LA-UR-21-27630
Approved for public release; distribution is unlimited.

Title:	Fast Emulation of Expensive Simulations using Approximate Gaussian Processes
Author(s):	Stetzler, Steven Grant Grosskopf, Michael John
Intended for:	Release for Steven to give as presentation at his university and other venues as a student
Issued:	2021-08-02

Los Alamos NATIONAL LABORATORY

Fast Emulation of Expensive Simulations using Approximate Gaussian Processes

Steven Stetzler
Mentor: Mike Grosskopf

7/28/2021

Introduction

- Nuclear Computational Low-Energy Initiative (NUCLEI) collaboration.
  - NUCLEI uses Density Functional Theory (DFT) simulations to predict the structure and binding energies of nuclei over a wide range of proton (Z) and neutron (N) numbers.
- The DFT simulations utilize a particular parameterization of a Skyrme energy density functional called UNEDF1 which depends on **12 free parameters that must be fit to data** (M Kortelainen et al 2014).
- Fitting involves comparing (e.g.) predicted binding energies of nuclei to experimentally measured values.
  - We use only binding energies as observables, but DFT with UNEDF1 will predict structure (shape) observables as well.

UNEDF1 Model Fitting

- Model:

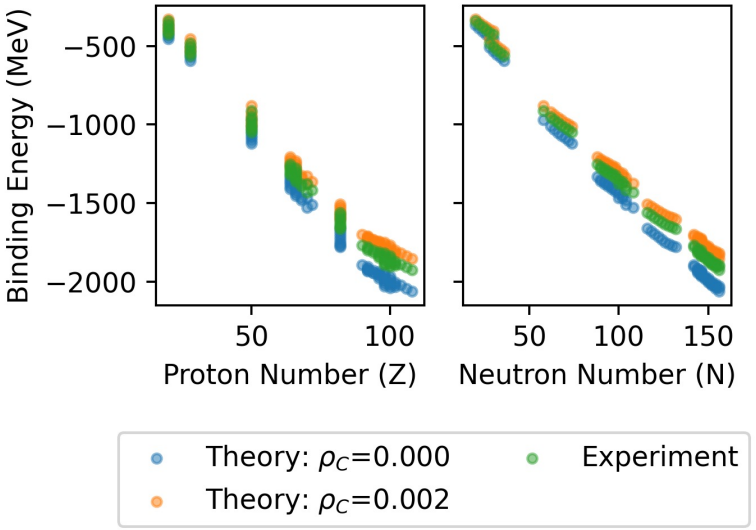
$$y(Z,N) = \eta(Z,N,\theta) + \epsilon$$

Observables (binding energy)

Observation noise: $\epsilon \sim N(0, 2\ MeV)$

DFT simulator $\eta(Z,N,t)$ where θ is the "true" calibrated value of simulation parameters t

- Data: 500 parameter values (t) x 79 (Z,N) Treat output as univariate: 39,500 data points in 14 dimensions



Uncertainty Quantification and the Need for Emulators

- We wish to quantify the uncertainty of the fit parameters θ in a Bayesian framework

$$p(\theta|y) \sim p(y|\theta)p(\theta)$$

- Markov chain Monte Carlo is used to draw samples from $p(\theta|y)$.
  - **Problem**: this would require very many ($O(10^6)$) evaluations of the DFT simulation.
- **Solution**: Build a fast substitute for the full DFT simulation when evaluating the likelihood $p(y|x)$ → Build an Emulator
- A Gaussian process is the emulator of choice (N Schunck et al 2020).
  - It is flexible, non-linear, and naturally quantifies uncertainties. However, GPs are slow for medium-large data $\sim O(10^5)$)

Emulation with Gaussian Process

- Model simulator output statistically given a fixed number of simulator runs:

$$x = (Z, N, t)^T$$
$$\eta(x) \sim N\big(\mu(x), \Sigma(x, x'; k)\big)$$
$$\mu(x) = 0$$
$$\Sigma_{i,j} = k\big(x_i, x_j; \sigma, l\big) = \sigma e^{-\frac{(x_i - x_j)^2}{l^2}}$$

- Predictions at new inputs x^*:

$$\eta(x^* \mid x) \sim N\big(\tilde{\mu}, \tilde{\Sigma}\big)$$
$$\tilde{\mu}(x^*|x) = k(x^*, x)\Sigma_{nn}(x, x)^{-1}(f(x) - \mu(x))$$
$$\tilde{\Sigma}(x^*|x) = k(x^*, x^*) - k(x^*, x)\Sigma_{nn}(x, x)^{-1}k(x, x^*)$$

Exact Gaussian Process is too slow

- Predictions at new inputs x^*:

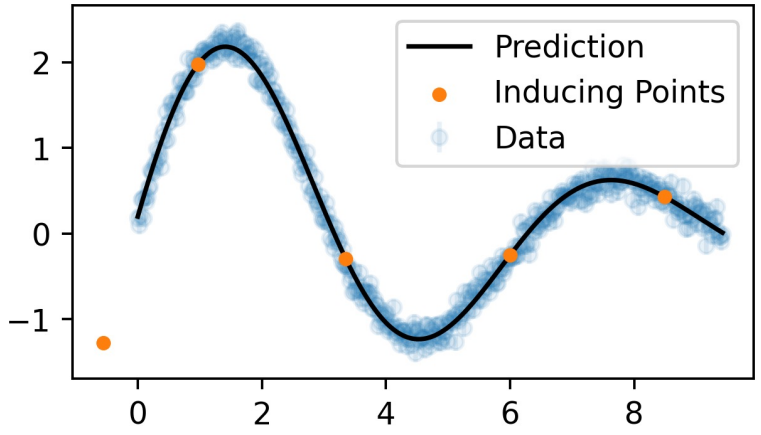
$$\eta(x^* \mid x) \sim N(\tilde{\mu}, \tilde{\Sigma})$$
$$\tilde{\mu}(x^*|x) = k(x^*, x)\Sigma_{nn}(x, x)^{-1}(f(x) - \mu(x))$$
$$\tilde{\Sigma}(x^*|x) = k(x^*, x^*) - k(x^*, x)\Sigma_{nn}(x, x)^{-1}k(x, x^*)$$

- $\Sigma_{nn}(x, x)^{-1}$ requires $O(N^3)$ operations and $O(N^2)$ memory!
  - $39500^3 \approx 61.6\times10^{12}$ operations and $39500^2 \approx 11.6$ GB of memory

- What makes the data "big" is our choice to treat the multivariate output of the simulation (79 values of Z, N) as univariate
  - Previous work has incorporated multivariate output with a PCA approach, reducing input data size to only 500 and using several exact GPs to predict PCA weights (N Schunck et al 2020).

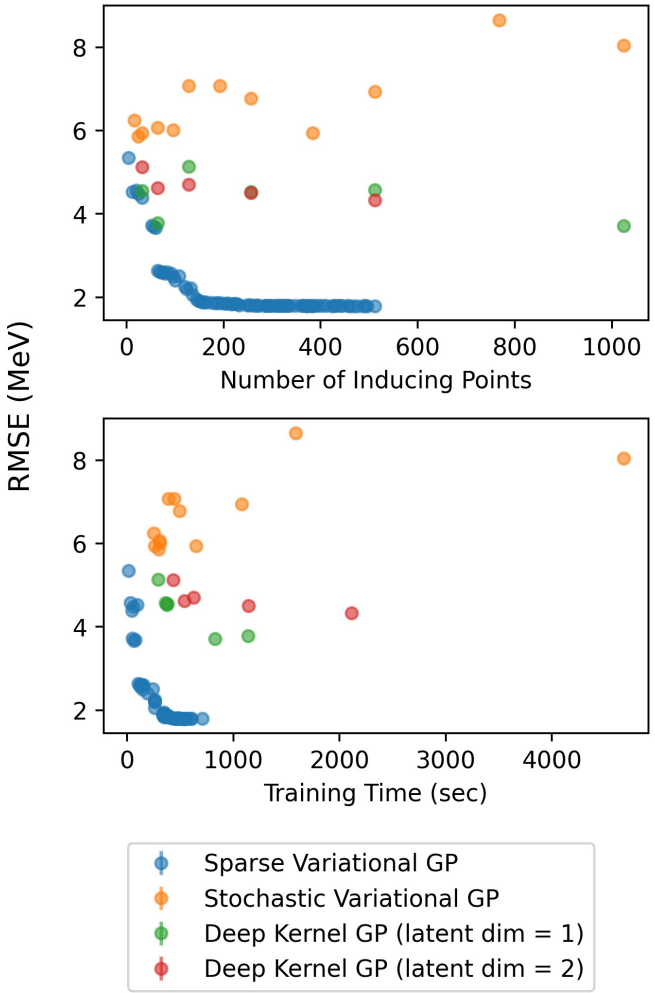
Approximate Gaussian Process

- Sparse Variational GP:
  - Use m “inducing points” to represent the large dataset as a smaller one. Learn inducing point locations in a variational manner (M Titsias 2009).
  - Speedup! $O(m^3)$ ops. and $O(m^2)$ memory!
- Stochastic Variational GP:
  - Learn inducing point outputs as well: enables mini-batch learning through stochastic gradient descent (J Hensman et al 2015).
- Deep Kernel Learned GP:
  - Neural network to construct a non-stationary kernel and use Structured Kernel Interpolation for fast matrix inverse (A Wilson et al 2016).



Results: Accuracy

- Analyzed performance of these models
- Sparse Variational GP outperformed other options
  - Stochastic Variational GP: likely the optimization problem became too hard when additionally learning GP outputs/covariance
  - Deep Kernel GP: perhaps too hard of an optimization problem with learning neural network parameters
- “Best” model
  - Sparse GP with 152 inducing points
  - Approximately balances trade-off between accuracy and prediction time

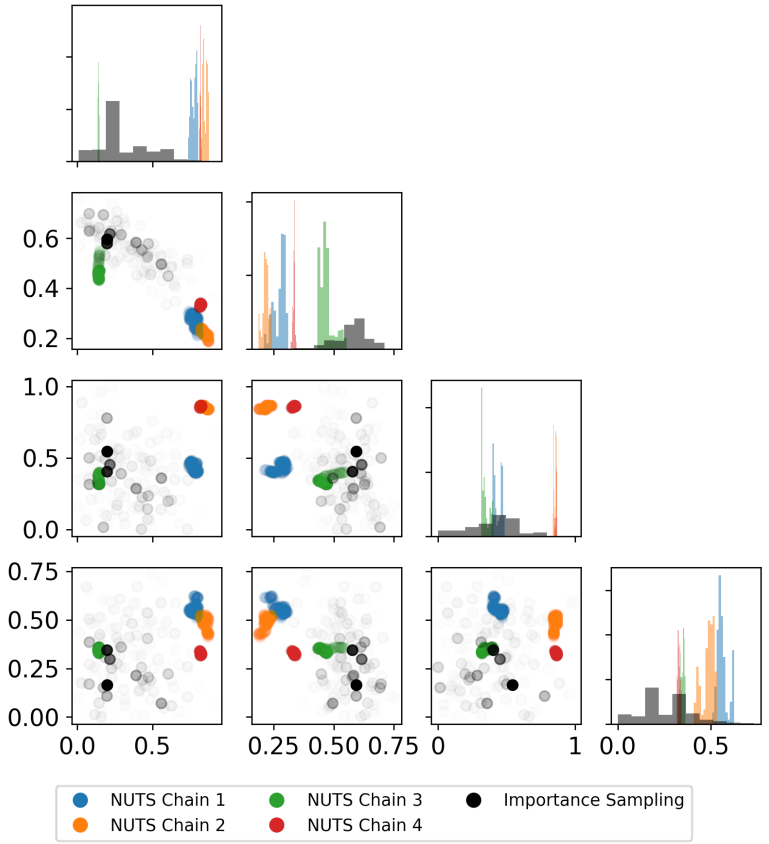


Results: Calibration

- Used best performing Sparse Gaussian Process for calibration, finding best fit simulation parameters θ
  - 152 inducing points
- Bayesian approach: $\theta \sim Uniform(0, 1)$
- Three approaches to sampling from posterior
  - Metropolis Hastings with tuned proposal distribution
  - NUTS/HMC with Pyro python package (derivatives enabled through autodiff)
  - Importance sampling: draws from prior are weighted by likelihood
- **Goal**: compare calibration using Sparse GP emulator to calibration with PCA approach (used NUTS in Stan)

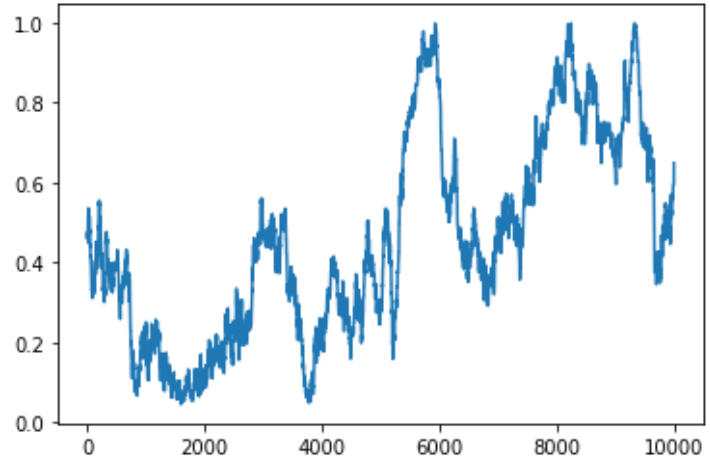
Results: Calibration

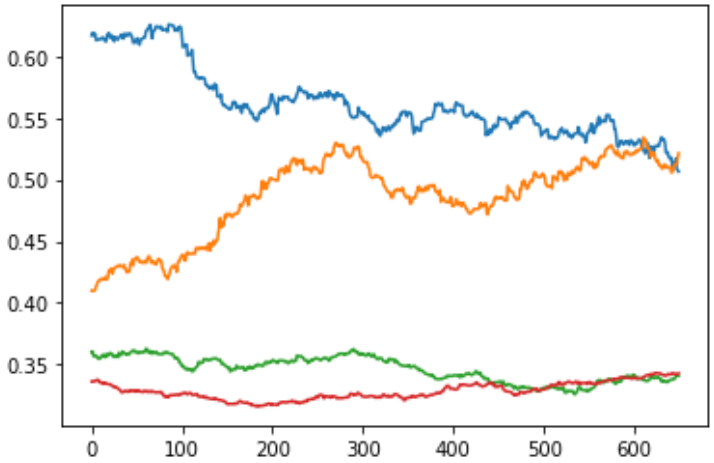
- All attempts to draw samples from posterior performed poorly
- Importance sampling
  - Very few points have weight
  - ~20 effective samples from ~3.5 million samples from prior
- NUTS sampler
  - Chains didn't mix (4 independent chains)
  - Pyro implementation/integration was very slow to sample



Results: Calibration

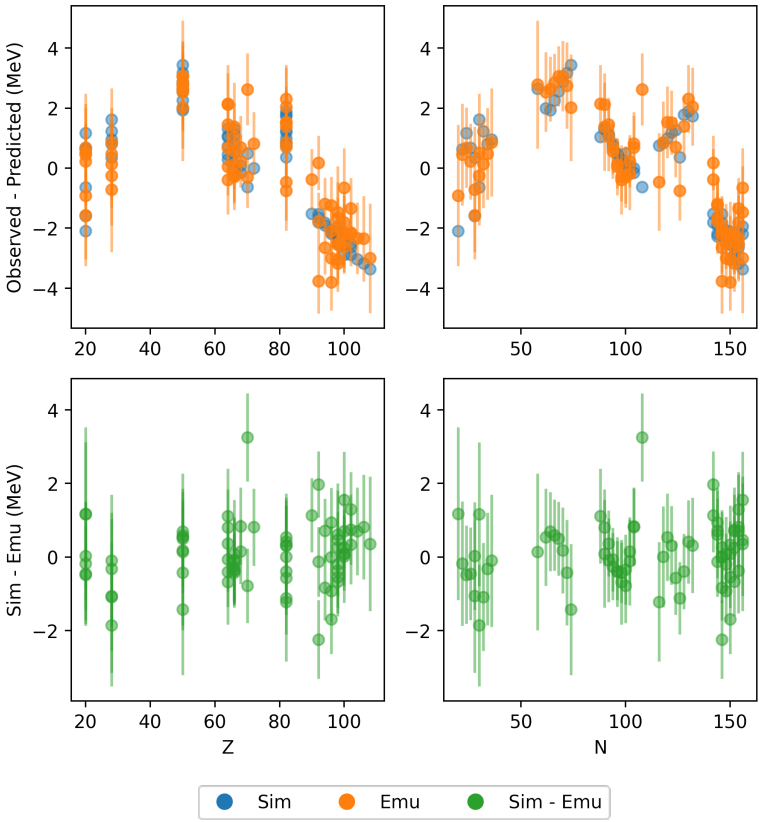
- Trace plots show poor mixing of Markov chains using MH and NUTS sampler
- MH sampler
  - Samples had very high autocorrelation in all dimensions
- NUTS sampler
  - High autocorrelation
  - Chains didn’t mix
  - NUTS worked well with calibration using the PCA-based emulator





Issues: Looking at Residuals

- Investigating why calibration has been a challenge
- One possible reason: emulator is a poor approximation of the simulator and/or the data
  - Residuals show structure, but difference between emulator and simulator is much smaller than difference between emulator/simulator and data



Paths Forward

- Better understand source of sampling issues
  - Comparing PCA-based emulator and the Sparse GP emulator
    - Is there structure the PCA-based emulator is capturing that the Sparse GP emulator isn't?
  - Further exploration of posterior geometry to understand issues with NUTS
- Implementation of discrepancy model: model the error between simulator/emulator and observed data

$$y(Z,N) = \eta(Z,N,\theta) + \delta(Z,N) + \epsilon$$

Software Produced

- `nuclei`
  - Python package for training and testing various approximate GPs
- `darwin_jupyterhub`
  - Python package for setting up a JupyterHub to access notebooks on Darwin
- `darwin_dask`
  - Access to Darwin nodes through Dask

`nuclei` module

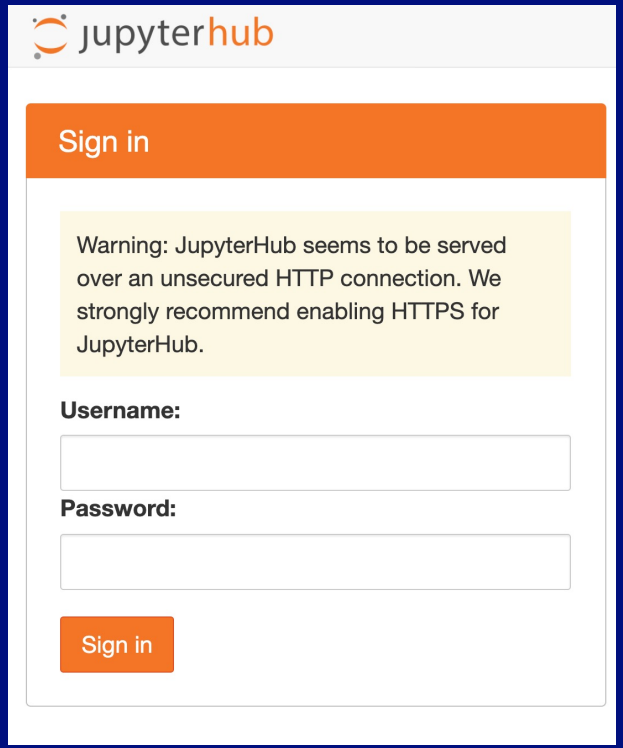
- Train models

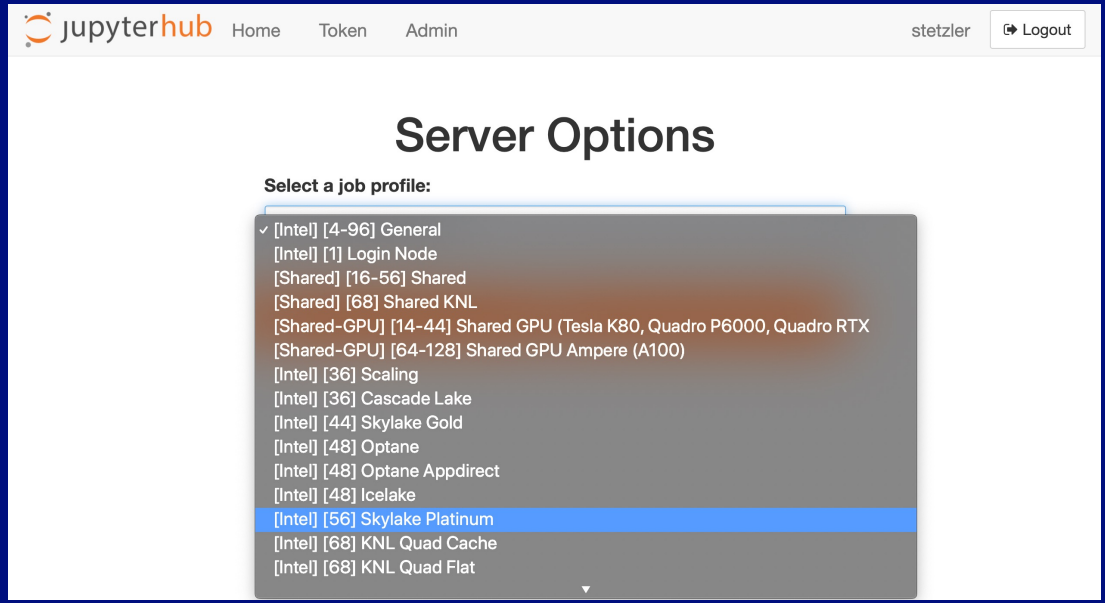
```
(nuclei) [stetzler@cn701 nuclei_summer]$ train-sparse-gp 64 \
> --threads 32 \
> --save-model-to $PWD/models/sparse_gp_64_inducing_points.pth \
> --num-epochs 5
Iter = 0, Loss = 258388.09603184176
Iter = 1, Loss = 69.92642153697821
Iter = 2, Loss = 25.78518403224415
Iter = 3, Loss = 16.146457302700284
Iter = 4, Loss = 12.058695938404439
training_time: 5.665445327758789
```

- Test models

```
(nuclei) [stetzler@cn701 nuclei_summer]$ test-gp \
> $PWD/models/sparse_gp_64_inducing_points.pth \
> --threads 32
rmse: 0.0051027920377533325
test_time_no_cache: 0.7319824695587158
test_time_with_cache: 0.6481351852416992
```

JupyterHub on Darwin





Dask on Darwin

- Start Cluster

```
(nuclei) [stetzler@cn701 ~]$ darwin start --queue scaling
Dask cluster running using [0-16] nodes from partition scaling
Scheduling 1 tasks per node.
Monitor cluster status at: tcp//192.168.100.158:8787
Connect with:
>>> from dask.distributed import Client
>>> client = Client('tcp://192.168.100.158:45136')
I'm sending scheduler logs to: /tmp/darwin_dask_scheduler_16274102530644764.log
I'm sending cluster logs to: /tmp/darwin_dask_cluster_16274102530644764.log

distributed.scheduler - INFO - Clear task state
distributed.scheduler - INFO -   Scheduler at: tcp://192.168.100.158:45136
distributed.scheduler - INFO -   dashboard at:                     :8787
```

- List Clusters

```
(nuclei) [stetzler@cn701 ~]$ darwin list
partition      nodes    processes  scheduler_addres            dashboard_address           cluster_pid
-----------  -------  -----------  --------------------------  -------------------------  -------------
scaling            0            1  tcp://192.168.100.158:45136  tcp//192.168.100.158:8787         276040
```

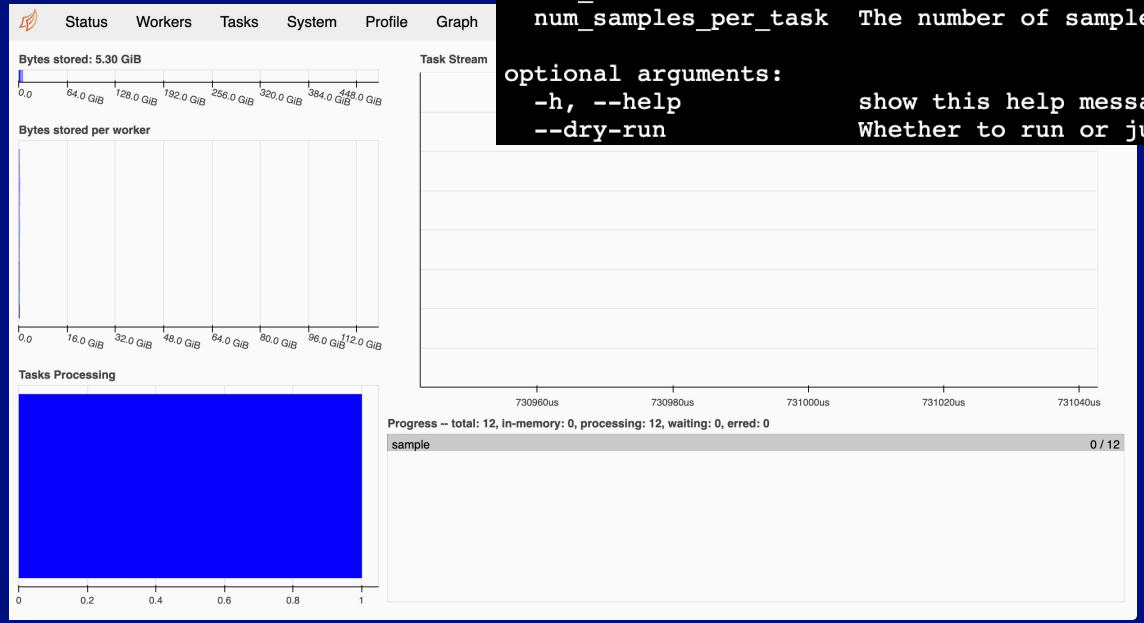
Dask on Darwin

- Connect and Run

```
(nuclei) [stetzler@cn701 nuclei_summer]$ python smc.py --help
usage: smc.py [-h] [--dry-run] cluster_address model_path num_tasks num_samples_per_task

positional arguments:
  cluster_address       The Dask cluster address to submit tasks to.
  model_path            The model to use for likelihood calculations.
  num_tasks             The number of tasks to run.
  num_samples_per_task  The number of samples per task.

optional arguments:
  -h, --help            show this help message and exit
  --dry-run             Whether to run or just print what will be done. (default: False)
```



Conclusions

- Assessing the capability of approximate GP emulators to balance emulator accuracy with computational speed to facilitate improved UNEDF1 calibration
  - Sparse GPs are straightforward to train and accurate
  - Calibration is not straightforward with MCMC (using MH or HMC/NUTS)
- Produced reusable software for continuing and building on this work as well as accessing and using Darwin cluster compute resources