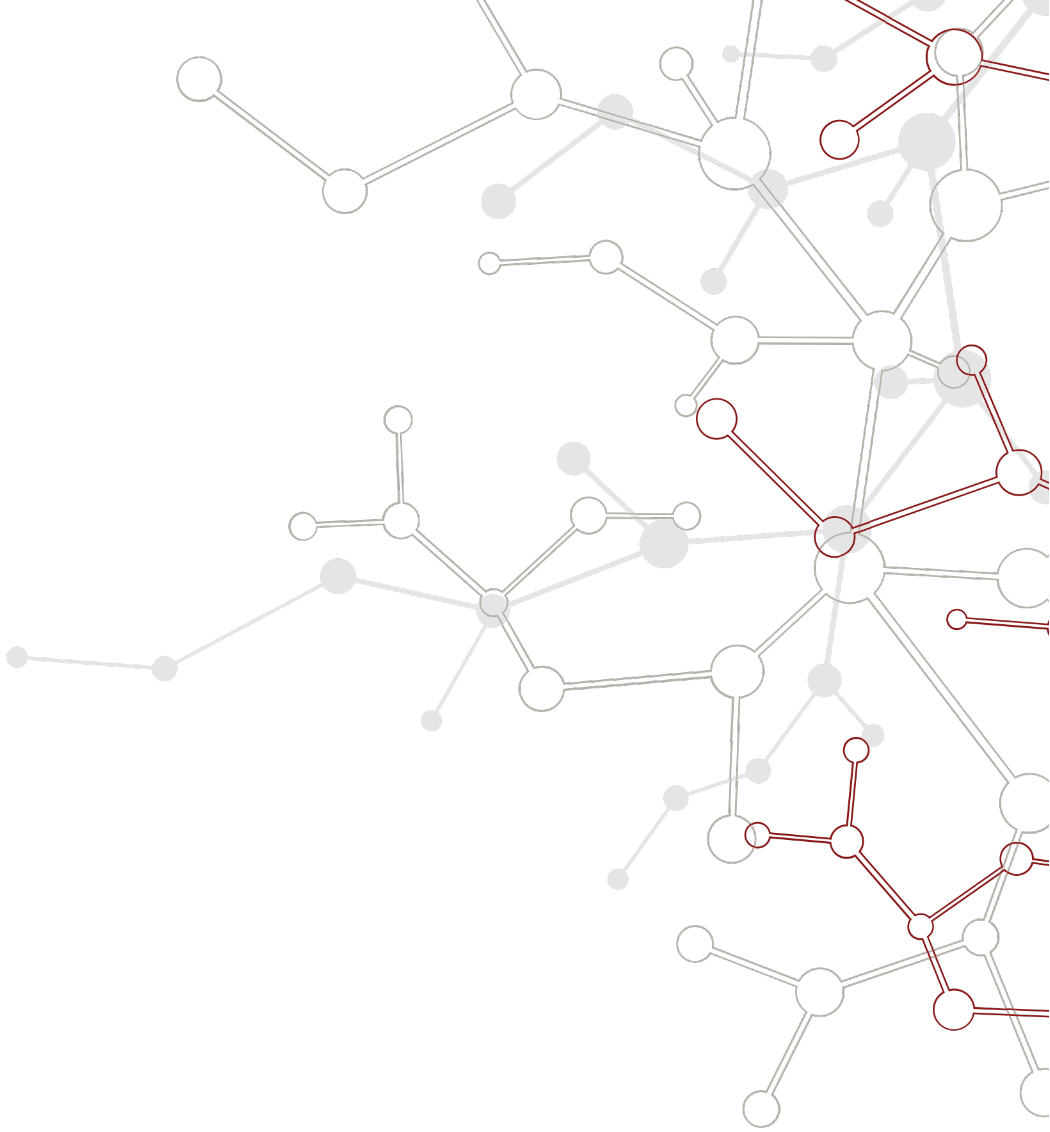
Distributed Tasking in Python with Legion



Elliott Slaughter
Staff Scientist, SLAC National Accelerator Laboratory

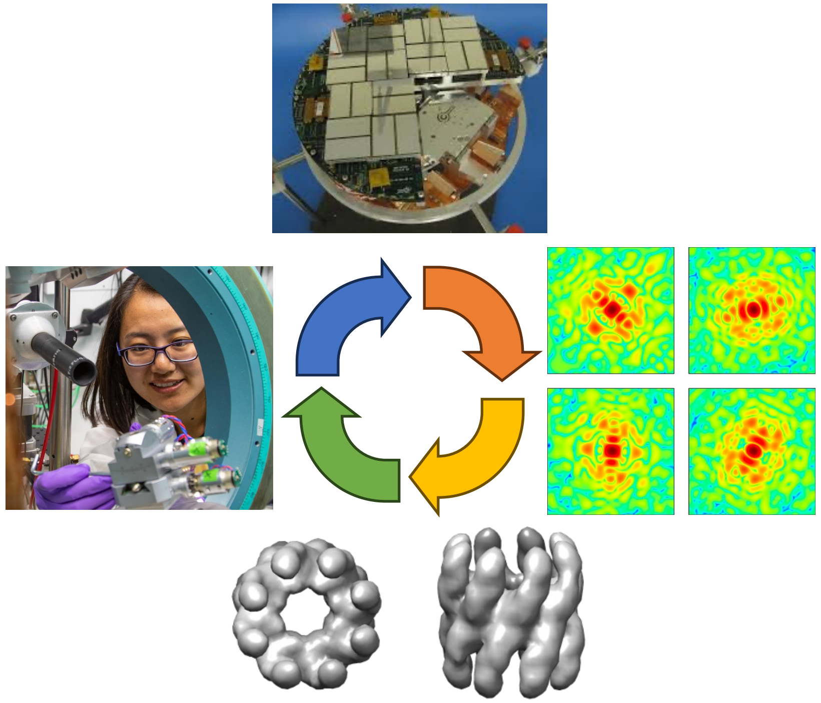
Python HPC BoF, SC'24
November 20, 2024

SLAC NATIONAL ACCELERATOR LABORATORY

Stanford University | U.S. DEPARTMENT OF ENERGY

A Brave New World of Distributed Python Programming

MHz Science



Up to 1 MHz repetition rate

Up to 100 Gb/s data rate

Diverse, unique science

Experiment in the loop

Users are not HPC experts

Scalable Software

Unmodified Python code

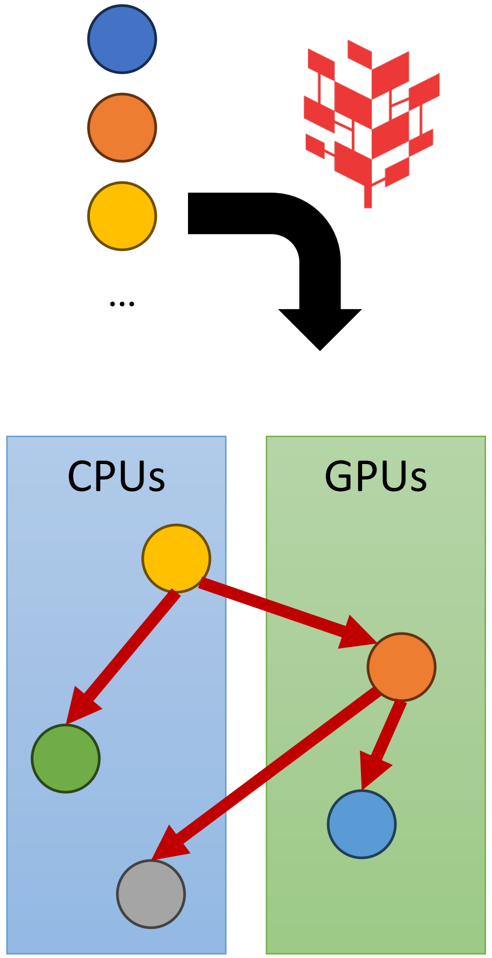
import numpy, scipy, pandas

numpy.fft.fft(...)

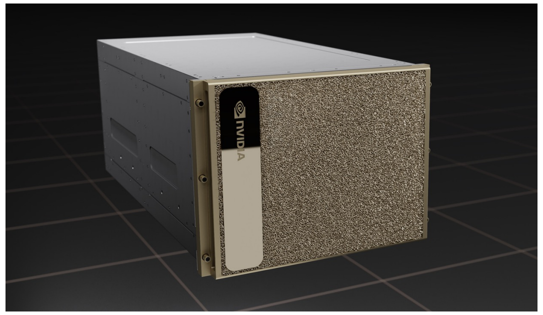
scipy.optimize.curve_fit(...)

pandas.DataFrame(...)

Flexible execution via Legion



Diverse Hardware



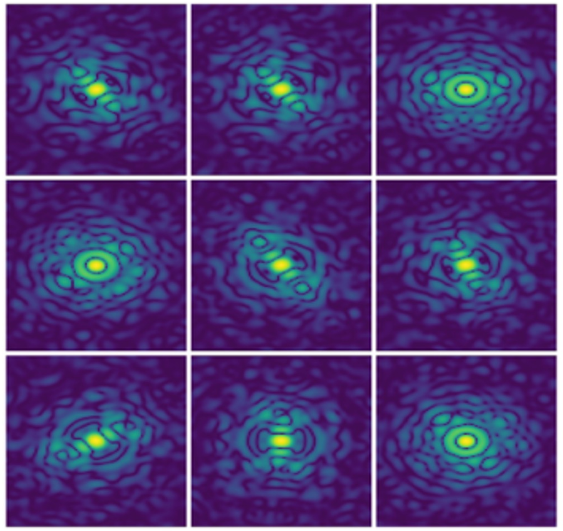
Distributed Tasking in Python: It’s Already Here

An (Incomplete) List of Python Tasking Frameworks for HPC

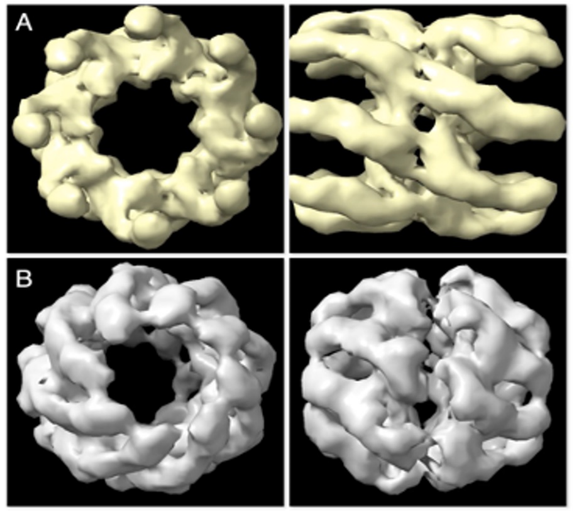
- Pygion (https://legion.stanford.edu/pygion/)
  - Legion programming model in Python
- cuPyNumeric / Legate (https://developer.nvidia.com/cupynumeric)
  - Write NumPy, run automatically on clusters of CPUs and GPUs
- FlexFlow (https://flexflow.ai/)
  - Drop-in PyTorch, Keras, ONNX interface with additional optimizations
- Charm4py (https://charm4py.readthedocs.io/)
  - Charm++ programming model in Python
- PyCOMPS (https://docs.idmod.org/projects/pycomps/)
  - Workflow orchestration in Python
- And others….

SpiniFEL: Single Particle Imaging for XFEL

Input: X-ray diffraction images



Output: 3D reconstruction of each protein conformation

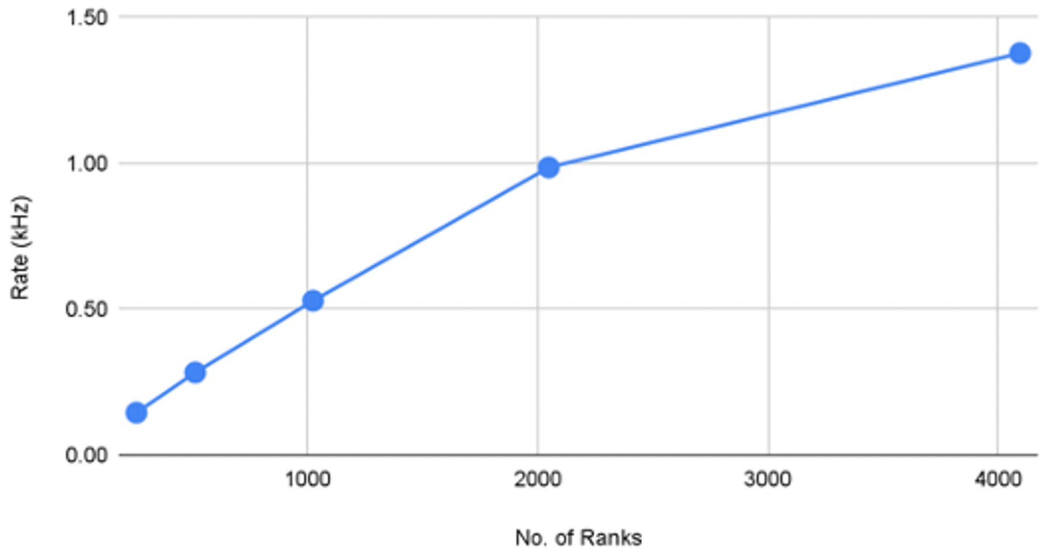


Design Principles

- Kitchen sink software design
  - If it exists, and it works, use it
  - NumPy, CuPy, Numba, hand-written CUDA, third-party CUDA libraries
- Pygion tasking as the orchestration layer

Lessons Learned

- Tasking layer was a non-issue
  - No production issues due to Pygion
- Kitchen sink approach caused porting issues



Weak scaling on up to 4096 GPUs on Frontier

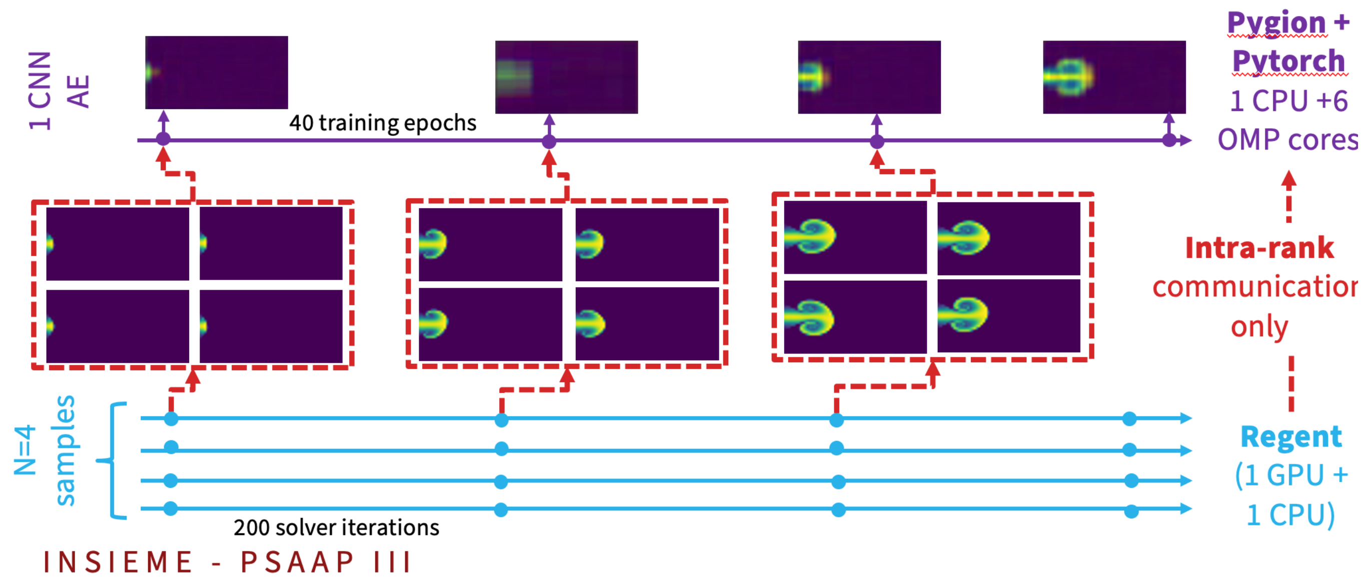
[Mirchandaney et al., WAMTA 2024]

nHTR: Sidecar Python Analysis

Pygion as an Interface for Fast Scripting

- nHTR runs main simulation in high fidelity on GPU using Regent
- Simultaneously using PyTorch (via Pygion) for online training of ML models during the ensemble run on free CPU cores
  - Otherwise unused by the main application
  - Runs asynchronously without slowing down the main simulation
  - Can return results asynchronously to the main simulation to drive simulation parameters

Case Study: online PyTorch training in nHTR



[Laurent and Maeda, APS 2022]

Thoughts, Challenges, Provocations

- Distributed tasking systems for Python already exist
  - Maybe not marketed widely enough?
- Need to be better integrated across frameworks/libraries
  - Even when backed by the same runtime system
- Python’s existing ecosystem doesn’t necessarily lead to portability or distributed support
  - Easy to get “stuck” if you make the wrong bet
- Need better modular Python code analysis and code generation tools
  - Numba is overfitted to the problem it solves, code generation is not flexible enough