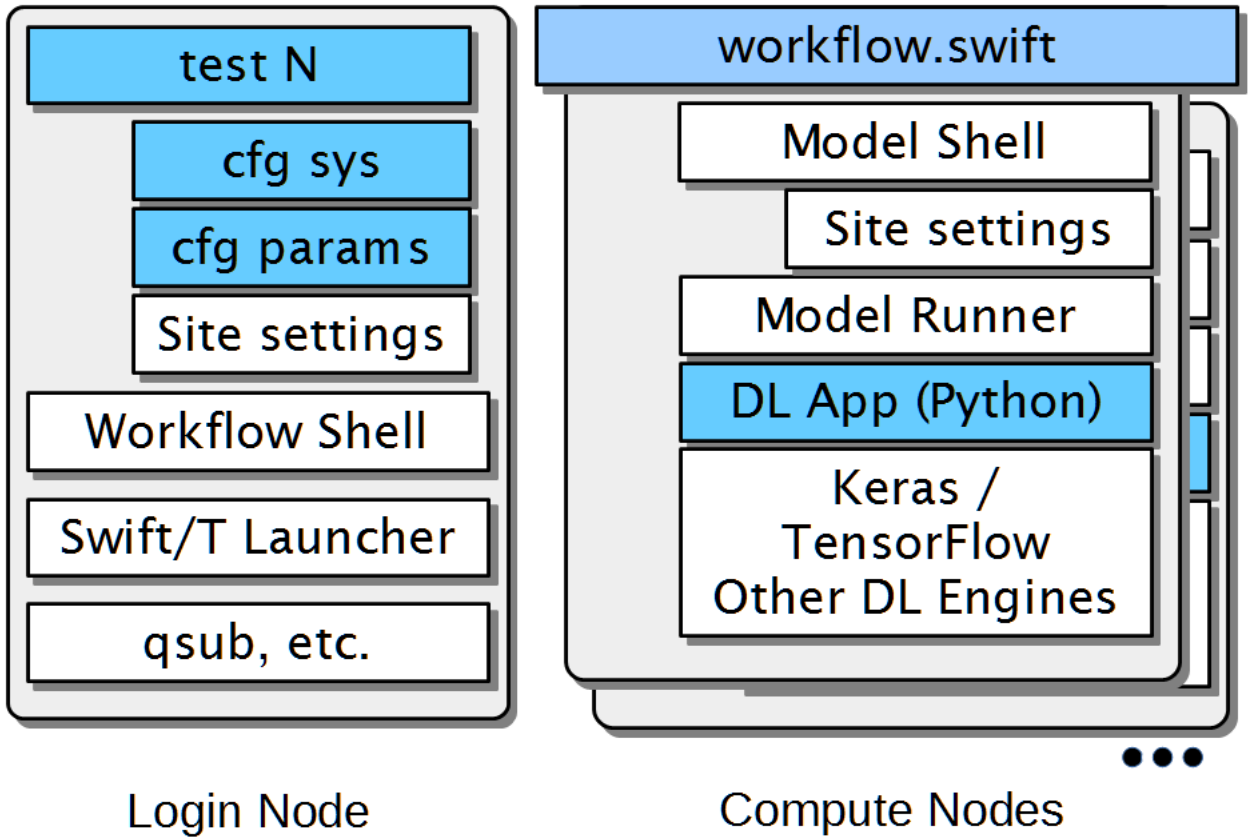
- CANDLE/Supervisor consists of several high-level workflows:
  - Capable of modifying/controlling application parameters dynamically as the workflow progresses and training runs complete
  - Distribute work across large computing infrastructure, manage progress
- Underlying applications are Python programs that use Keras/TensorFlow
- "User code" shown in blue
- "Provided tools" shown in white
- New studies would be developed by modifying the blue sections



- Wozniak et al. CANDLE/Supervisor: A workflow framework for machine learning applied to cancer research. BMC Bioinformatics 19(18), 2018.

CANDLE/Supervisor Implementation

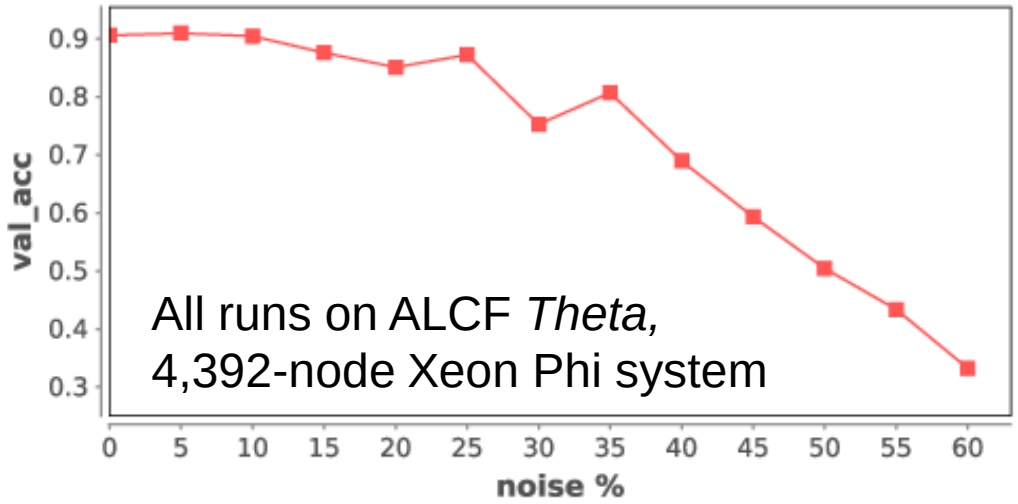
- Runs start with a test script
- CFG scripts contain settings for a system or parameters for a given study (e.g., search space)
- Reusable site settings
- The workflow shell script sets up the run
- Swift/T launches and manages the workflow
- Reusable Model scripts set up each app run
- The DL app uses Keras/TF plus CANDLE Python libraries



Example: Robustness under noise

- Simple parameter sweep- measure impact of bad data injection into NT3 model
- Workflow script loops over range of noise levels and trials
- Parallel loops expose all work to the system concurrently
- JSON fragment is assembled with CANDLE hyperparameters
- CANDLE **obj()** function invokes the model scripts to run NT3
- Resulting validation accuracy is simply printed to the screen
- This will be a topic for the hands-on session

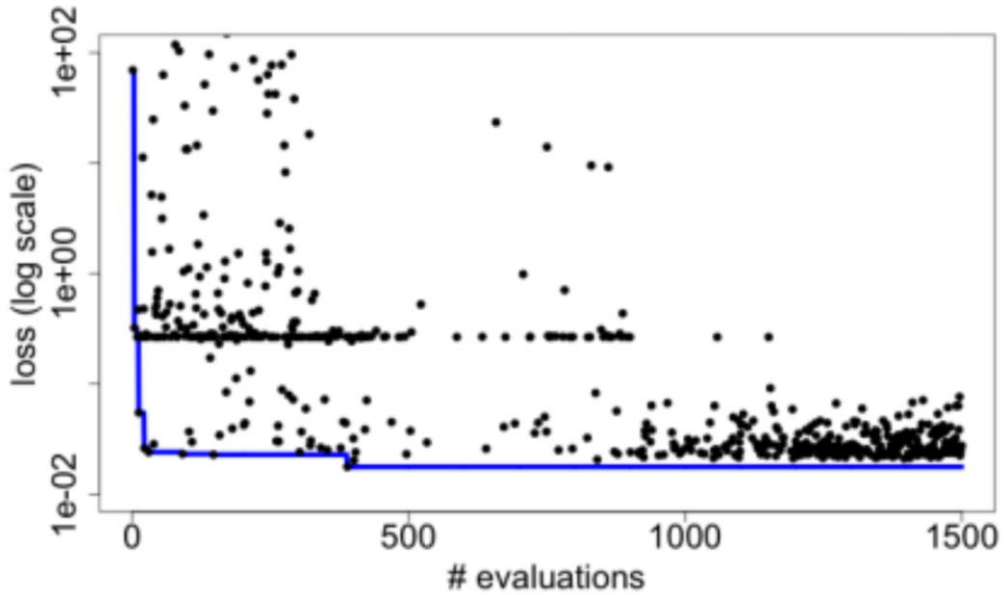
```
float noise_levels[] = [0:num_noises];
int trials[]         = [0:num_trials-1];

foreach level, i in noise_levels {
  foreach trial, j in trials   {
    run_id = "%02i-%02i" % (i, j);
    json = "{\"noise_level\":%f}"%noise_level;
    noise_level = level * noise_step;
    result = obj(json, run_id);
    printf("result %s : noise %0.3f : %s",
           run_id, noise_level, result);
  }
}
```



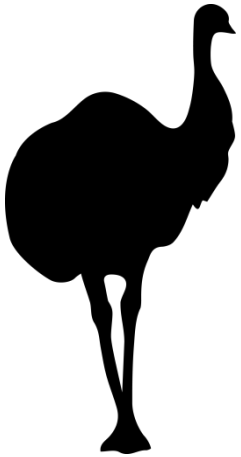
Example: Hyperparameter optimization

- Search for the best combination of hyperparameters for a given DL app- neuron counts, functions, etc.
- Optimize for validation loss, a measure of error
- Use an external R library to perform the optimization
- Iteratively receive trial hyperparameters, evaluate them with the obj() function, return results
- Algorithm terminates after convergence or iteration limit

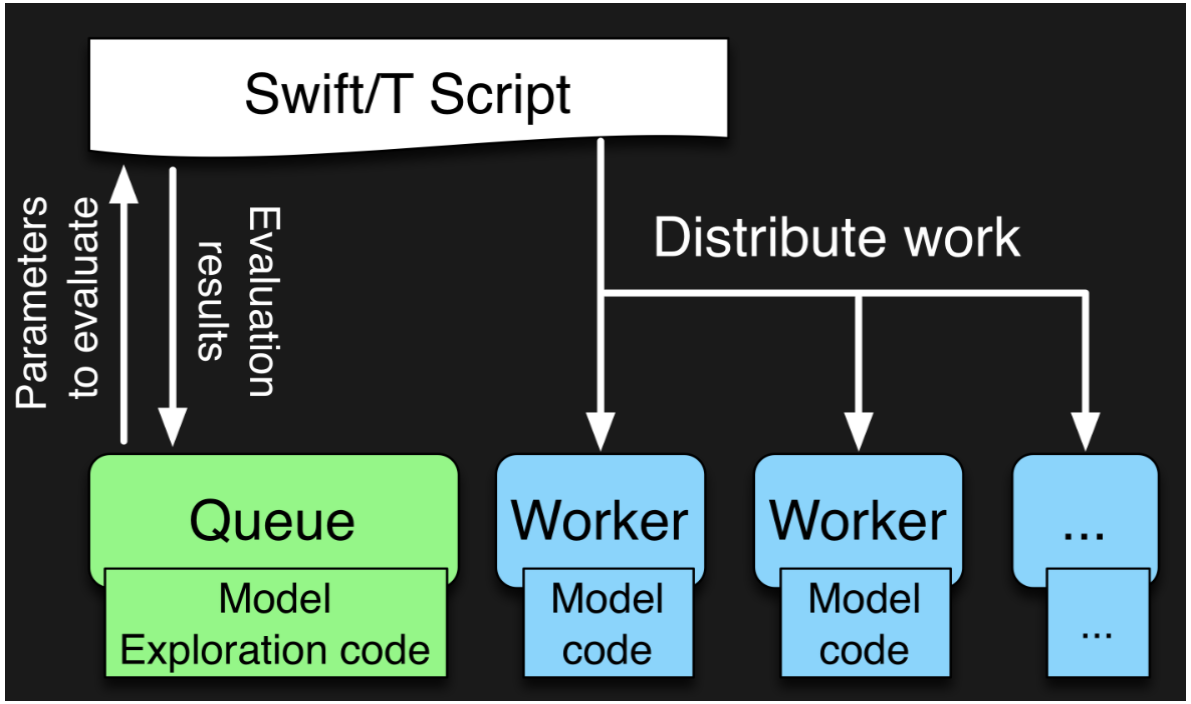
```
for (boolean b = true, int i = 1;
     b;      b=c, i = i + 1) {
  string params =  EQR_get(ME);
  boolean c;

  if (params == "DONE") {
    string finals =  EQR_get(ME);
    c = false;
  } else {
    string param_array[] = split(params, ";");
}
```



EMEWS workflow structure



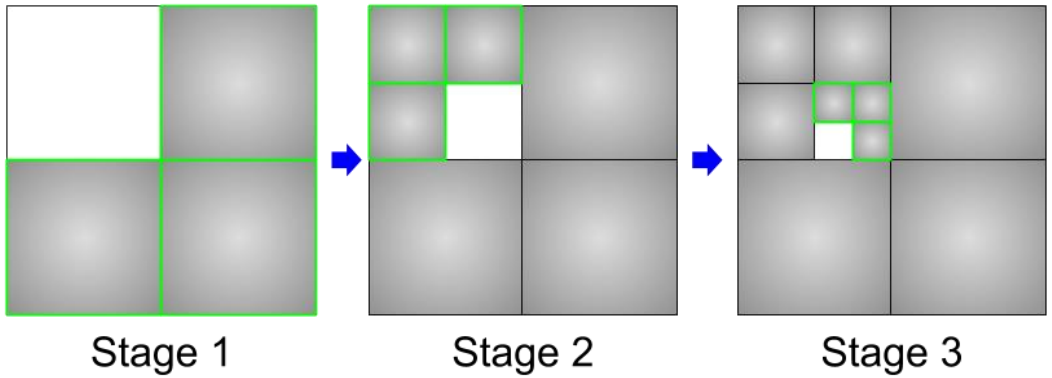
- The core novel contributions of EMEWS are shown in green, these allow the Swift script to access a running **Model Exploration (ME)** algorithm, and create an **inversion of control (IoC)** workflow
- Both green and blue boxes accept **existing, generic multi-language code-** could be anything; we use **optimization** and **deep learning modules**
- **http://emews.org**



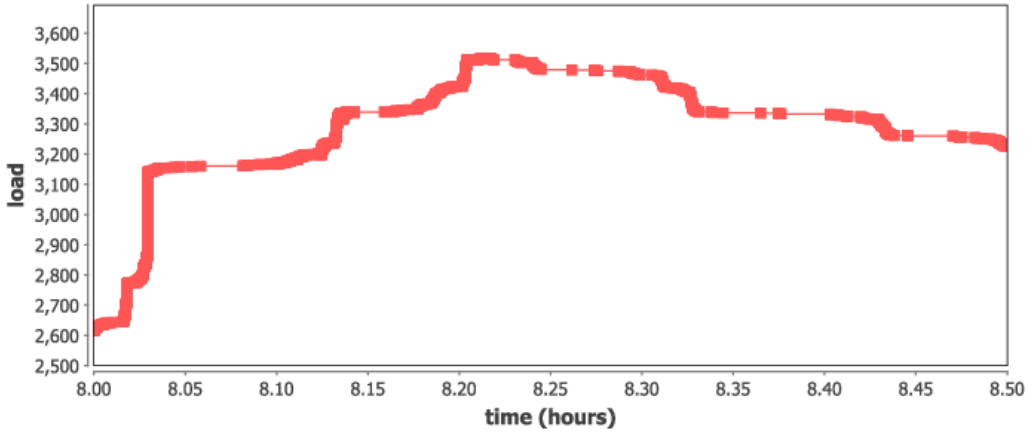
- Ozik, Collier, and Wozniak. From desktop to large-scale model exploration with Swift/T. Proc. Winter Simulation Conference 2016.

Example: Data analysis workflow

- Split up the training data into subsets, iteratively train on most remaining subsets.
- Weight sharing from one subset to the next (incremental learning)
- Allows for investigations into data quality and learning patterns
- Could also boost performance by preventing overloading data ingest limits
- Recursive calls define the datasets for train
- Runs at large scale on Summit, ramp-up/down

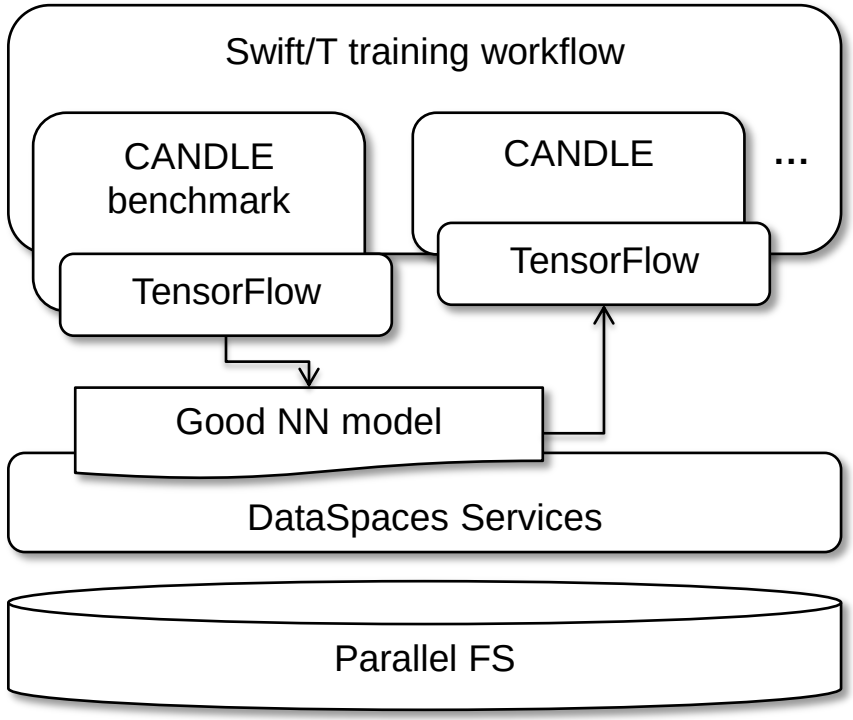


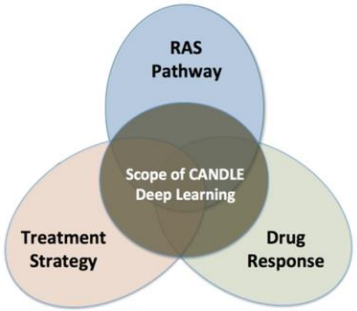

```
run_stage(int N, int S, string this, int stage,
          void block, string plan_id) {
  void parent = run_single(this, stage, block, plan_id);
  if (stage < S) {
    foreach i
      run_sta

    }}
}

run_single(st
  json_fragme
  json = "{\"
  block => ob
}
```

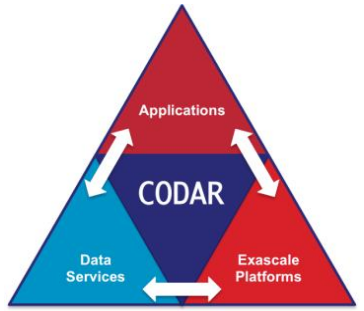


Advanced data management

- **CANDLE** workflows produce a great number of medium-sized ML models
- **Goal:** Cache these on compute node storage for *possible* later use
- Need to flush to global FS before end of run, but many models will be discarded
- **Accomplishment:** Integrated Swift/T workflow system used in CANDLE with DataSpaces client
- Accelerate CANDLE workflow performance, enable novel training strategies (parameter sharing)



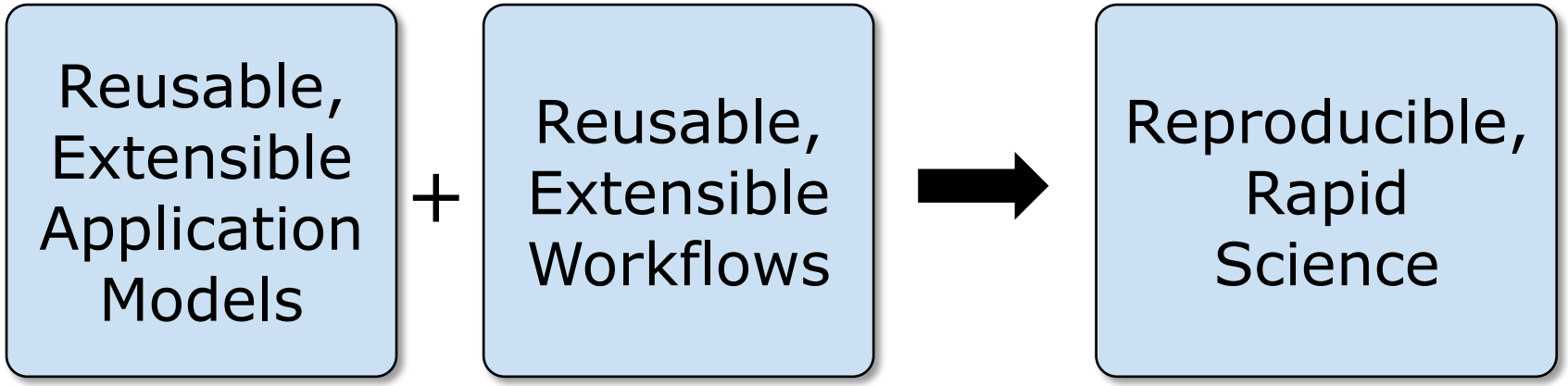


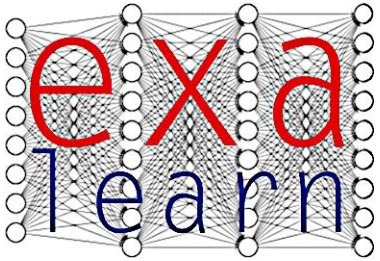


- Wozniak et al. Scaling deep learning for cancer with advanced workflow storage integration. Proc. MLHPC @ SC 2018.

Next steps in exascale learning

- Enable learning studies via massive parallelism for new workflows
- Quickly scale new methods from Python notebooks to supercomputers
- Support quick ‘what if’ scenario evaluation for a range of questions
- Integrate with parallel learning modules , parallel simulations, and programming models (Legion)





Thanks

- Thanks to the organizers

- Code and guides:
  - CANDLE GitHub: https://github.com/ECP-CANDLE
  - Swift/T Home: http://swift-lang.org/Swift-T
  - EMEWS Tutorial: http://www.mcs.anl.gov/~emews/tutorial

Questions?

Acknowledgments

This research was supported by the Exascale Computing Project (ECP), Project Number: 17-SC-20-SC, a collaborative effort of two DOE organizations - the Office of Science and the National Nuclear Security Administration, responsible for the planning and preparation of a capable exascale ecosystem, including software, applications, hardware, advanced system engineering and early testbed platforms, to support the nation's exascale computing imperative.