ALTERNATIVE PROGRAMMING VIA TRANSLATION

DISTRIBUTED FUNCTIONAL PROGRAMMING OVER MPI WITH SWIFT/T

JUSTIN M WOZNIAK

DATA SCIENCE & LEARNING
ARGONNE NATIONAL LABORATORY

PAW-ATM @ SC, Denver

November 12, 2023

EXASCALE COMPUTING PROJECT

QUESTION:

HOW CAN I USE FUNCTIONAL PROGRAMMING TO GET AUTOMATIC PARALLELISM AND RUN AT LARGE SCALE OVER MPI?

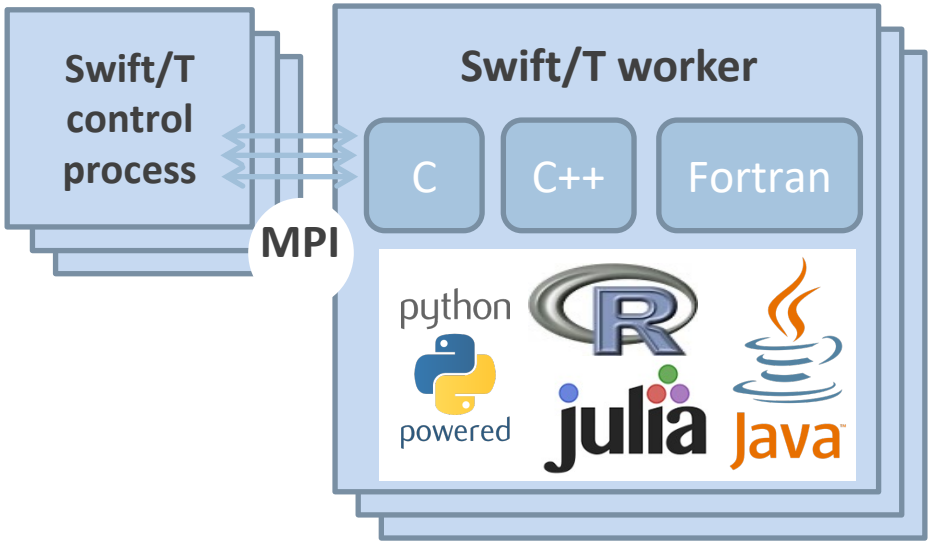
SWIFT/T: ENABLING HIGH-PERFORMANCE SCRIPTED WORKFLOWS

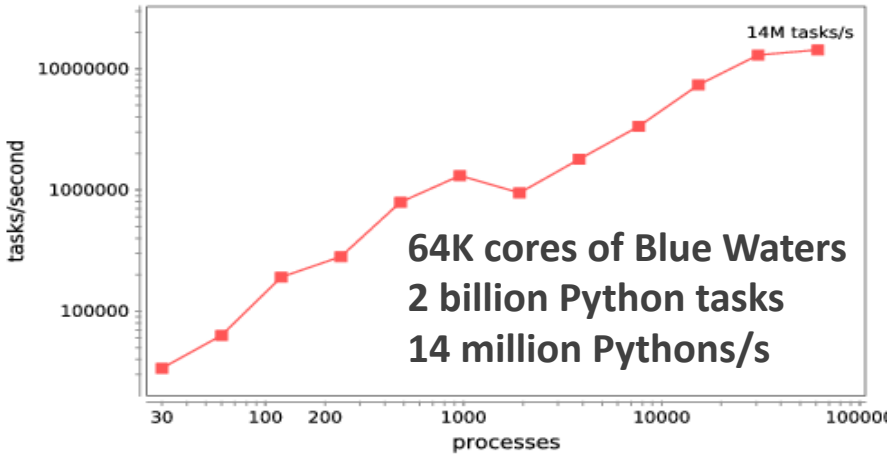
Supports tasks written in many languages

- Write site-independent scripts, translates to MPI
- Automatic task parallelization and data movement
- Invoke native code, script fragments
- Rapidly subdivide large partitions for MPI jobs in multiple ways

SWIFT/T

2018 R&D 100 WINNER






```
$ spack install stc
```

```
$ conda install -c swift-t swift-t
```

Swift/T: Scalable data flow programming for distributed-memory task-parallel applications
Proc. CCGrid 2013.

Argonne NATIONAL LABORATORY

THE SWIFT PROGRAMMING MODEL

All progress driven by concurrent dataflow

```
(int r) myproc (int i, int j)
{
    int x = F(i);
    int y = G(j);
    r = x + y;
}
```

- `F()` and `G()` implemented in native code or external programs
- `F()` and `G()` run in concurrently in different processes
- `r` is computed when they are both done
- This parallelism is *automatic*
- Works recursively throughout the program's call graph

SWIFT SYNTAX

- Data types

```
int i = 4;
string s = "hello world";
file image<"snapshot.jpg">;
```

- Shell access

```
app (file o) myapp(file f, int i)
{ mysim  "-s" i @f @o; }
```

- Structured data

```
typedef image file;
image A[];
type protein_run {
  file pdb_in; file sim_out;
}
bag<blob>[] B;
```

- Conventional expressions

```
if (x == 3) {
    y = x+2;
    s = strcat("y: ", y);
}
```

- Parallel loops

```
foreach f,i in A {
    B[i] = convert(A[i]);
}
```

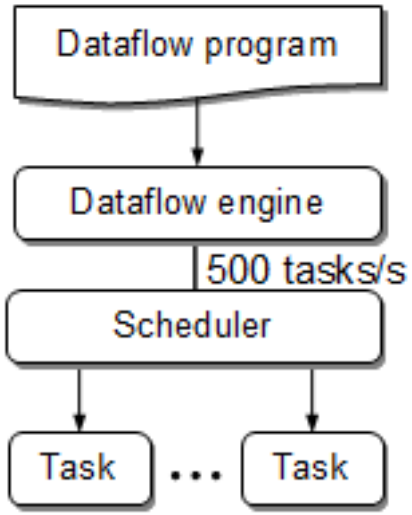
- Data flow

```
merge(analyze(B[0], B[1]),
      analyze(B[2], B[3]));
```

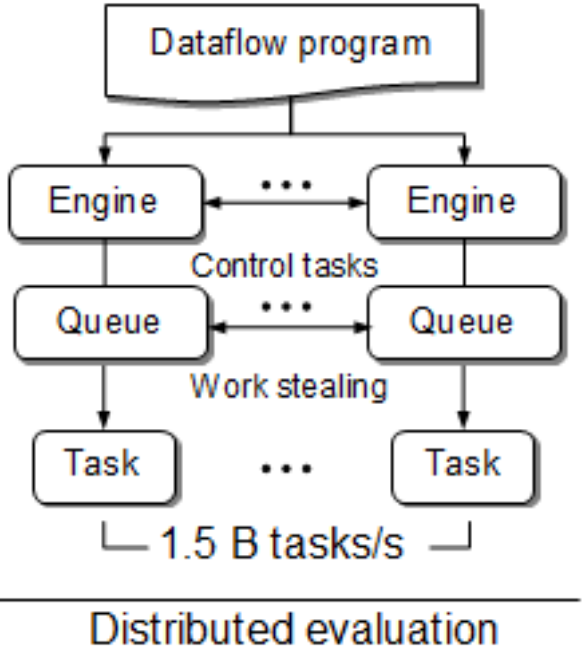
- **Swift: A language for distributed parallel scripting.** J. Parallel Computing, 2011
- **Compiler techniques for massively scalable implicit task parallelism.** Proc. SC, 2014

CENTRALIZED EVALUATION IS A BOTTLENECK AT EXTREME SCALES

Had this (Swift/K):



Now have this (Swift/T):



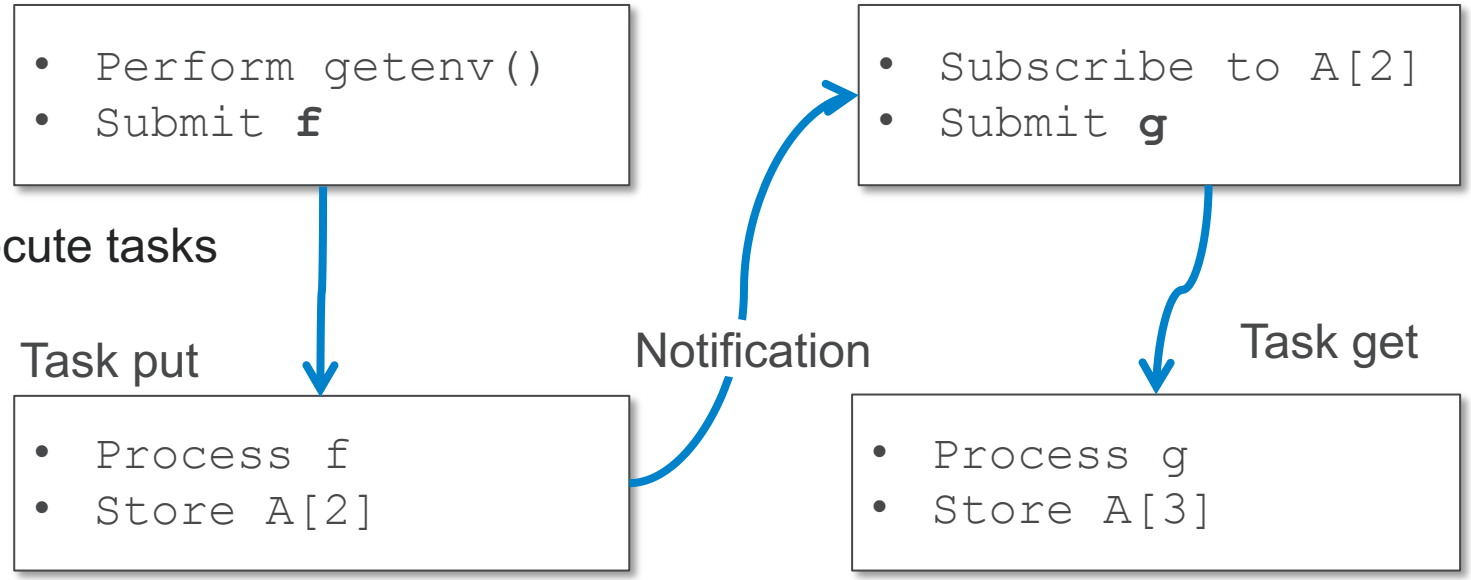
Turbine: A distributed-memory dataflow engine for high performance many-task applications. Fundamenta Informaticae 28(3), 2013

DISTRIBUTED DATAFLOW PROCESSING

- Code

```
A[2] = f(getenv("N"));
A[3] = g(A[2]);
```

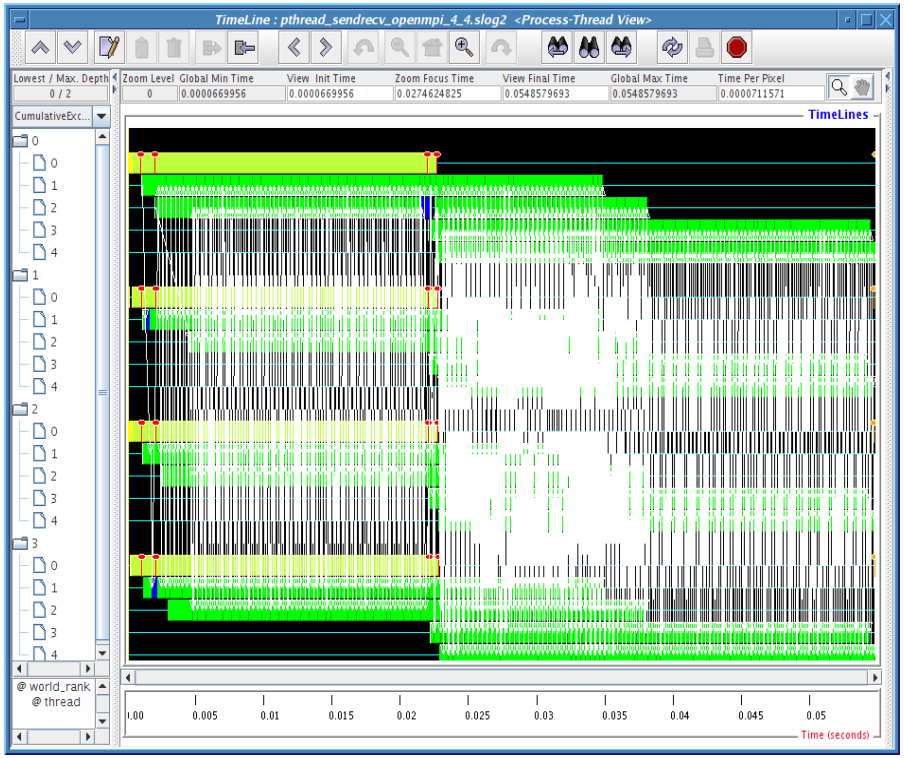
- Engines: evaluate dataflow operations
- Workers: execute tasks



Language features for scalable distributed-memory dataflow computing
Wozniak et al. Proc. Data-Flow Execution Models for Extreme-Scale Computing @ PACT 2014.

MPI: THE MESSAGE PASSING INTERFACE

- Programming model used on large supercomputers
- Can run on many networks, including sockets, or shared memory
- Standard API for C and Fortran; other languages have working implementations
- Contains communication calls for
  - Point-to-point (send/recv)
  - Collectives (broadcast, reduce, etc.)
- Interesting concepts
  - Communicators: collections of communicating processing and a context
  - Data types: Language-independent data marshaling scheme

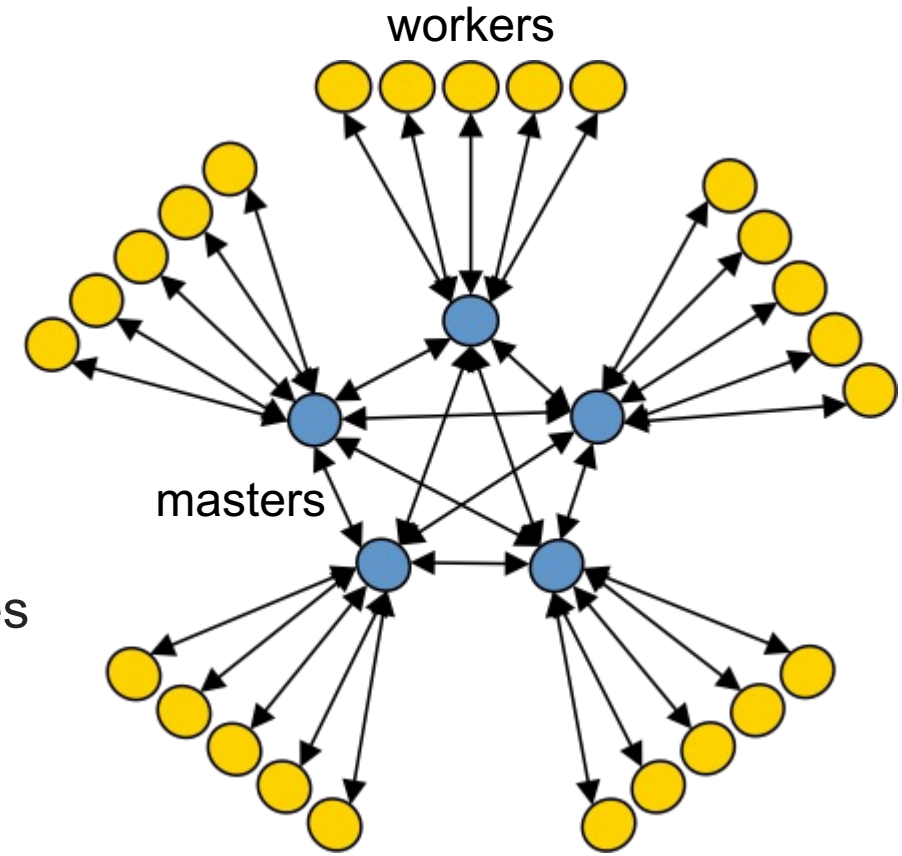


ASYNCHRONOUS DYNAMIC LOAD BALANCER

ADLB for short

- An MPI library for master-worker workloads in C
- Uses a variable-size, scalable network of servers
- Servers implement work-stealing
- The work unit is a byte array
- Optional work priorities, targets, types

- For Swift/T, we added:
  - Server-stored data
  - Data-dependent execution



More scalability, less pain: A simple programming model and its implementation for extreme computing. Lusk et al. SciDAC Review 17, 2010

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)

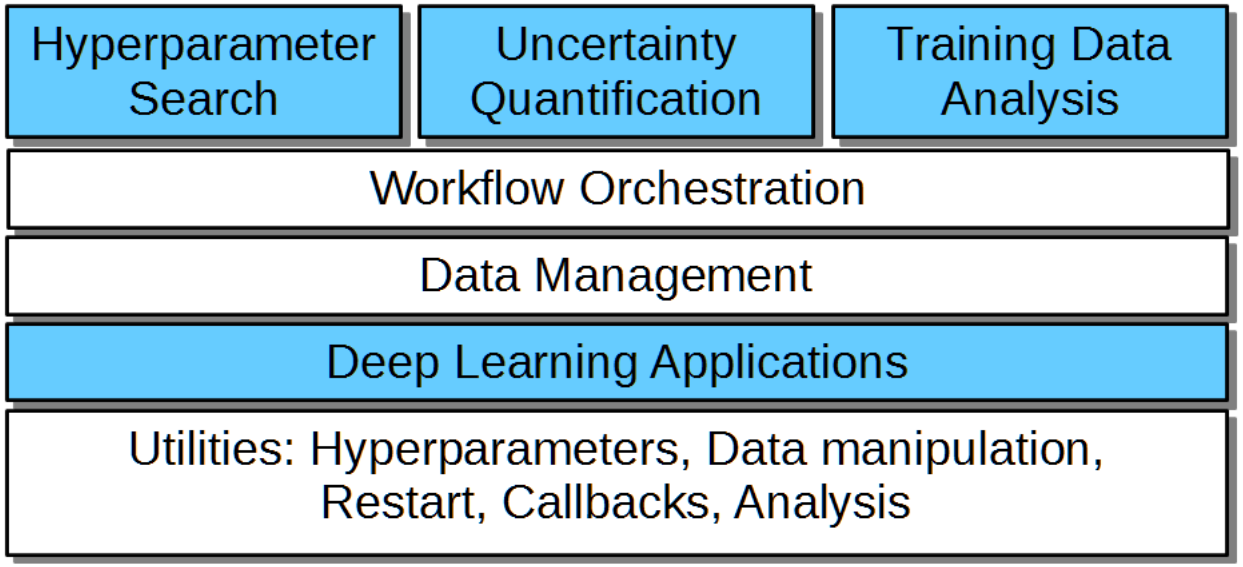
- MPI implementation (MVAPICH, Open MPI)

etc. …

CANDLE/SUPERVISOR OVERVIEW

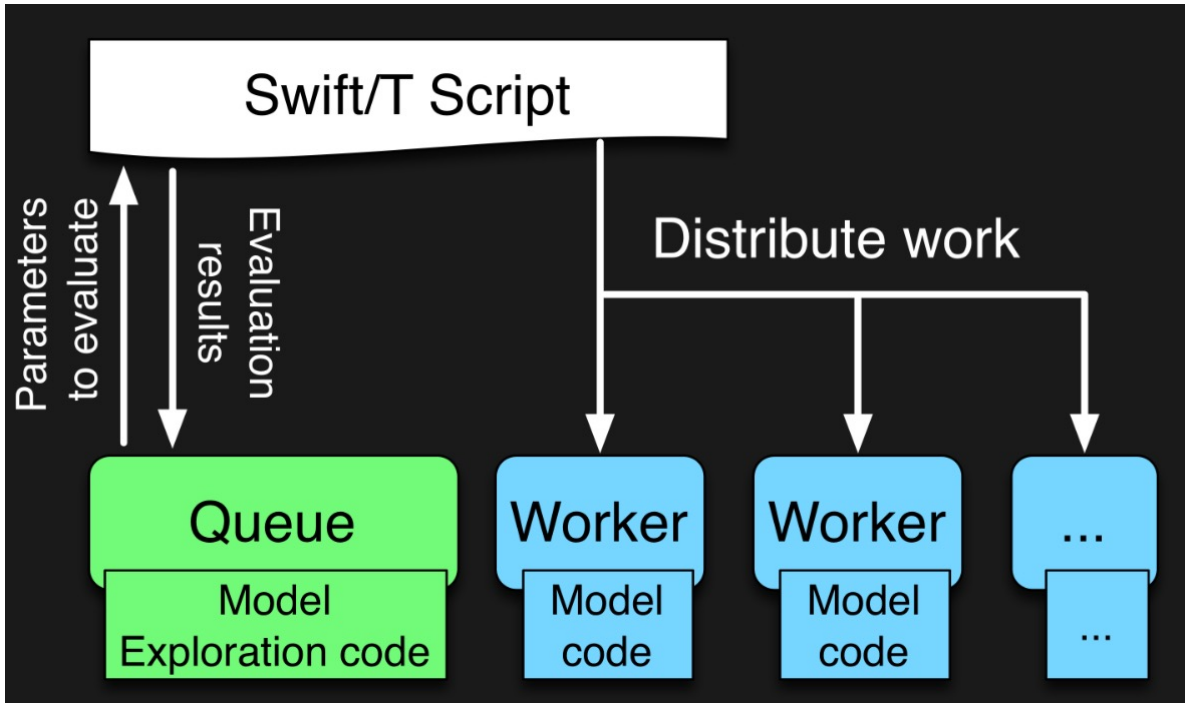
- CANDLE/Supervisor consists of several high-level workflows:
  - Capable of modifying/controlling application parameters dynamically
  - Distribute work across large computing infrastructure, manage progress
- Underlying applications are Python programs that use TensorFlow/PyTorch

- "User code" shown in blue
- "Utilities" shown in white
- New studies would be developed by modifying the blue sections

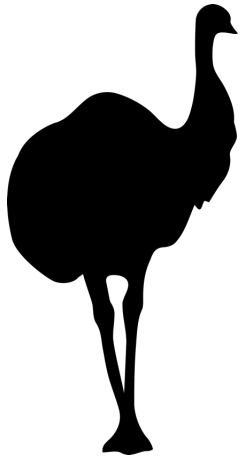


CANDLE/Supervisor: A workflow framework for machine learning applied to cancer research
Wozniak et al. BMC Bioinformatics 19(18), 2018.

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**

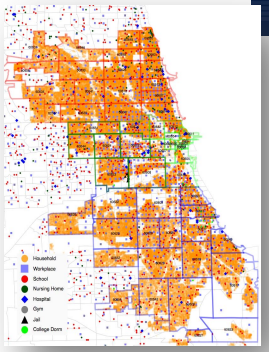


ML-DRIVEN WORKFLOWS FOR COVID-19 POPULATION MODELING

ACM Gordon Bell Special Prize
for High Performance Computing-Based COVID-19 Research
Finalist 2020

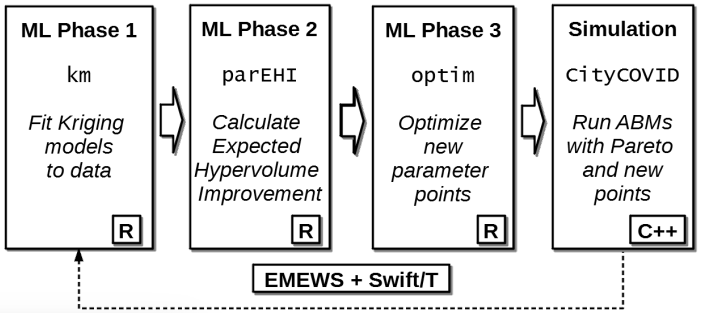
ML-driven parameter fitting monitored by DB

- Large ensemble of RepastHPC-based C++/MPI simulations over 10 million simulated persons, modeling COVID transmission in Chicago
- R-based ML libraries (+MKL) propose and evaluate simulation parameters, also using parallel resources, fitting against real-world data reports
- Python-based DB API posts results to Postgres for real-time human monitoring from remote sites
- Transforms Theta into a data pipeline

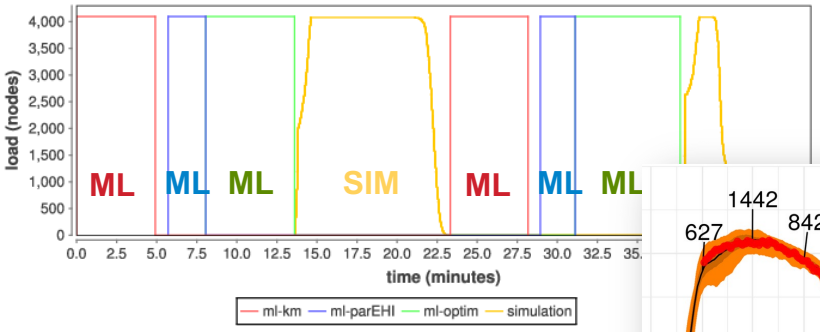


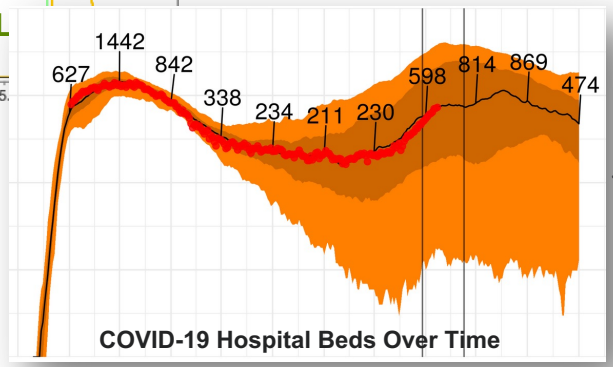
Chicago data tables

CityCOVID HPC Workflow



All phases use full system resources





Actionable epidemiological predictions

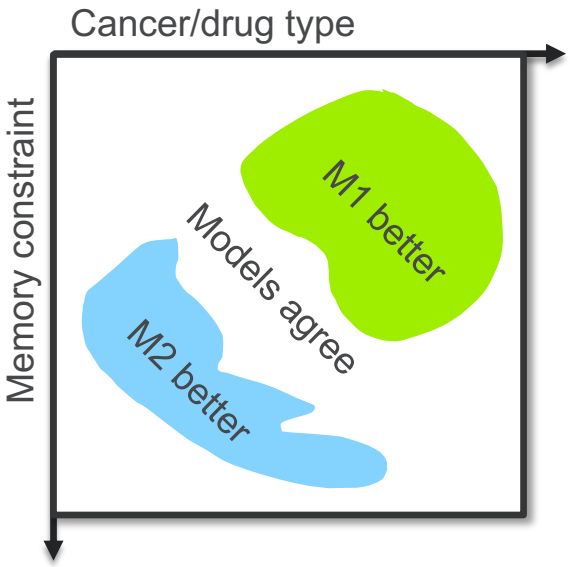
DB

A population data-driven workflow for COVID-19 modeling and learning. Collier, Ozik, Wozniak, Macal, and Binois. Int. J. High Perform. Comput. Appl. 2021.

NEW COMPARISON WORKFLOW: "CMP"

Starting with simple sweep over model behavior

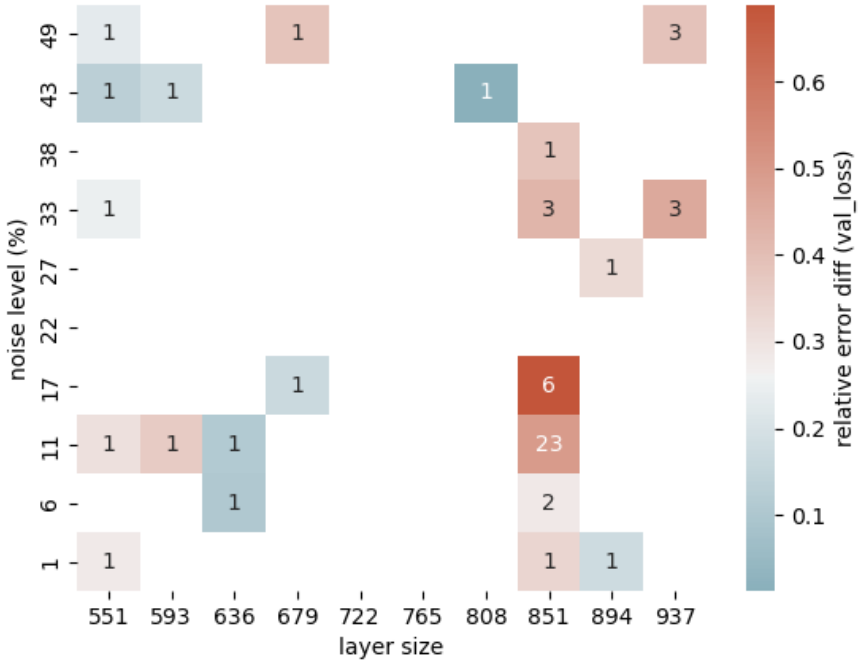
- Initial problem:
  - Want to perform cross-model comparison for two models
  - Want to specify range of hyperparameter values for comparison
  - Need to compare across cross-validation of underlying data
  - Want to compare error values at coarse-grained level
- Approach:
  - Recast model comparison run as a new CANDLE-compliant "model"
    - Accepts two real CANDLE-compliant models
    - Trains and reports differences in errors
    - Can run flat sweep problem above
- Near-future work
  - Flat sweep above produces hundreds to thousands of training runs
  - For larger problems (more hyperparameter comparisons), will need to apply CANDLE parameter exploration workflow



Conceptual example

BENEFITS OF AUTOMATED SEARCH

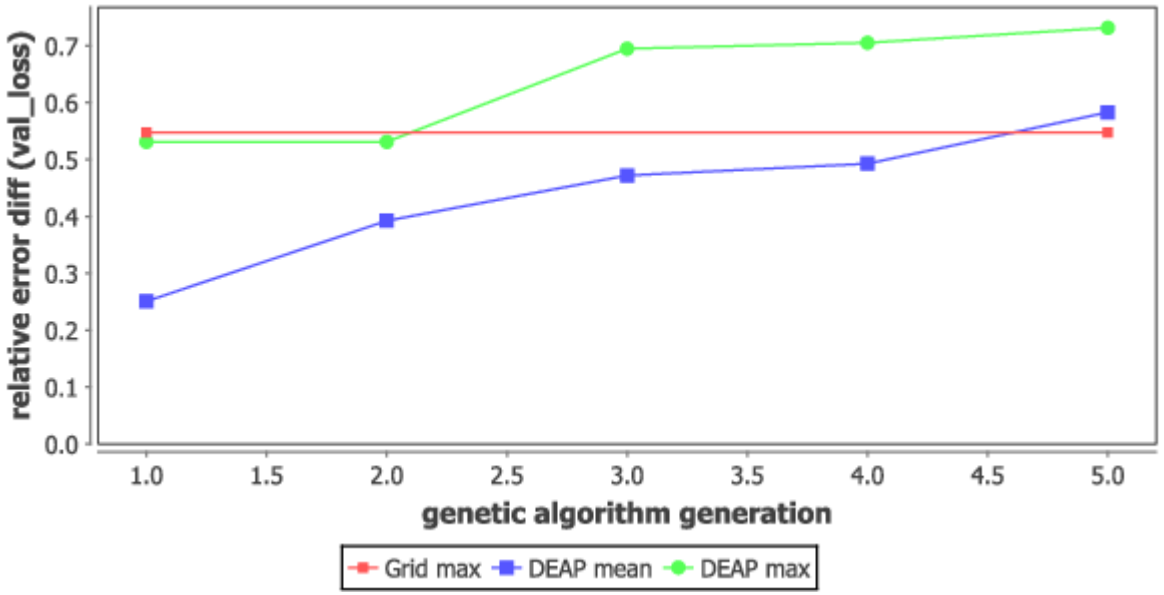
DEAP quickly identifies the best regions and focuses search there



Heat map shows the best value in that cell and the number of samples evaluated in that cell.

Automated searches were run on CELS Lambda, ~20s/epoch on the NVIDIA V100, 10 epochs.

POPULATION PERFORMANCE VS. GRID SEARCH



Grid search: 100 samples DEAP: 16 samples/generation
After 2 generations of DEAP, its smaller population has a better data point than grid search, and after 4 generations the entire population is better

---

An automation framework for comparison of cancer response models across configurations. Wozniak et al. Proc. eScience 2023.

SUMMARY, ONGOING AND FUTURE WORK

Swift/T is automatic parallelism on an MPI-based runtime

- Swift/T is a functional dataflow programming model that translates into MPI
- Allows for workflow-level programming in MPI environments
- Uses conventional programming syntax and constructs
- Offers powerful integration with other languages and MPI applications
- Can be used to address a wide range of applications, particularly in steering simulation and machine learning ensembles
- Plan to extend Swift/T with more MPI use cases and resource managers

QUESTIONS?

THANKS

- Thanks to the organizers

- Code and guides:
  - Swift/T Home: http://swift-lang.org/Swift-T
  - EMEWS Tutorial: http://emews.org

- This research was supported by the Exascale Computing Project (17-SC-20-SC), a joint project of the U.S. Department of Energy's Office of Science and National Nuclear Security Administration, responsible for delivering a capable exascale ecosystem, including software, applications, and hardware technology, to support the nation's exascale computing imperative.