Steering Workflows with Artificial Intelligence

Cleared for public release

EXASCALE COMPUTING PROJECT

Logan Ward
Computational Scientist
Data Science and Learning Division
Argonne National Laboratory

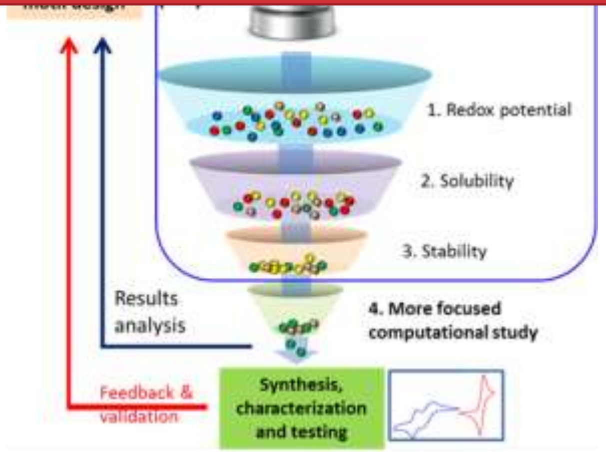
16 April 2025

National Nuclear Security Administration

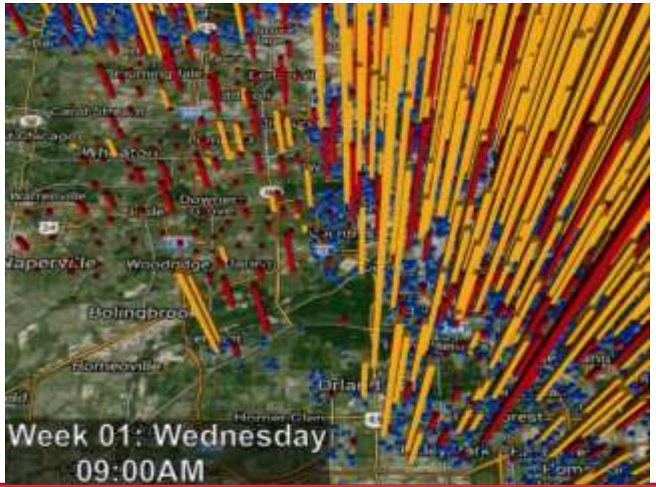
U.S. DEPARTMENT OF ENERGY | Office of Science

“Computational Campaigns” are a common tool

High-Throughput Design

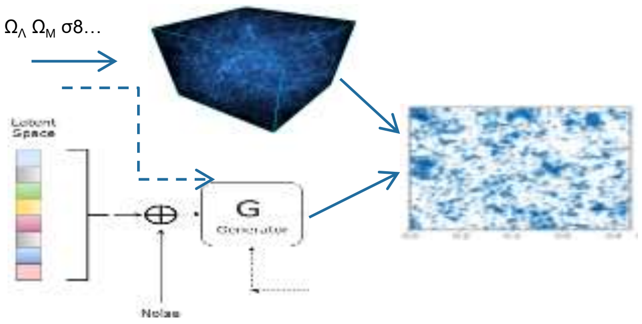


Source: Cheng et al. JPCL (2015)

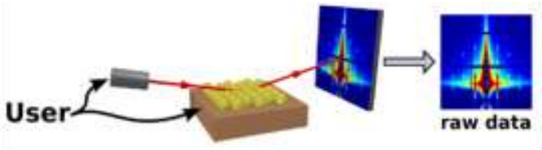


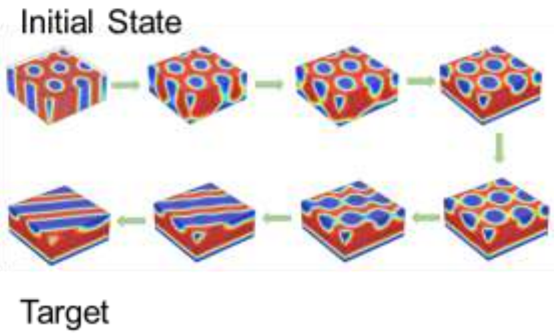
Digital Twins, Forecasting

Parameter Estimation



How do we adapt these approaches to Exascale computing?



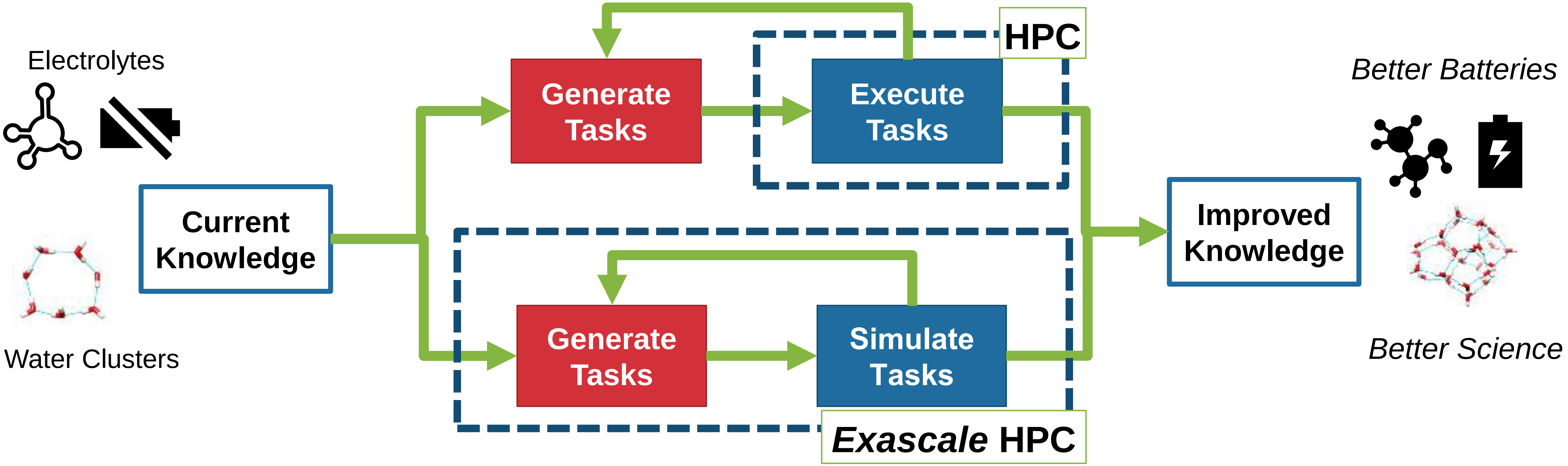


Learning Control Policies

Expanding Computational Campaigns to the ExaScale

Current Model: Humans steer HPC, HPC performs simulations *(Months-Years)*

Current Model Won't Scale. Humans are **slow.** Slow decisions, slow to learn



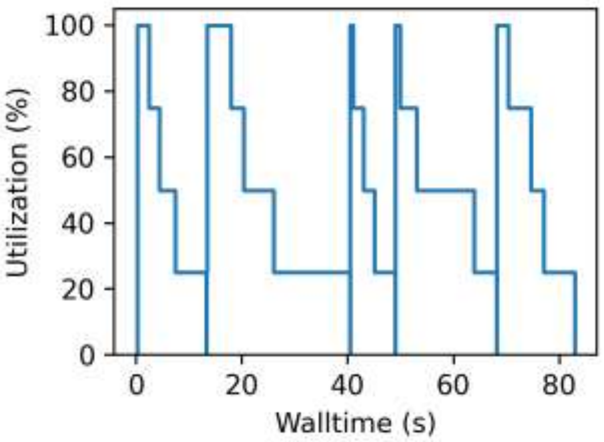
Our goal: HPC steering itself (Days-Weeks)!

EXASCALE COMPUTING PROJECT

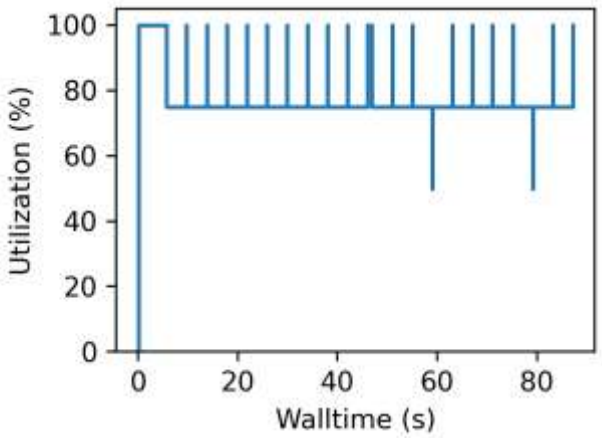
Parallelism makes active learning on HPC difficult

Root Problem: Sequential search is impractical, we must run >1 simulation at once

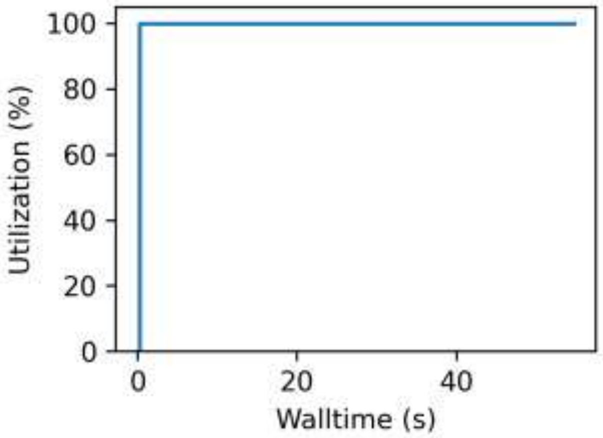
Consider a few parallel strategies…



Wait for N tasks to complete, then pick next batch



Pick new tasks as soon as one completes



Maintain a task queue

↑ Most information per decision
↓ Least utilization

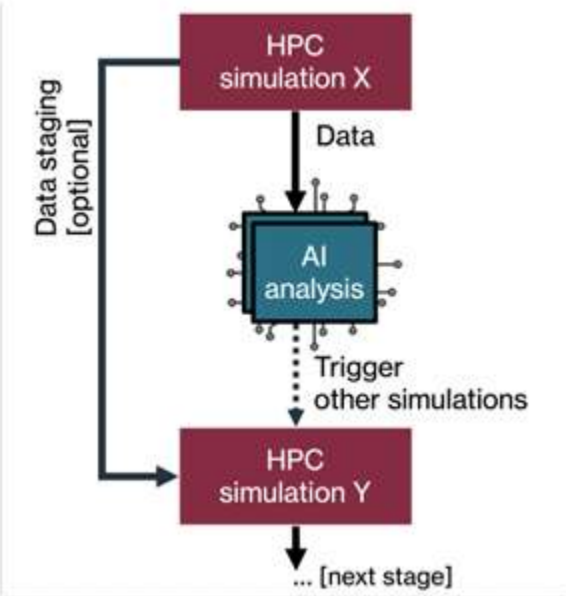
↓ Least information per decision
↑ Greatest utilization

Bottom Line: Active learning on HPC requires intelligent policies

Today's Talk:
- Show the broad scope of AI+HPC for Workflows
- Illustrate one way of building steered workflows
- Encourage a collective ecosystem

What kinds of application patterns exist?

There's some nice work on this by Shantenu Jha’s team



Motif / Scope	Interaction Patterns	Coupling Patterns	Example Use Case
Steering AI improving HPC	- Control and data flow in one direction: data from HPC to AI, control from AI to HPC - One AI to one or many HPC - Optionally human in the loop	- Real-time requirements - Dynamic composition with HPC simulations spawned or terminated on the fly - Usually running in one facility	*AI-out-HPC* - Command-and-control of physical experiments and simulations (e.g. between shots feedback for plasma physics)
Multistage Pipeline AI improving HPC	- Data flows in one direction from HPC to one or many AI or HPC components - AI filters control many HPC simulations - Typically interaction done without human in the loop	- Real-time requirements - Dynamic composition with branching in the workflow based on filters - Running in one facility	*AI-in-HPC and AI-out-HPC* - Large-scale MD simulations using AI sampling of a system with many degrees of freedom
Inverse Design AI improving HPC HPC improving AI	- Control flow from AI to HPC - Multiple HPC simulations and/or instruments sending data to AI (one or many) - Typically interaction done without human in the loop	- Real-time is optional (AI can use existing datasets) - Execution can be concurrent or asynchronous - Running in one facility	*AI-in-HPC* - Materials discovery to address the problem of data sparsity and reduce the need for domain-specific knowledge
Digital Replica AI improving HPC	- Data/control flow in both directions combining exper-	- Real-time requirements with monitoring and visual-	*AI-about-HPC* - Digital twin of a fusion

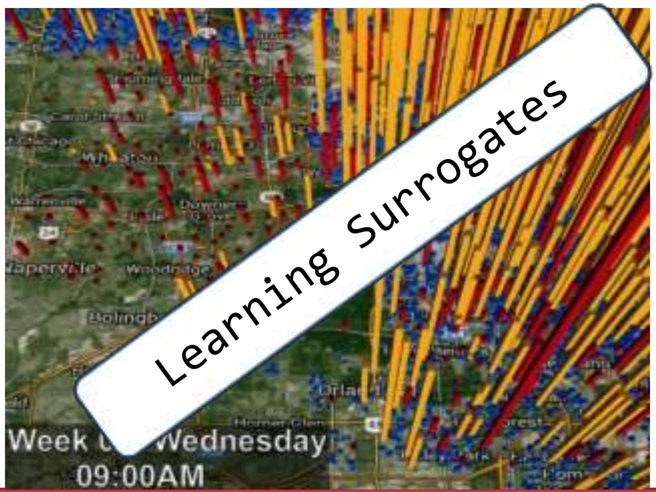
Read this! [Brewer et al. arXiv/2406.14315](https://arxiv.org/abs/2406.14315)

EXASCALE COMPUTING PROJECT

“Computational Campaigns” are a common tool

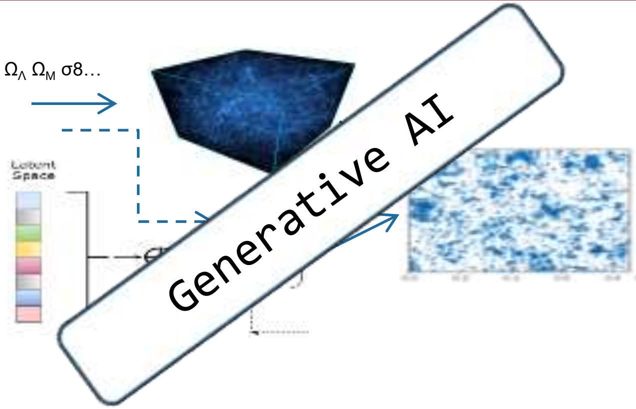
High-Throughput Design

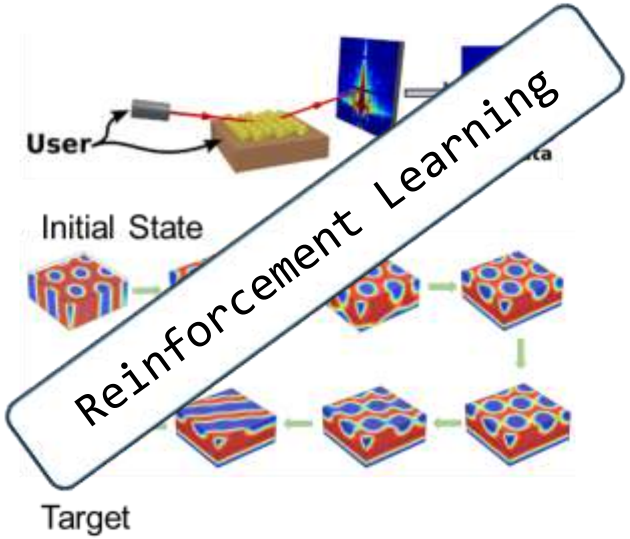
Source: Cheng et al. JPCL (2015)



Digital Twins, Forecasting

Parameter Estimation





Learning Control Policies

Our Approach: Colmena

Ward et al. MLHPC @ SC'21

What kind of “intelligence” goes into steering applications

Observation: We have many policy ideas…

– *Submit a new simulation* ***once another completes*** Event-triggered

– *Retrain a model* ***after 8 successful computations*** Conditional logic

– ***Allocate more nodes to inference*** *after models finish training* Resource management

and others are possible.

Solution: We need a way of programming ***agents*** to encode such policies

1. Agents must be able to react to events
2. Allow the agent to hold state
3. Ability to re-allocate resources between pools
4. Separate agent from *how to run tasks* and *interface with HPC*

Building a Colmena app: Defining the “tasks” and **“thinker”**

Key points:

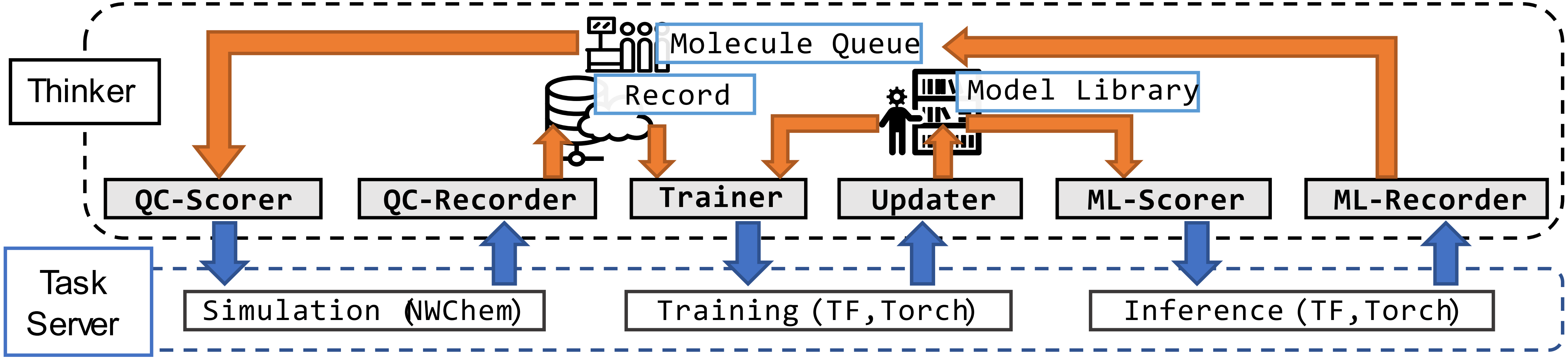
1. Subclass the “BaseThinker” abstract class
2. Mark “agent” operations form the policy
3. Communicate with method server via queues
4. Communicate with other via Threading primitives

How does it work:

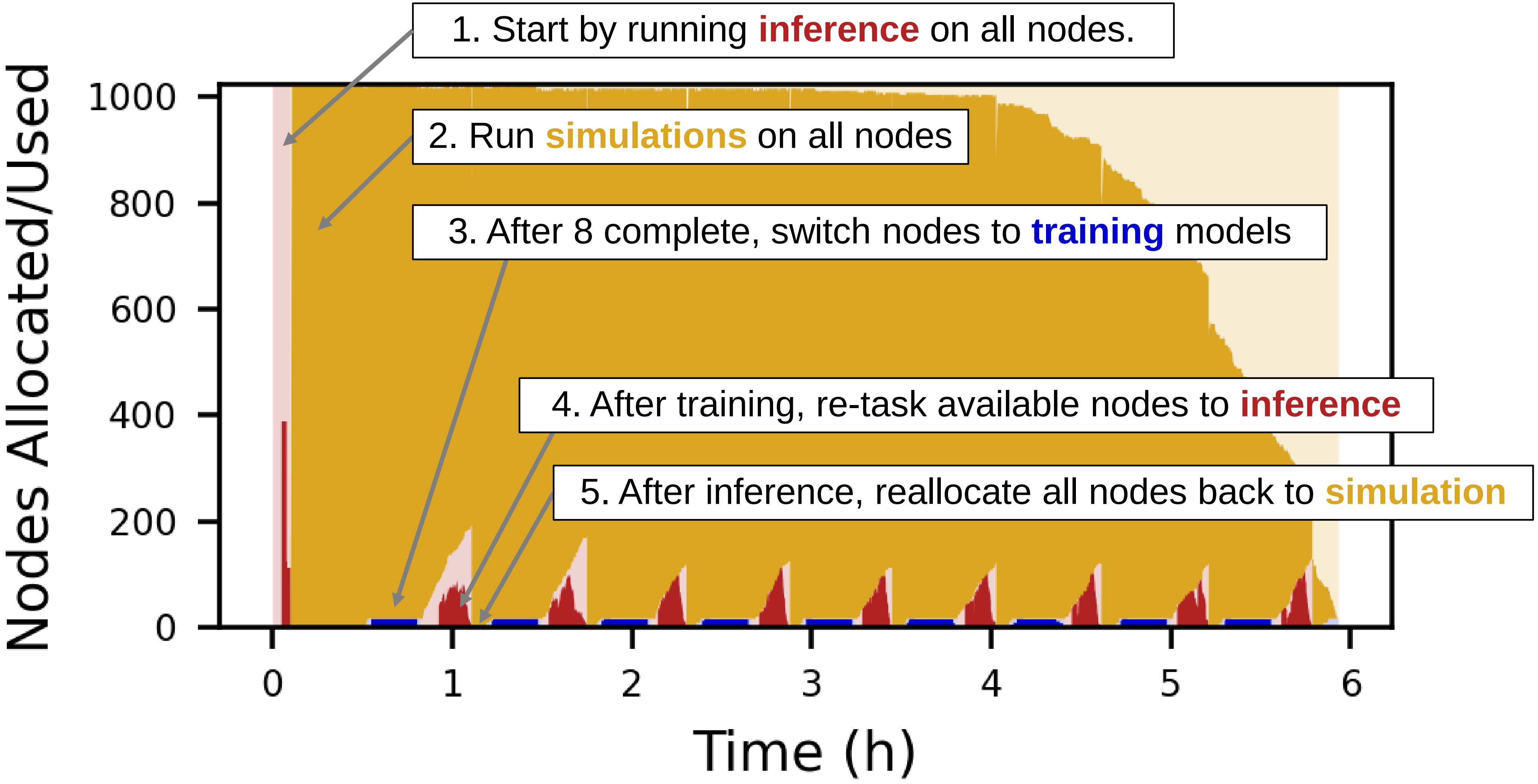
– “.run()” launches all agents

```
class Thinker(BaseThinker):
  def __init__(self, queue):
    super().__init__(queue)
    self.remaining_guesses = 10
    self.best_guess = None
    self.best_result = inf

  @result_processor(topic='simulate')
  def consumer(self, result):
     # Update the best result, check for termination
     if result.value < self.best_result:
    self.best_result = result.value
    self.best_guess = result.args[0]
     self.remaining_guesses -= 1
     if self.remaining_guesses == 0:
    self.done.set()

  @agent
  def producer(self):
    while not self.done.is_set():
   # Make a new guess
      self.queues.send_inputs(self.best_guess,
       method='task_generator', topic='generate')
      # Get the result, push new task to queue
      result = self.queues.get_result(topic='generate')
      self.queues.send_inputs(result.value,
```

What does our “active learning application” look like

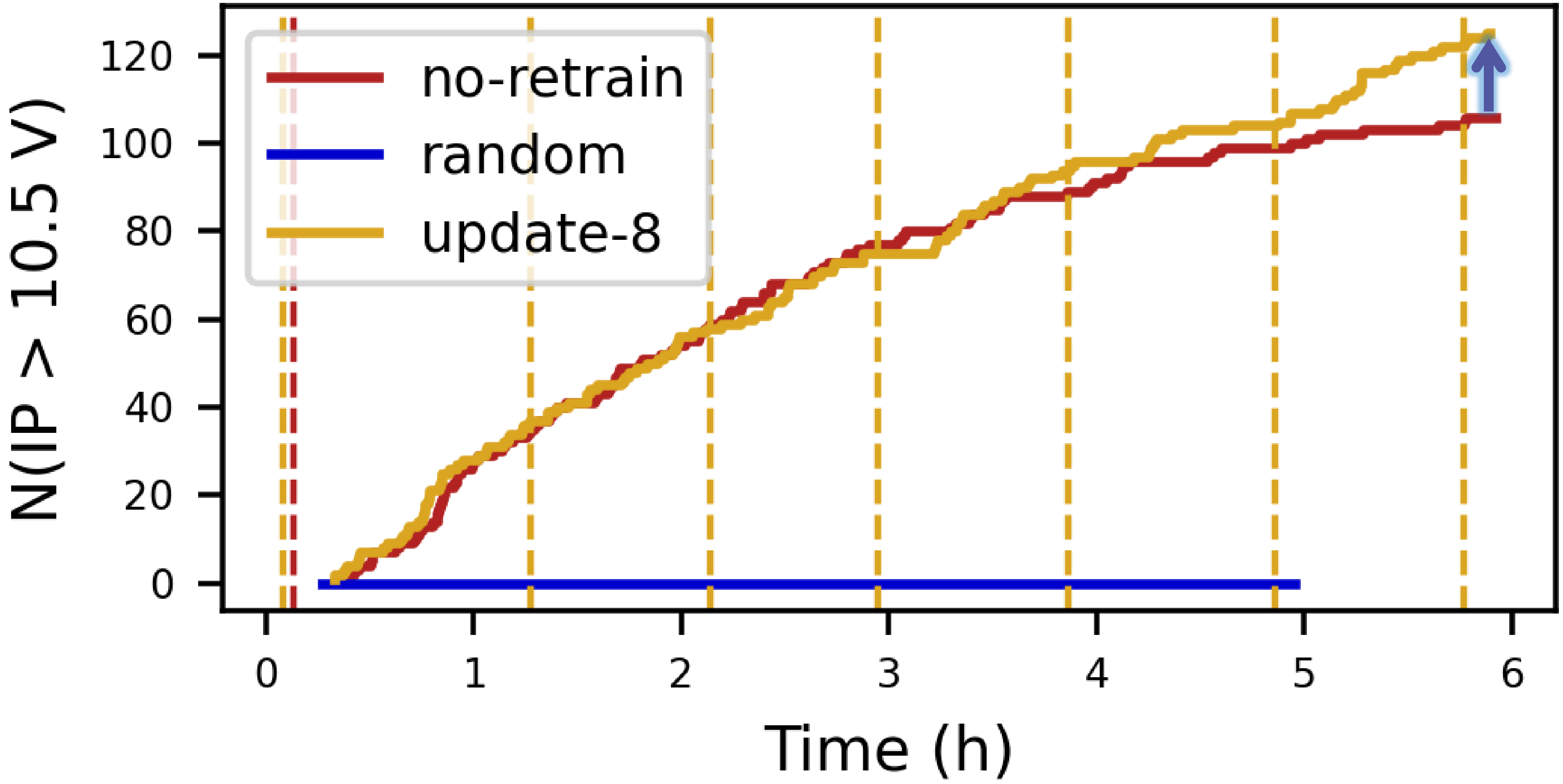


What is the application behavior?



EXASCALE COMPUTING PROJECT

Did the application have good scientific performance? **[Yes]**

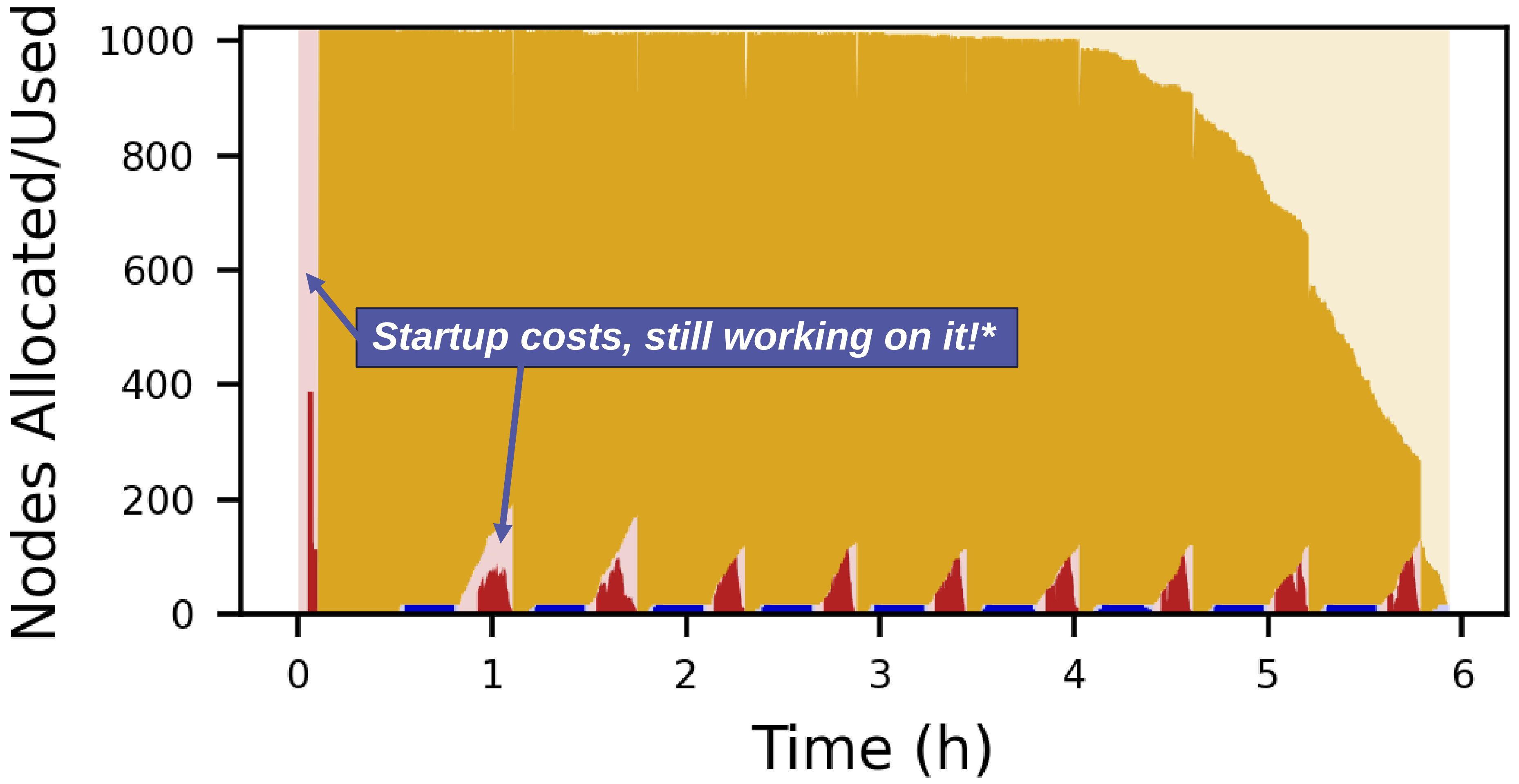


Found 10% more high-performing molecules with same allocation size

What made scaling hard?

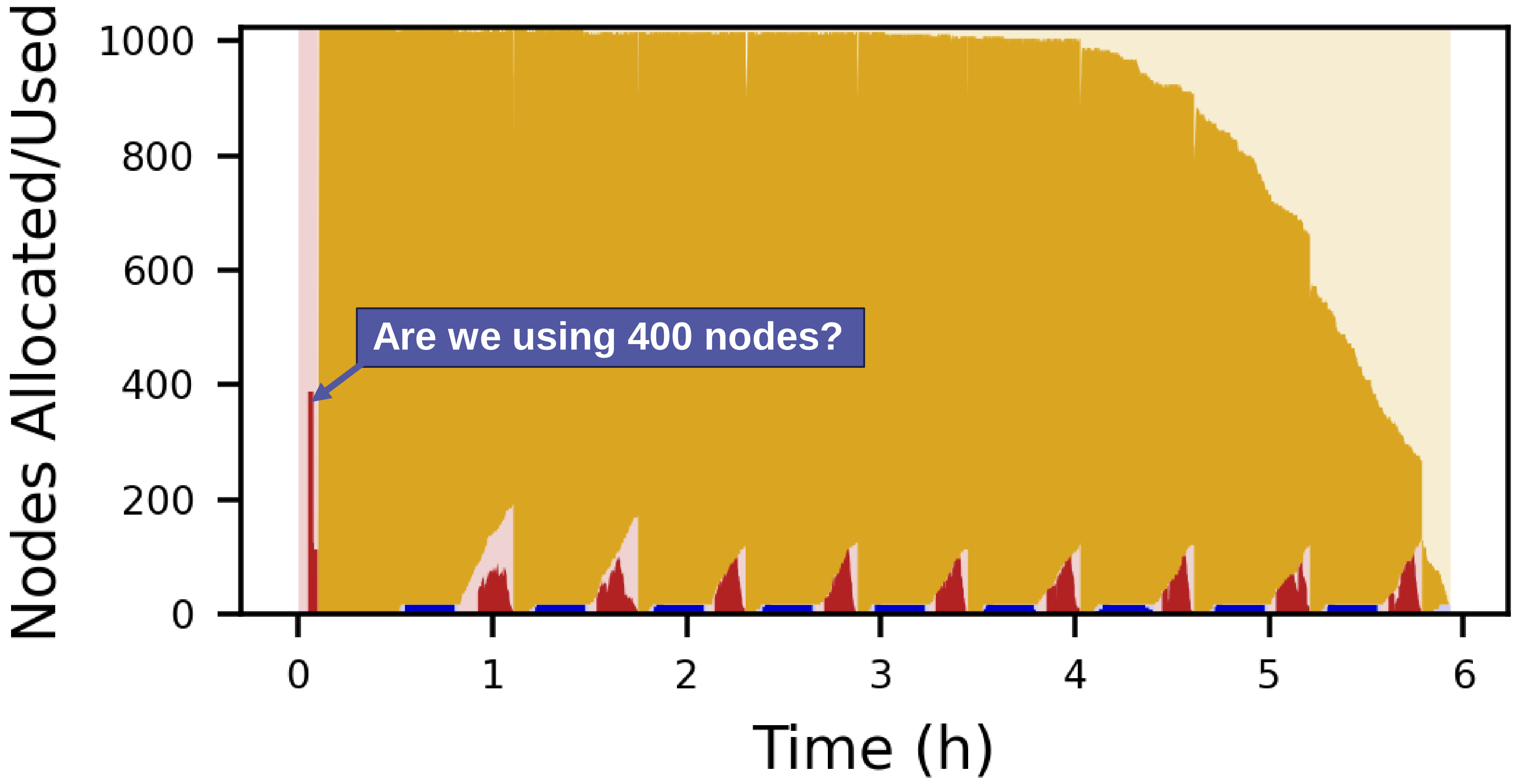
Ward et al. MLHPC @ SC'21, Ward et al. HCW @ IPDPS'23

Let’s talk performance problems

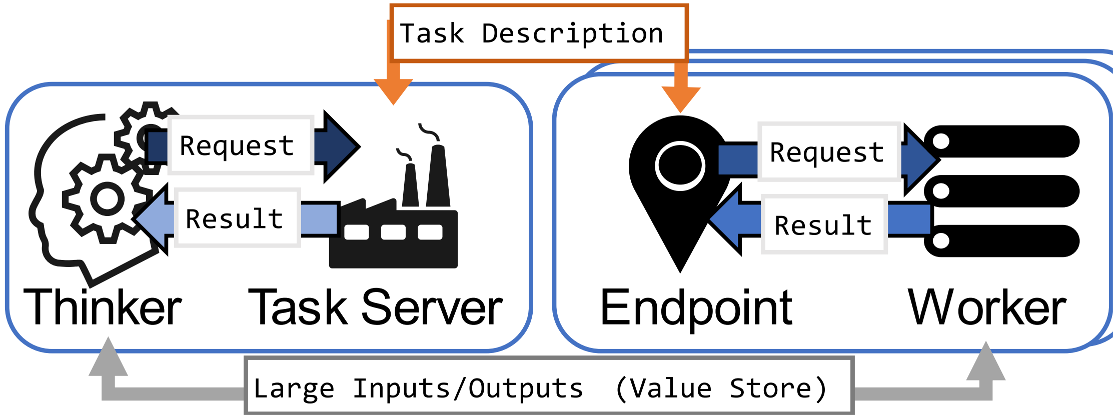


* See Kamatar et al., eScience (2023)

Let’s talk performance problems

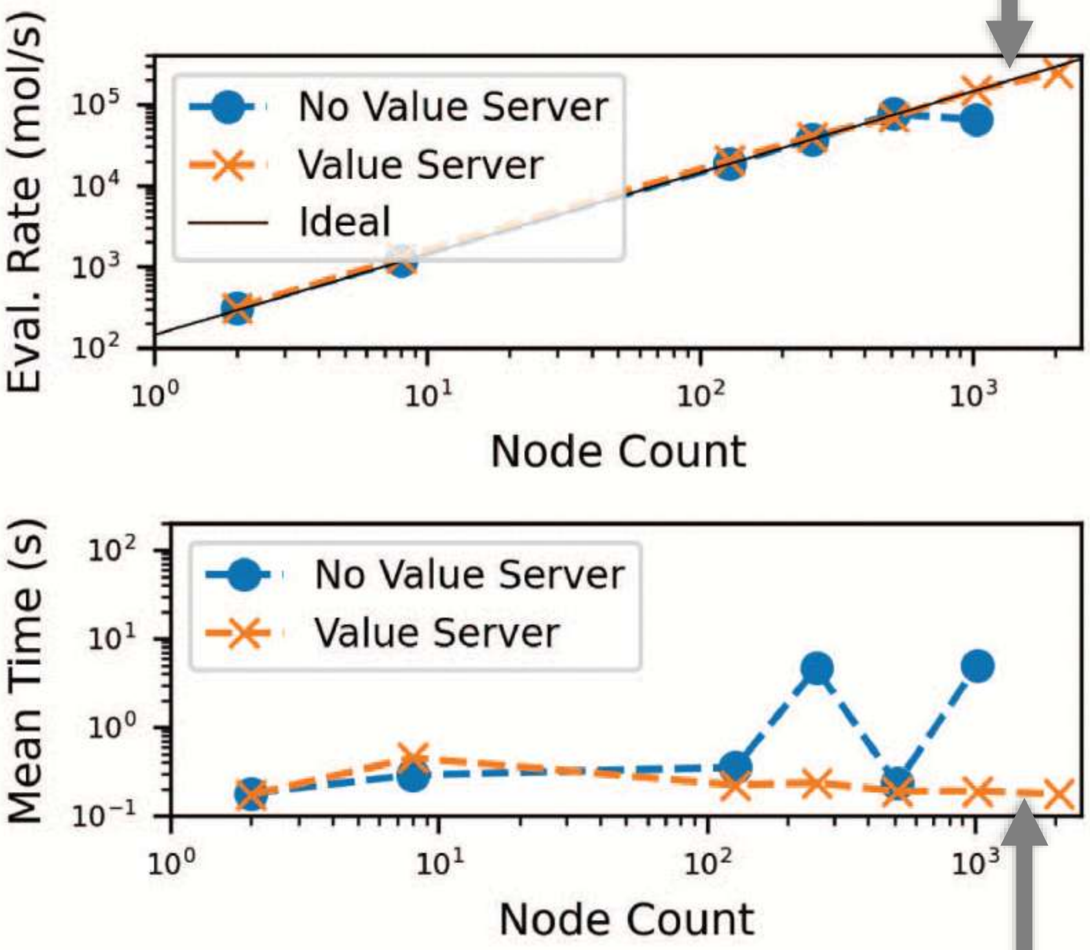


Adding a “value store” as a secondary channel



Data goes directly from Thinker to Worker

Scaling is improved…



by reducing task deploy time.

ProxyStore: Data side channel with minimal code changes

Core Concept: A make a value store backed by filesystems, Redis, Globus, …

```
store = RedisStore(name=‘redis-store’)   # Make a store
p = store.proxy(my_object)  # Put the data in a store
assert isinstance(p, type(my_object))  # p is a lazy reference to the object
```

Automatic Proxying

Just set a threshold in the queue

```
queues = PipeQueues(
    proxystore_name=‘redis-store’,
    proxystore_threshold=1000
)
```

Colmena will automatically make proxies, but they won't be reused

Manual Proxying

Make your own proxies, use them in a function

```
proxy = store.proxy(inputs)
self.queues.send_inputs(proxy, method=‘f’)
```

Proxies can be re-used across tasks, but you manage their deletion

That's it. No changing your application code!

EXASCALE COMPUTING PROJECT

ProxyStore is its own thing. Not part of Colmena

https://github.com/proxystore/proxystore

README.md

ProxyStore

docs passing | pre-commit.ci passed | tests passing

Python Lazy Object Proxy Interface for Distributed Stores

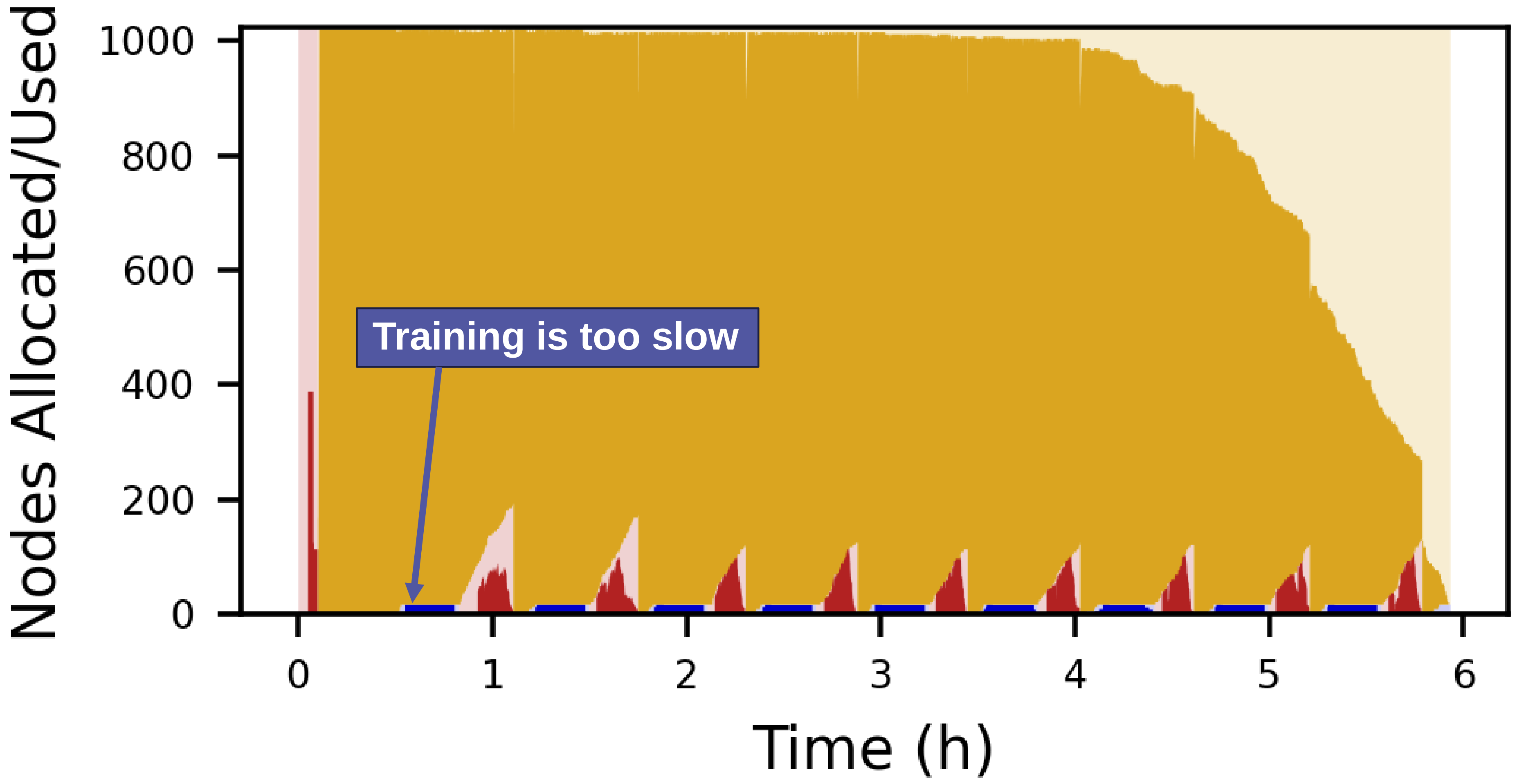
Installation

Install via pip:

```
# Base install
pip install proxystore
# Extras install for serving Endpoints
pip install proxystore[endpoints]
```

More details are available on the Get Started guide. For local development, see the Contributing guide.

Let's talk performance problems

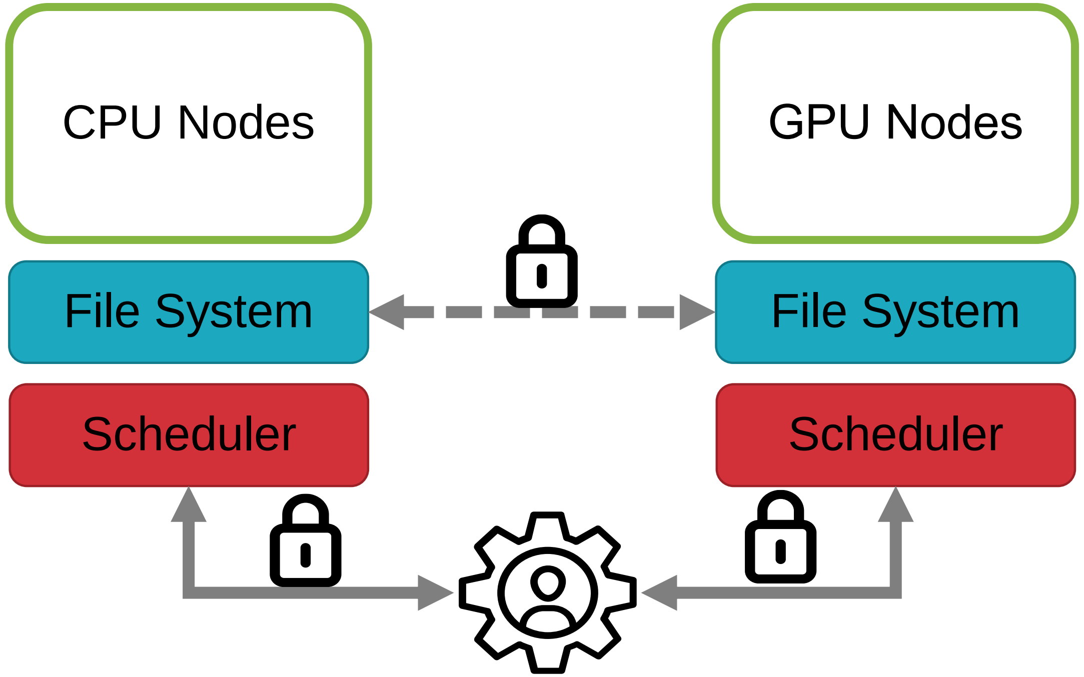


Science Workflows Require Diverse Compute, Especially with AI

There's some great hardware for training

but it's elsewhere

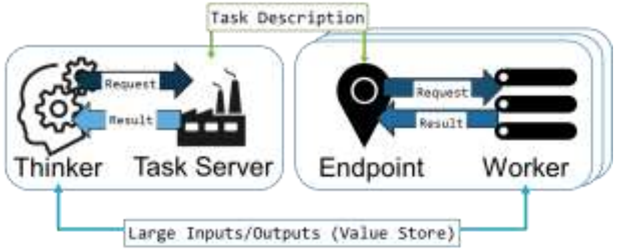
- Need open ports, or SSH tunnels
- Moving large data becomes a problem



Images: ALCF, NVIDIA

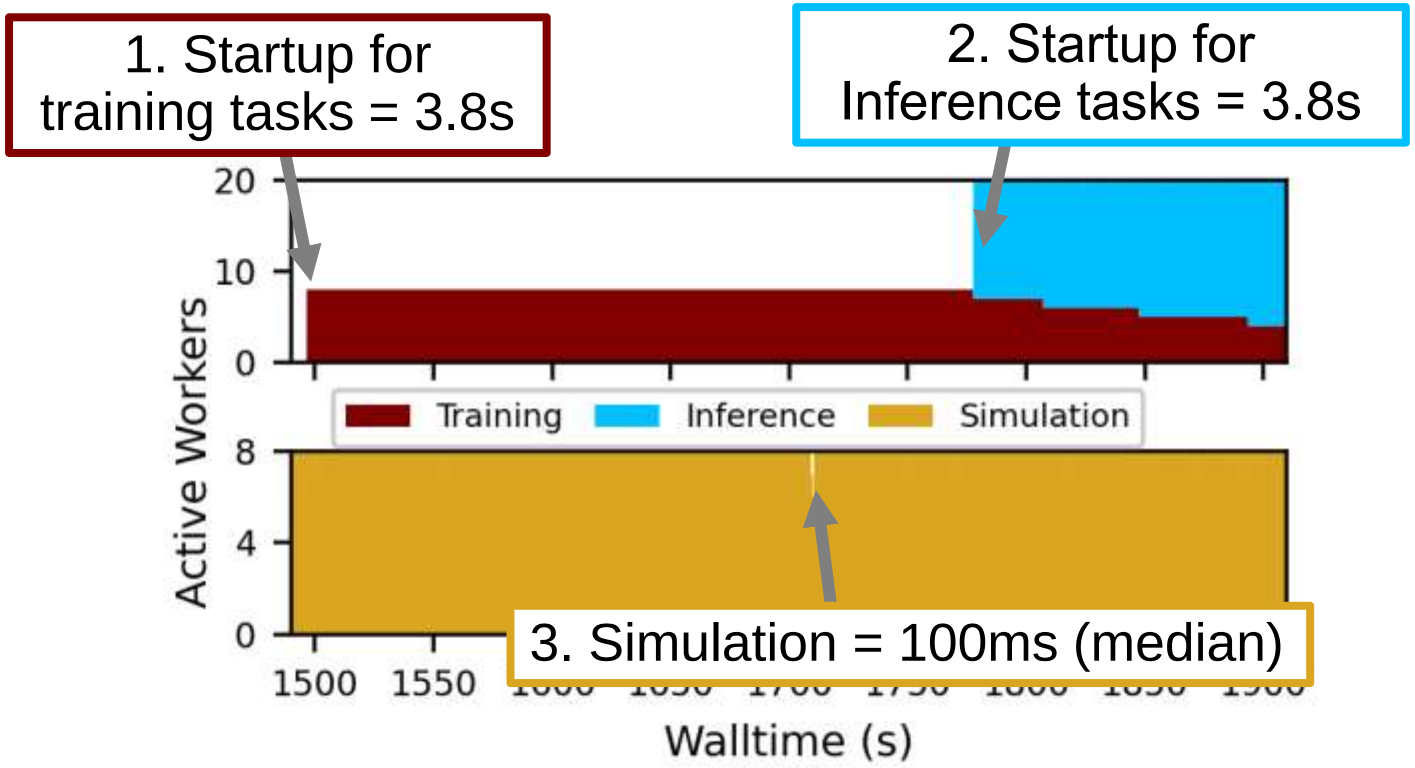
Globus Compute for Tasks, ProxyStore for Data

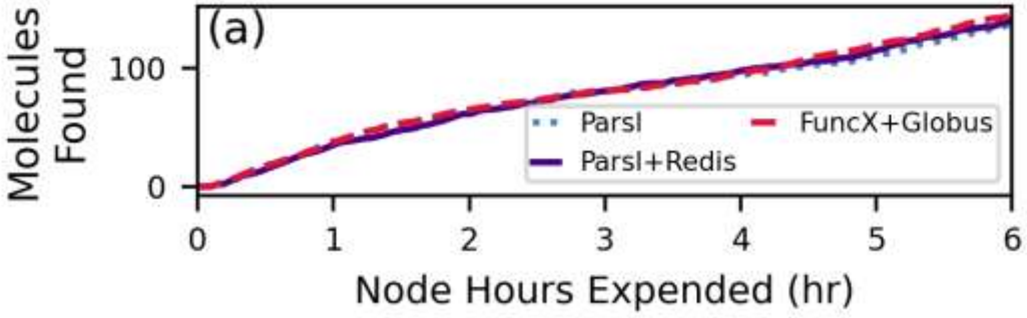
funcX → Globus Compute



ProxyStore ✖ globus

Few latencies are visible,
all small compared to task duration

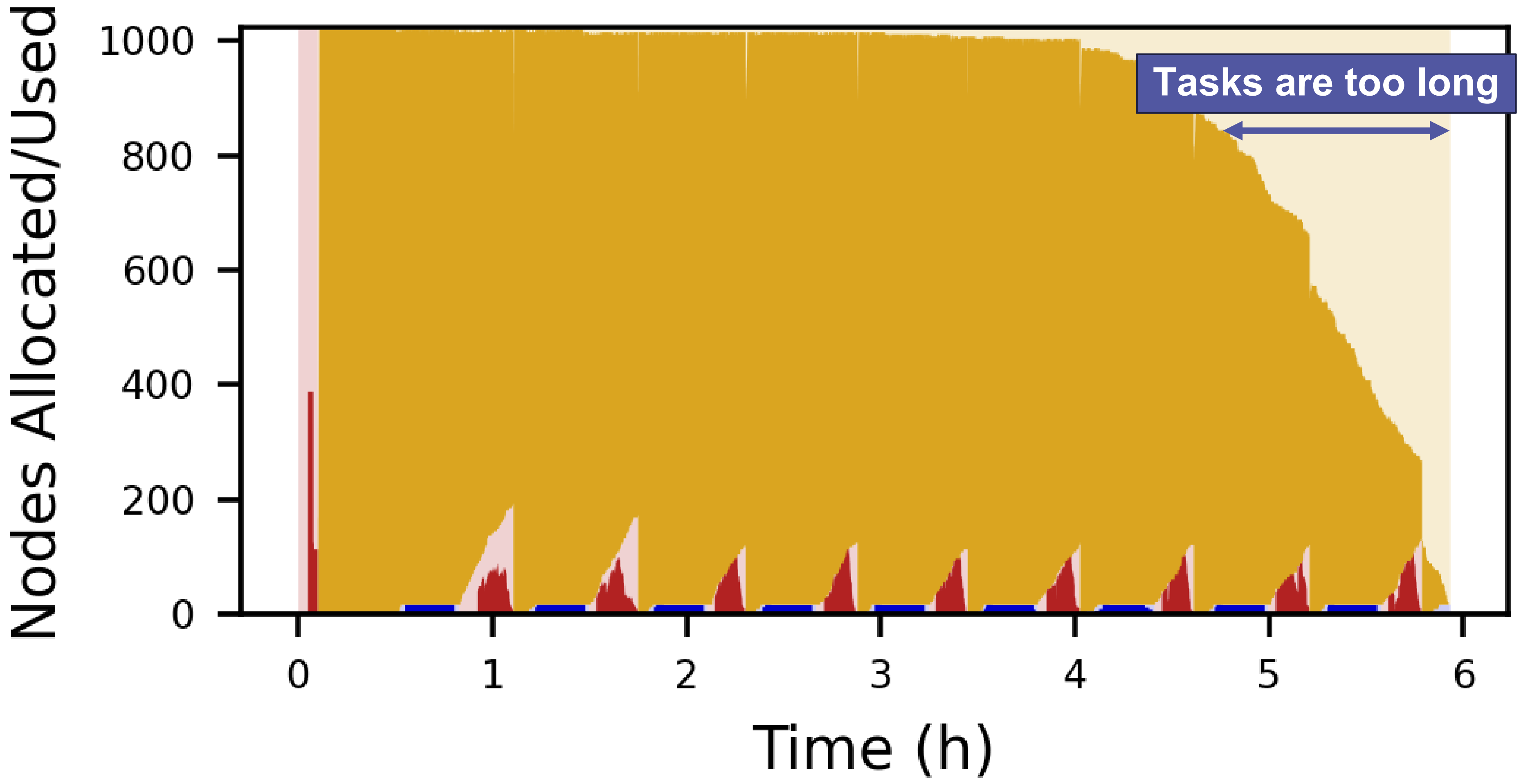




Science output is unaffected by convenience

More details: Ward et al. HCW (2023)

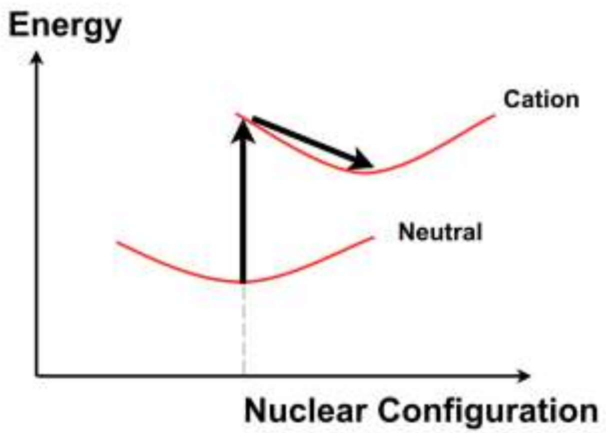
Let's talk performance problems



Solution 1: Other ECP projects making applications faster!

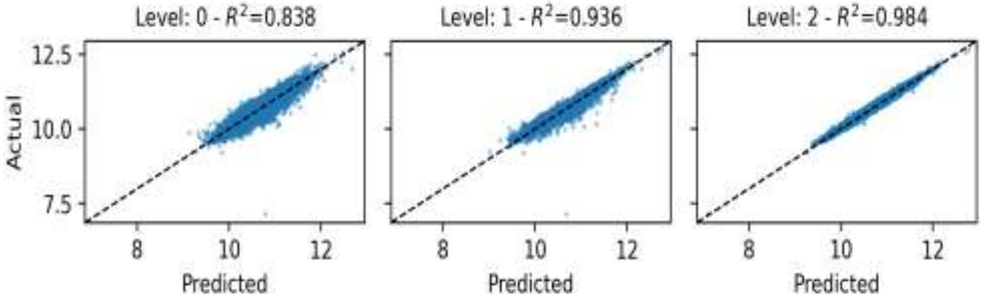
Breaking Pipelines into Pieces

Ionization energy is multiple steps

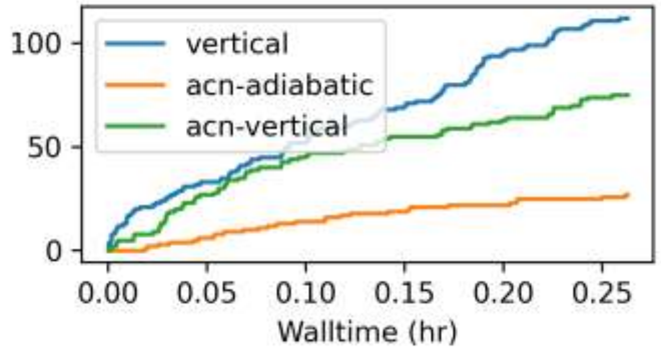


Source: Hutchison, [Chem StackExchage!]

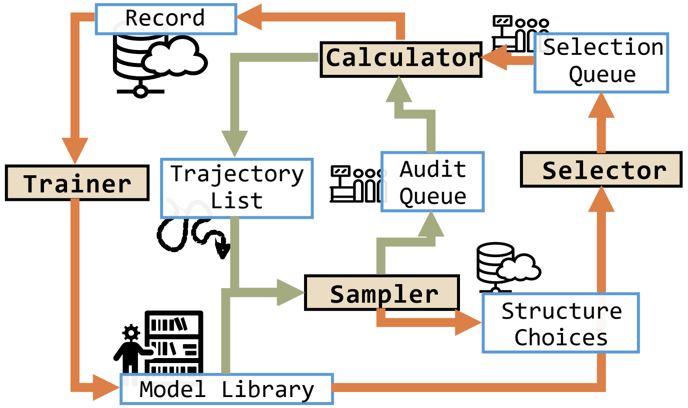
We can make better inferences after each step



Then run full-fidelity only on the best



An intricate policy that needs Colmena



What are our latest projects with Colmena?

Generating Materials for CO2 Storage

Generate Linkers: Diffusion Model

- Multi-GPU training for rapid updates
- Distribute generation across nodes

Assemble MOFs: CPU-bound

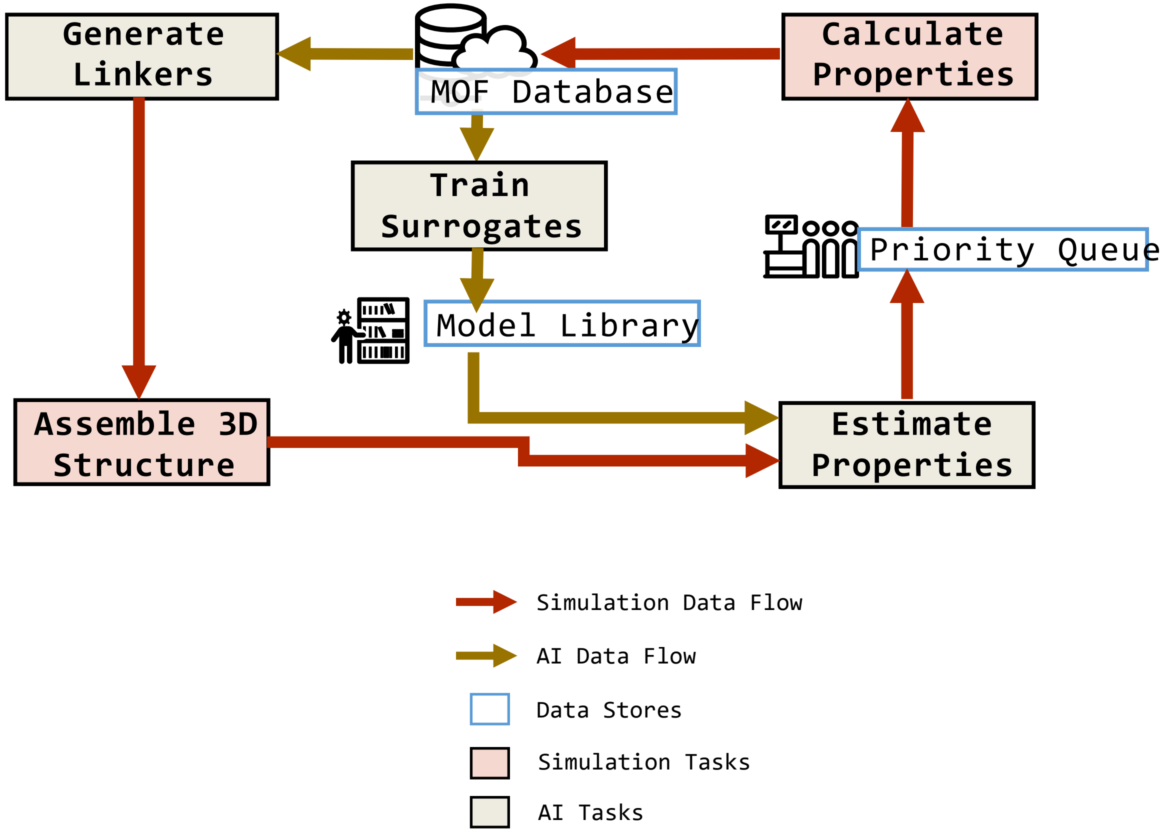
- Scattered across idle CPUs

Estimate Properties: ML, Cheap Physics

Compute Properties: Expensive Physics

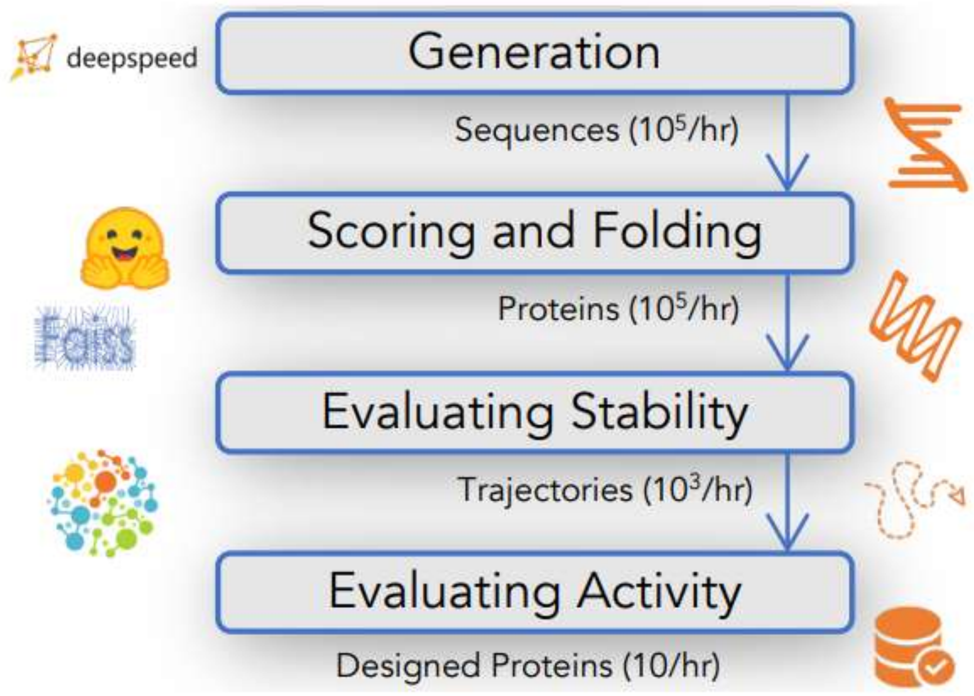
- Classical MD (LAMMPS), DFT (CP2K), ...

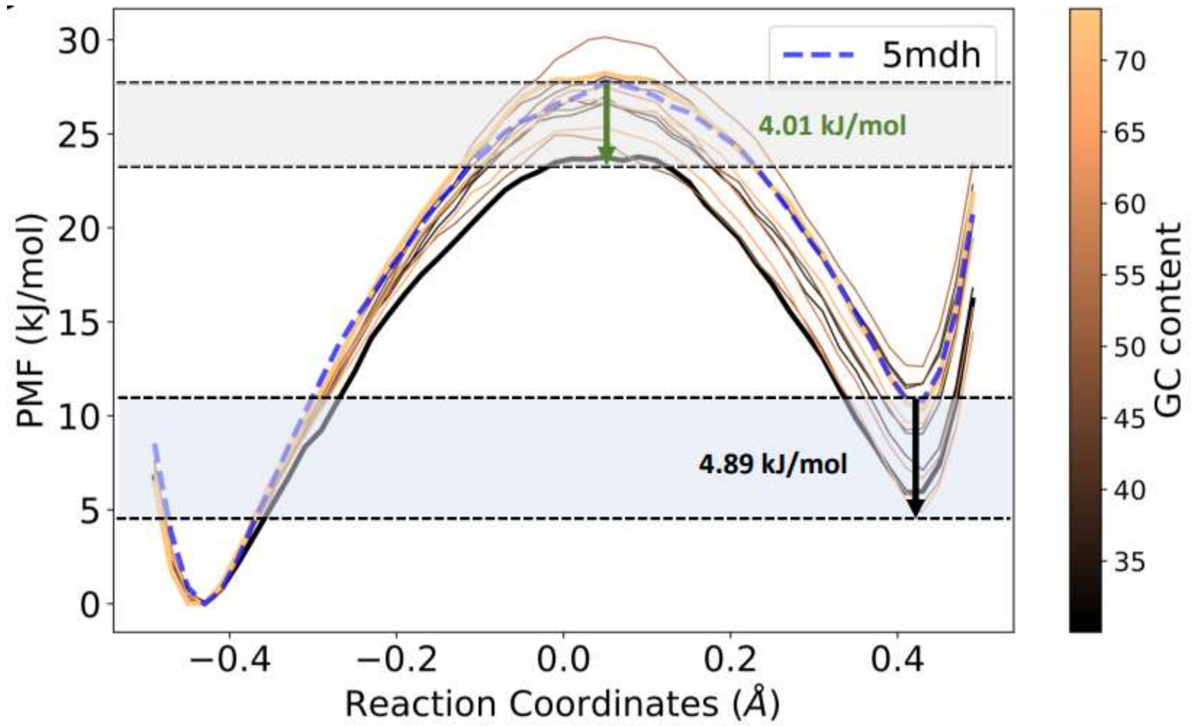
All at the same time



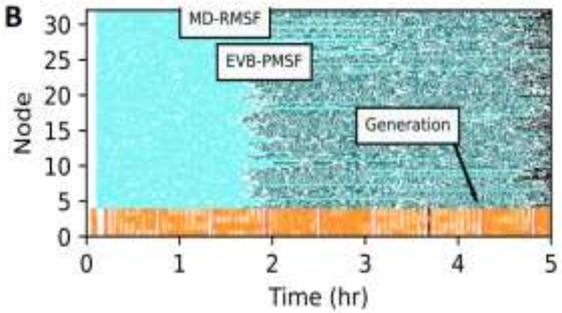
Work in progress: https://github.com/globus-labs/mof-generation-at-scale

Protein Design on with Genome-Scale Language Models





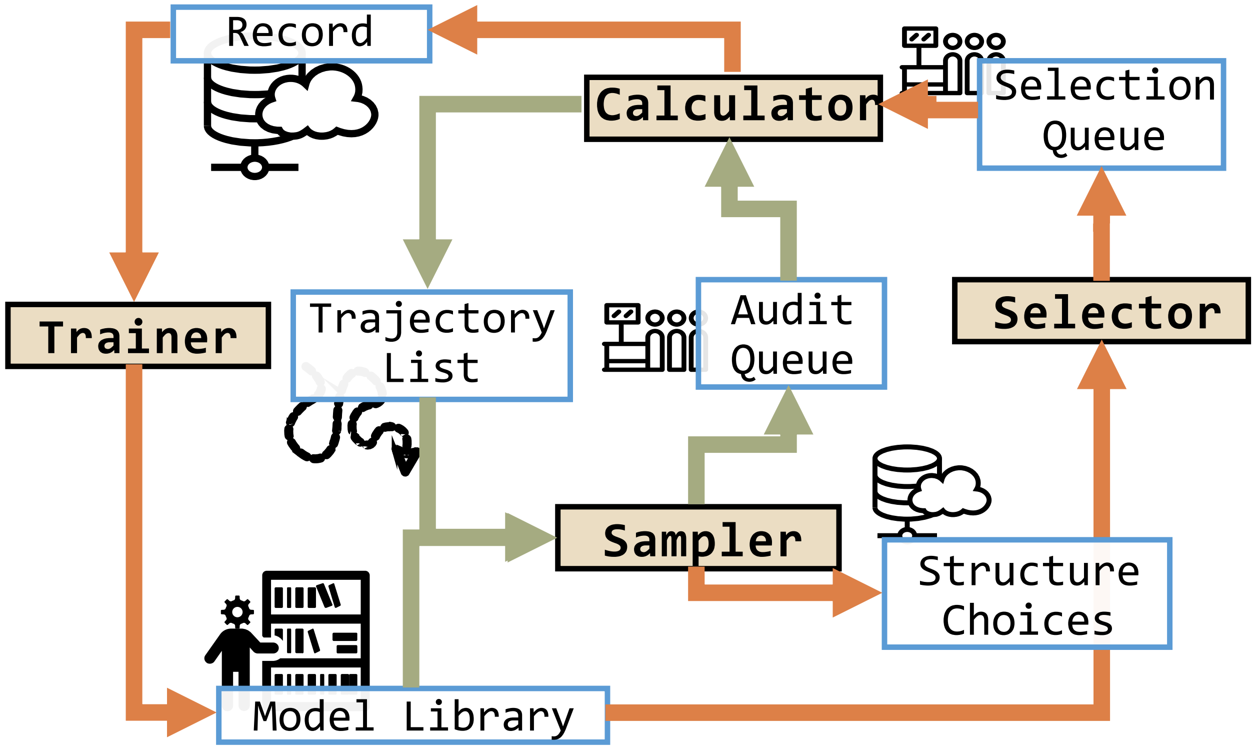
You guessed it,
all at the same time!



Ref: Dharuman et al. SC-W'23

Colmena is only one option

Why Colmena? Sophisticated policies



Why might you avoid Colmena?

- **Extremely large task throughput.** Best performance ~1000 decisions/second
  - Typical time from “result received” to “submitted” ~1ms
  - Possible™ with multiple task servers / thinkers, but not our main motivation

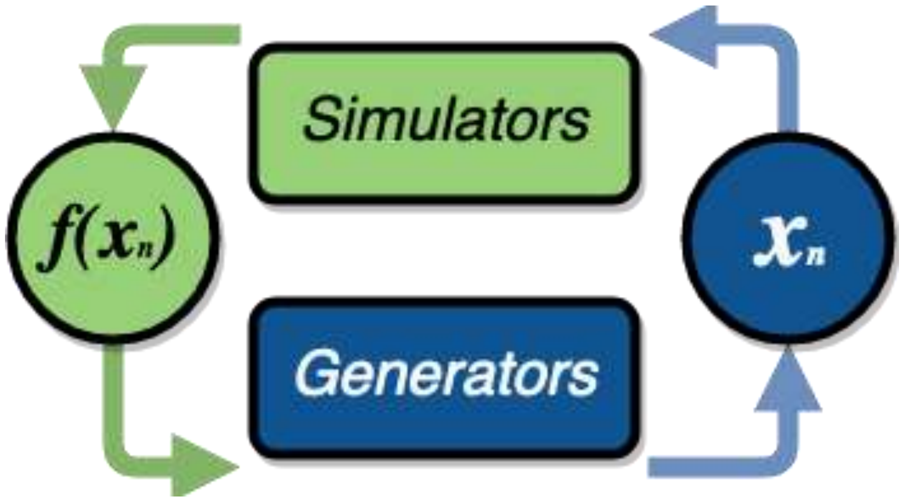
- **Human-in-the-loop workflows.** Consider things like Step Functions/Globus Automate instead

- **Intra-worker coordination.** Breaks our programming model. That’s what Decaf/Ray/etc is for

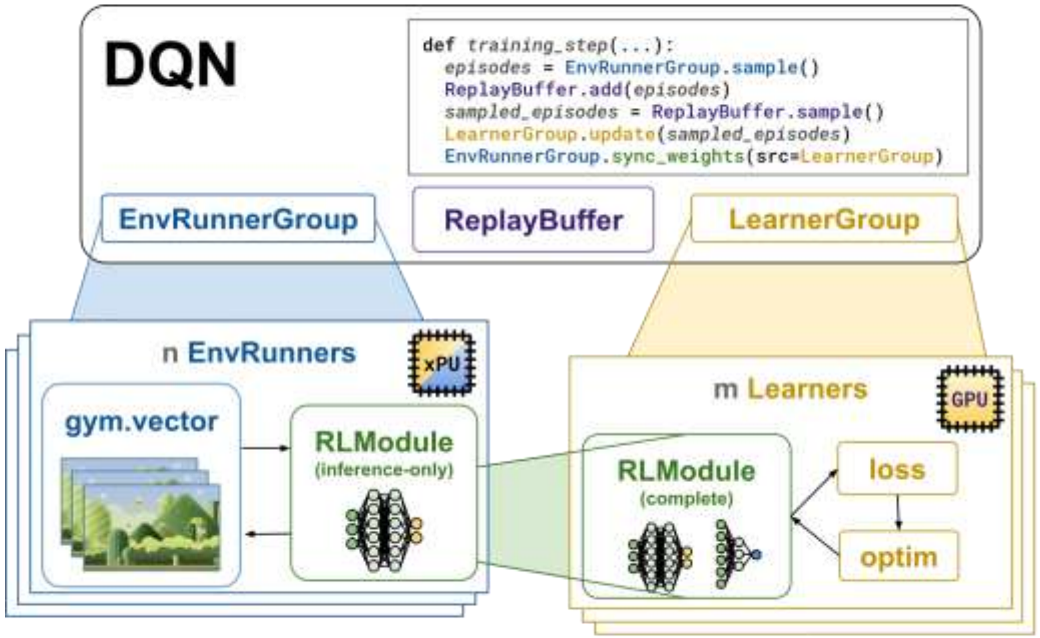
- **“Batteries included” for different domains (e.g., HPO).** We’re still on low-level problems

There are other programming models

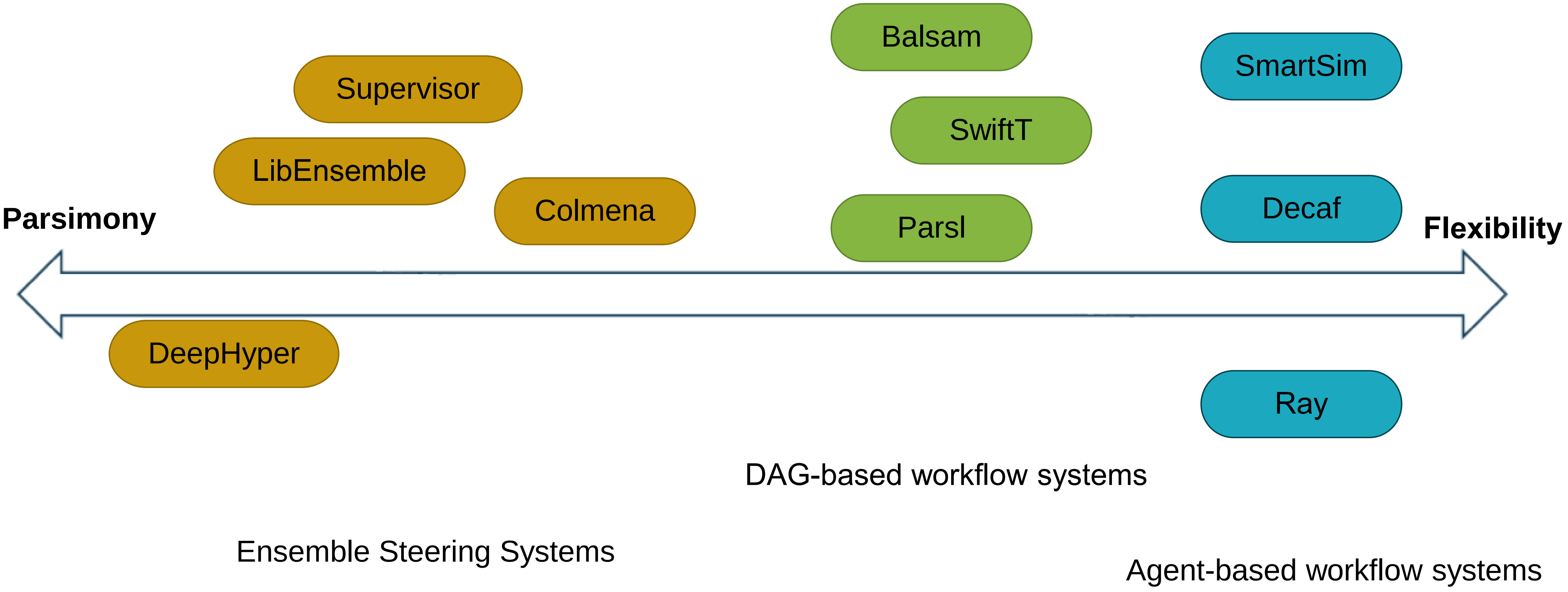
libEnsemble: Two functions



Ray: Decentralized Steering



What does it all fit in?



There are other axes: Performance vs configuration, “batteries”

We each have a list of problems…
and should not intertwine solutions with workflow packages

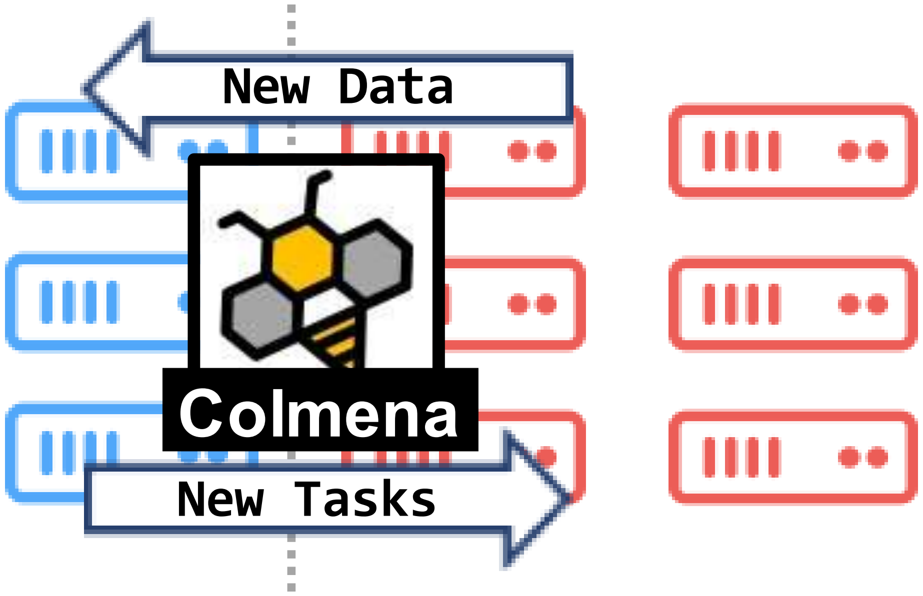
My Wishlist:	**Solution outside Colmena***
1. *Apps to respect GPU boundaries* ☑	Added GPU pinning to Parsl
2. *Communicating datasets is slow* ☑	ProxyStore
3. *Someone else to handle model versioning* ❓	MLFlow?
4. Python takes too long to start ❓	ALCF’s Copper?
5. A mechanism to monitor/halt tasks ❓	Redis? + …
6. To not care about which accelerator ❓	SYCL + Apptainer?
7. To never learn a new ML4Sci package ❓	Garden + 🤗?

*I work on Colmena

Summary: Colmena is for deploying AI+Simulation HPC

Key points:

- AI will play an increasing role in **controlling campaigns of simulations**
- Successful exascale computational campaigns will require **deploying AI on HPC**
- **Colmena** provides a Python library for building applications to interleave simulation and AI workflows
- We need a **broad collection of AI+HPC tools**, decoupled from individual workflow applicatons



See also: https://colmena.rtfd.io/ , https://github.com/exalearn/colmena