Academy: Empowering Scientific Workflows with Federated Agents

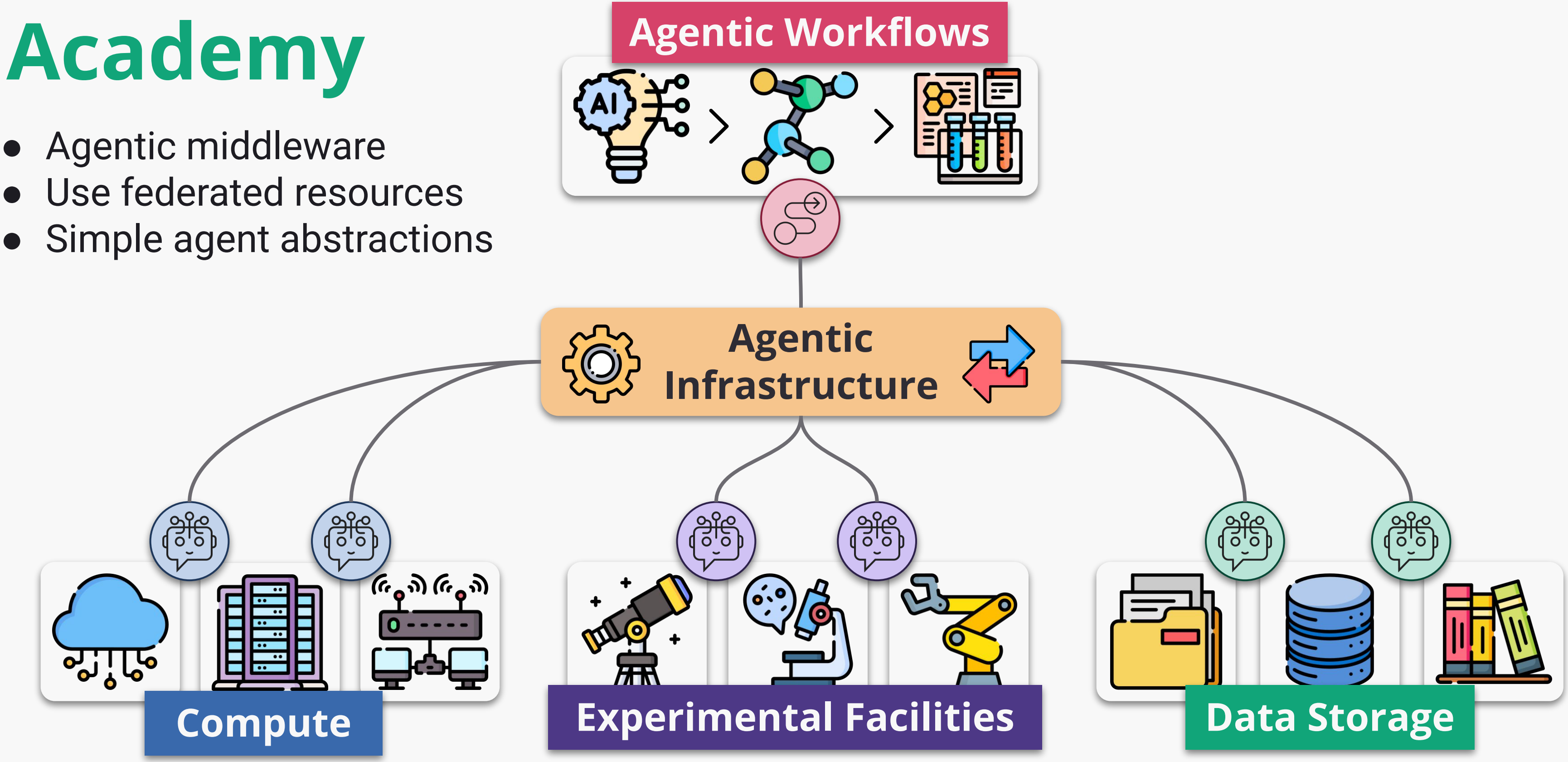
Greg Pauloski, Yadu Babuji, Ryan Chard, Alok Kamatar, Mansi Sakarvadia,
Kyle Chard, Ian Foster

JLESC: AI for Science Breakout

14 May 2025

Academy

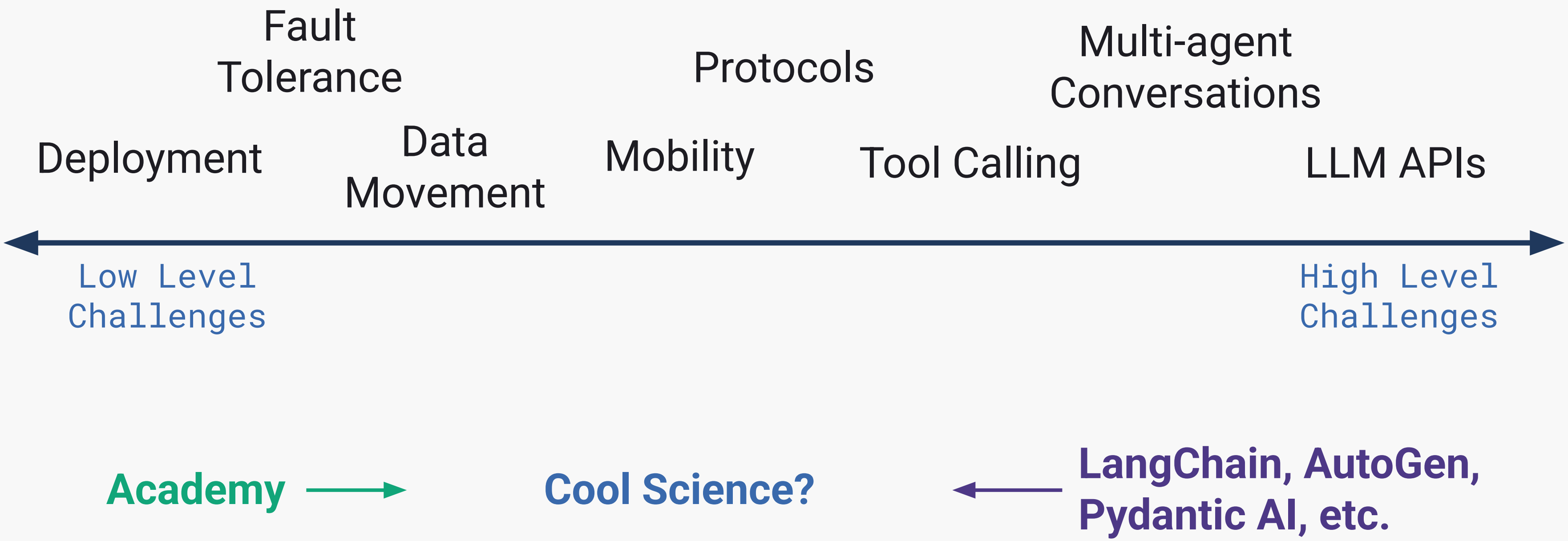
- Agentic middleware
- Use federated resources
- Simple agent abstractions



Agentic Middleware

Software layer that transparently manages the lifecycle, communication, and coordination of autonomous agents across distributed computing environments.

Agentic Middleware: Scope & Challenges



Fault Tolerance

Protocols

Multi-agent Conversations

Deployment

Data Movement

Mobility

Tool Calling

LLM APIs

Low Level Challenges

High Level Challenges

Academy →

Cool Science?

← **LangChain, AutoGen, Pydantic AI, etc.**

Agentic Middleware: Using Research Infrastructure

Centralized

- Agents co-located (workstation, cloud)
- Research infrastructure available via APIs (REST, SDKs, …)
- Use infrastructure via tool calling

++ Rapidly growing library ecosystem

-- Limited APIs for infrastructure

LangChain, AutoGen, Pydantic AI, etc.

Decentralized

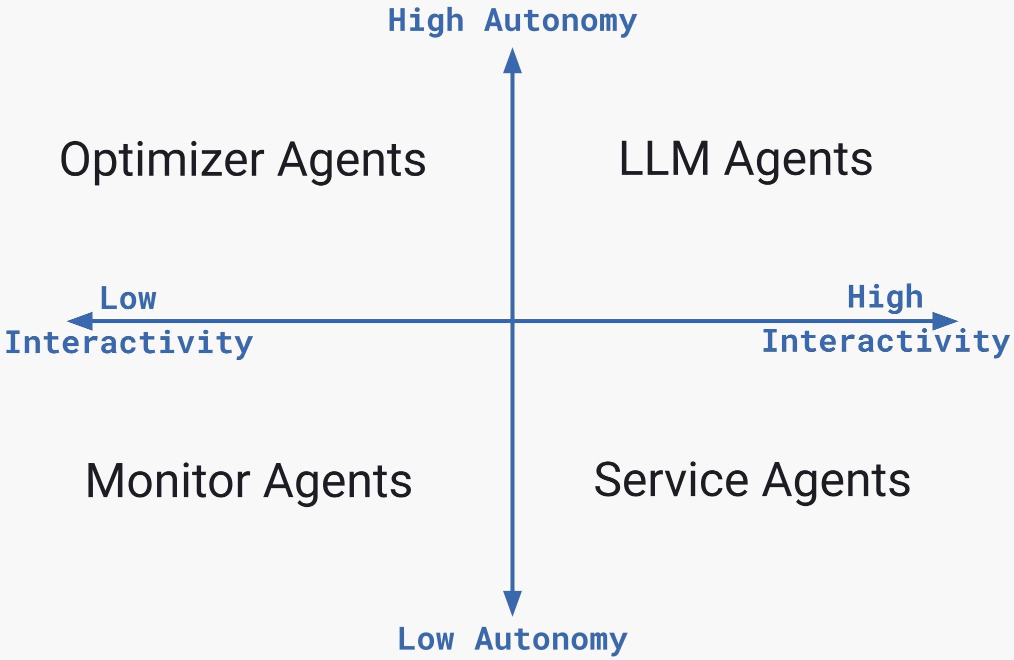
- Agents distributed across infrastructure
- Agents interact asynchronously
- Use infrastructure directly (actuate a robot, submit job, …)

++ Data locality, perf., loose coupling

-- Deployment complexity

Academy

Agentic Middleware: Agent Behaviors



High Autonomy

Optimizer Agents

LLM Agents

Low Interactivity

High Interactivity

Monitor Agents

Service Agents

Low Autonomy

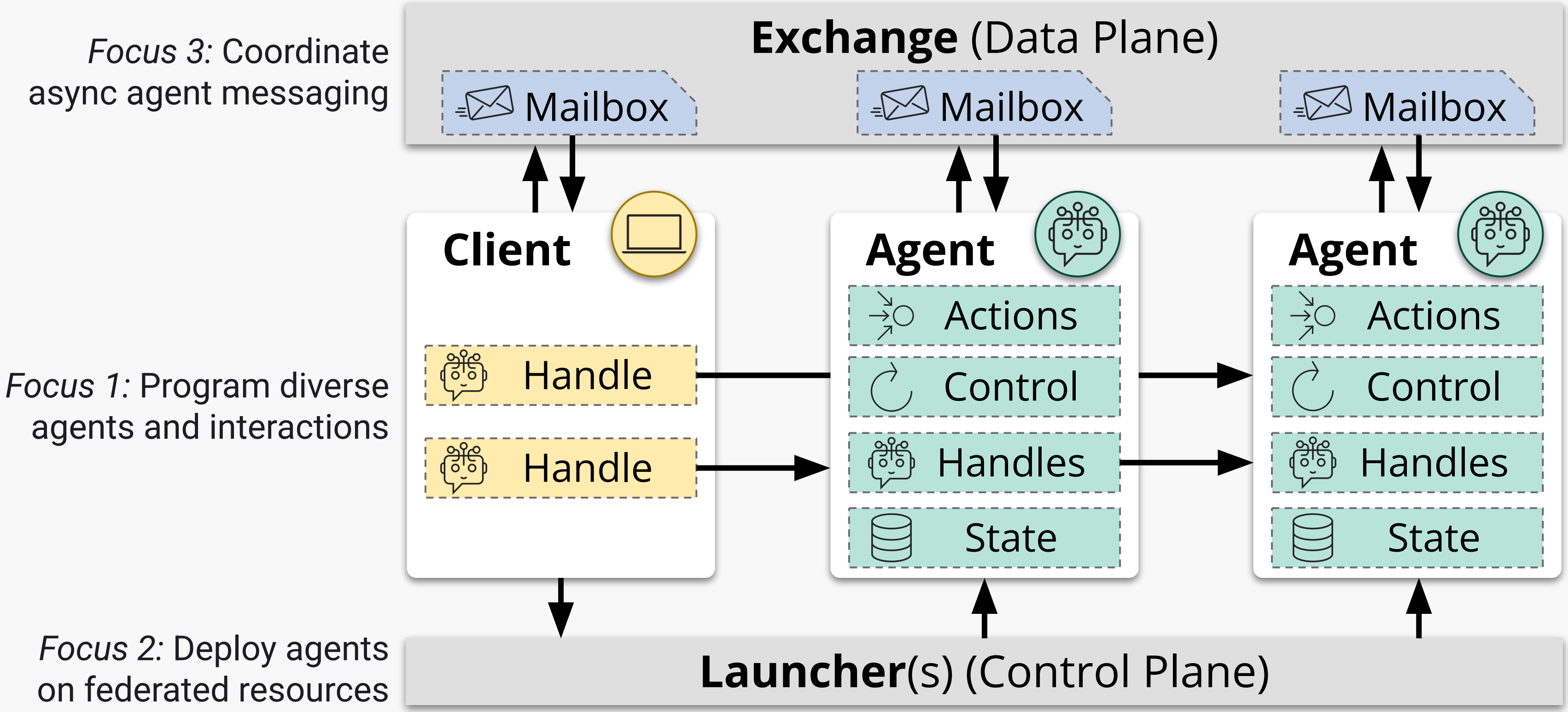
Other defining aspects:

- Persistent vs ephemeral
- General vs narrow purpose
- Embodiment

Long-running agentic science apps will incorporate many kinds of agent behaviors.

Academy primitives support the creation diverse agent types.

How does **Academy** support the expression of **diverse agent behaviors** and deployment across **distributed/federated resources**?


Focus 3: Coordinate async agent messaging
Exchange (Data Plane)
Mailbox
Mailbox
Mailbox
Client
Agent
Agent
Actions
Actions
Focus 1: Program diverse agents and interactions
Handle
Control
Control
Handle
Handles
Handles
State
State
Focus 2: Deploy agents on federated resources
Launcher(s) (Control Plane)

https://academy.proxystore.dev/latest/concepts/

Communication & Execution

Exchange

- Asynchronous communication through mailboxes
- Every agent/client in system has a unique mailbox
- Local & distributed implementations
- Optimized for low-latency
- Hybrid communication model
- Prefer direct communication between agents when possible; fall back to indirect communication via object store
- Pass-by-reference with ProxyStore for large data

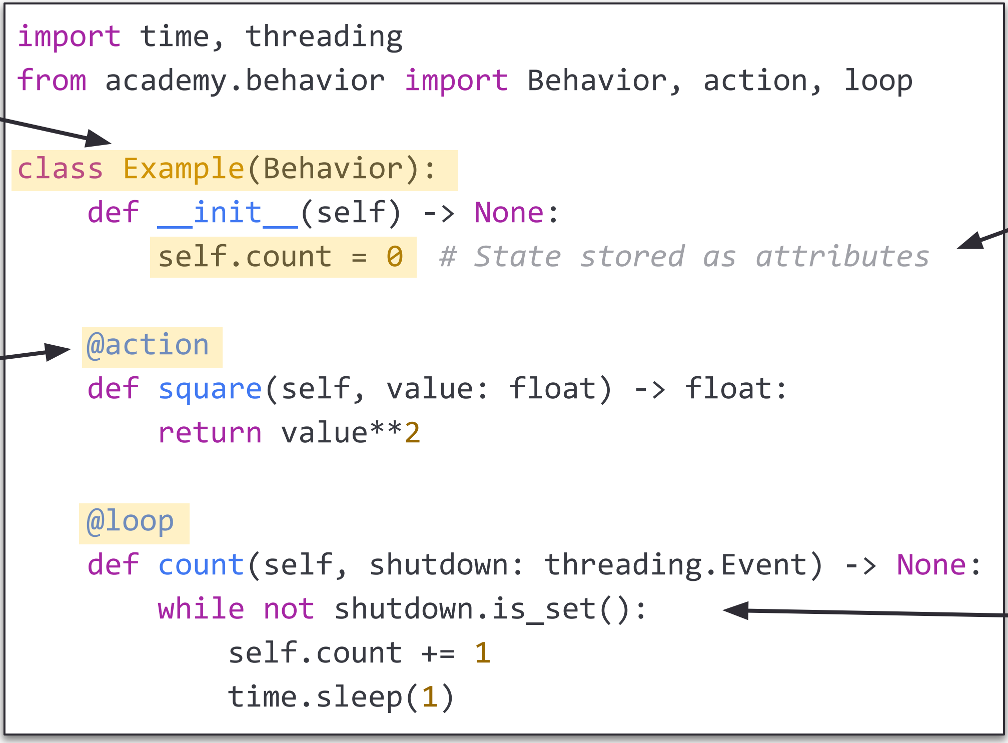
Launcher

- Not required but enables remote execution of agents
- Returns handle to launched agent
- Local threads or processes
- Distributed with Parsl
- Federated with Globus Compute

Writing Apps in Academy

Agents defined by a behavior

Clients & other agents can request actions

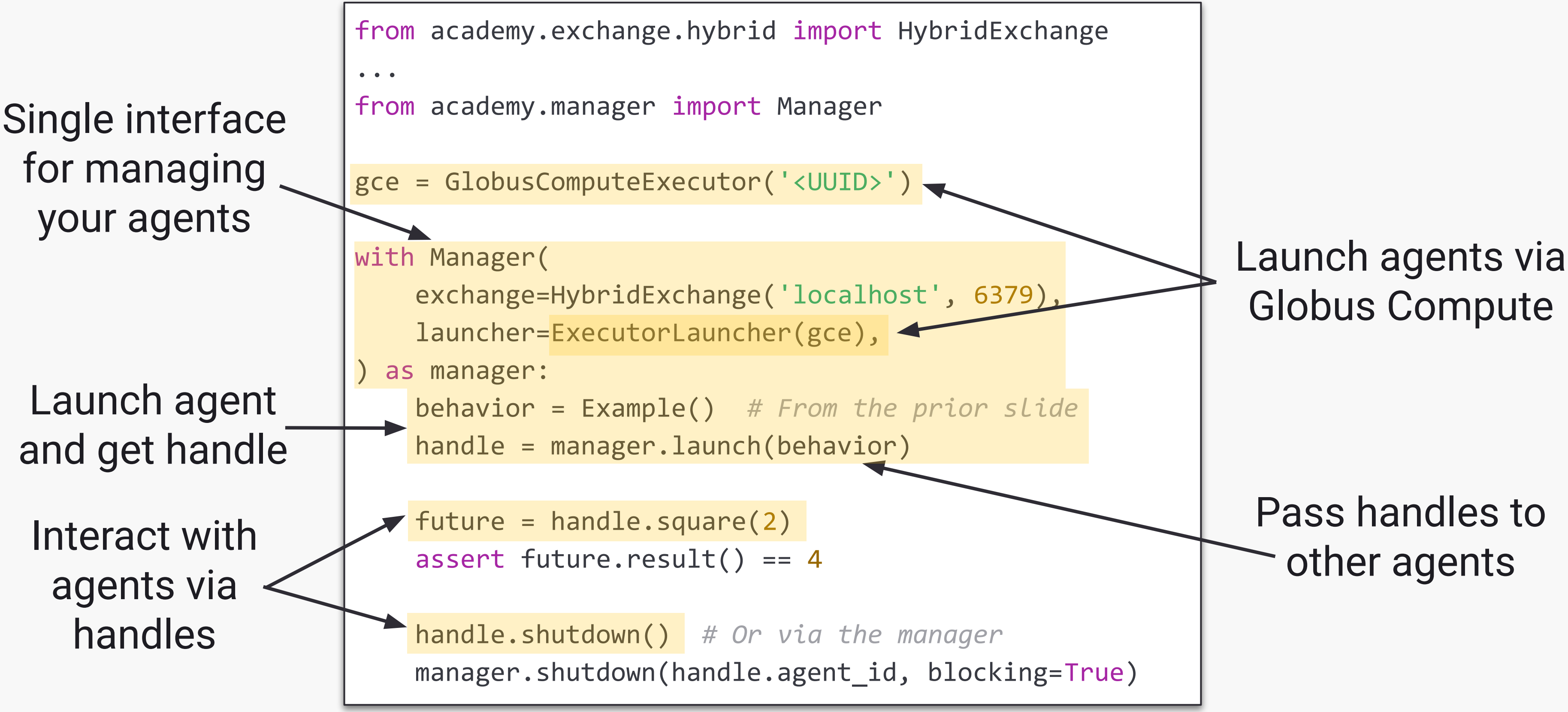


```python
import time, threading
from academy.behavior import Behavior, action, loop

class Example(Behavior):
    def __init__(self) -> None:
        self.count = 0  # State stored as attributes

    @action
    def square(self, value: float) -> float:
        return value**2

    @loop
    def count(self, shutdown: threading.Event) -> None:
        while not shutdown.is_set():
            self.count += 1
            time.sleep(1)
```

Instance of a behavior is state

Control loops for autonomous behavior

https://academy.proxystore.dev/latest/get-started/



```
from academy.exchange.hybrid import HybridExchange
...
from academy.manager import Manager

gce = GlobusComputeExecutor('<UUID>')

with Manager(
    exchange=HybridExchange('localhost', 6379),
    launcher=ExecutorLauncher(gce),
) as manager:
    behavior = Example()  # From the prior slide
    handle = manager.launch(behavior)

    future = handle.square(2)
    assert future.result() == 4

    handle.shutdown()  # Or via the manager
    manager.shutdown(handle.agent_id, blocking=True)
```

Single interface for managing your agents

Launch agent and get handle

Interact with agents via handles

Launch agents via Globus Compute

Pass handles to other agents

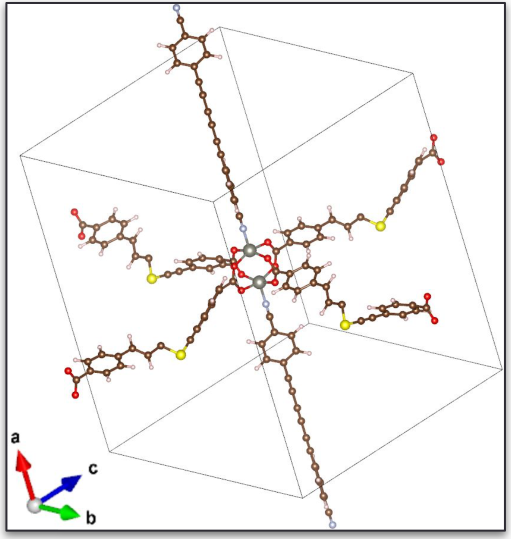
https://academy.proxystore.dev/latest/get-started/

Use Case: MOF Discovery

Metal Organic Frameworks (MOF)

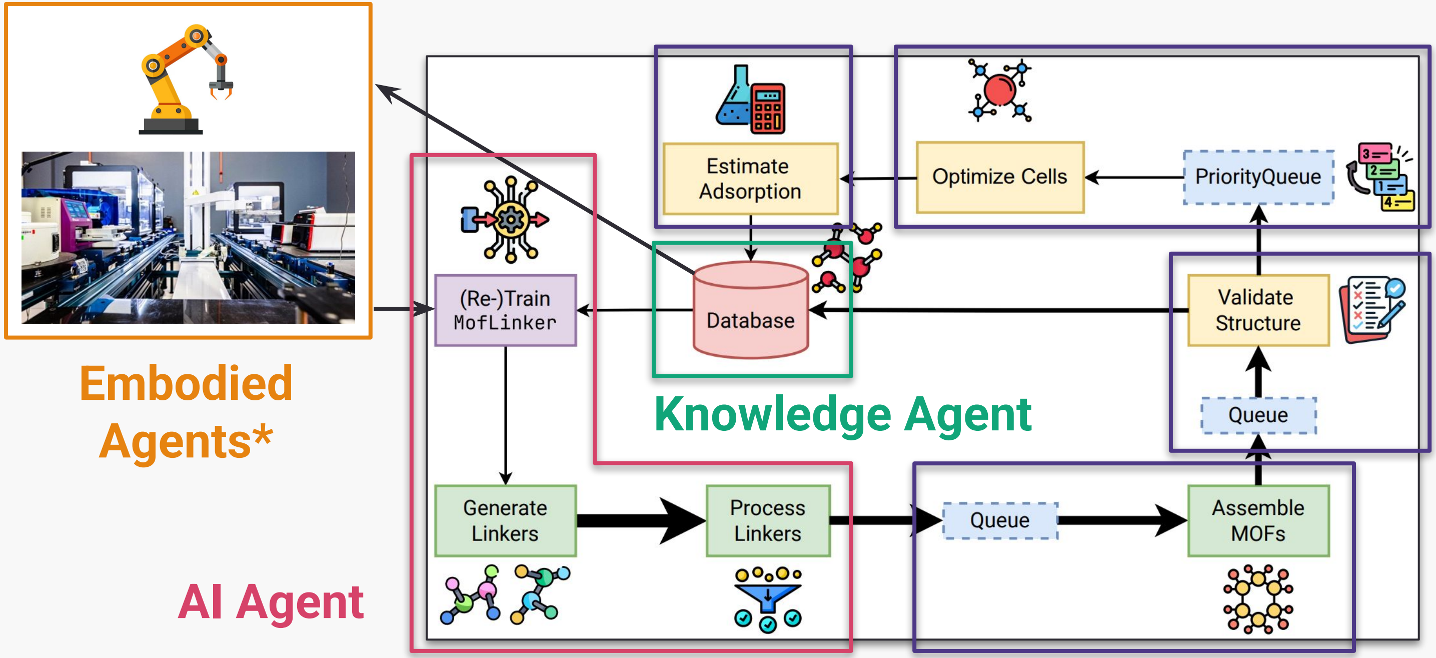
- ➔ Composed of organic molecules (ligands) and inorganic metals (nodes)
- ➔ The sponges of materials science!
- ➔ Porous structures that adsorb and store gases
- ➔ Topologies can be optimized for targeted gas storage → **Carbon Capture**

How to efficiently discover MOFs with desirable properties for target applications?



Intractable search space of ligand, node, & geometry combinations

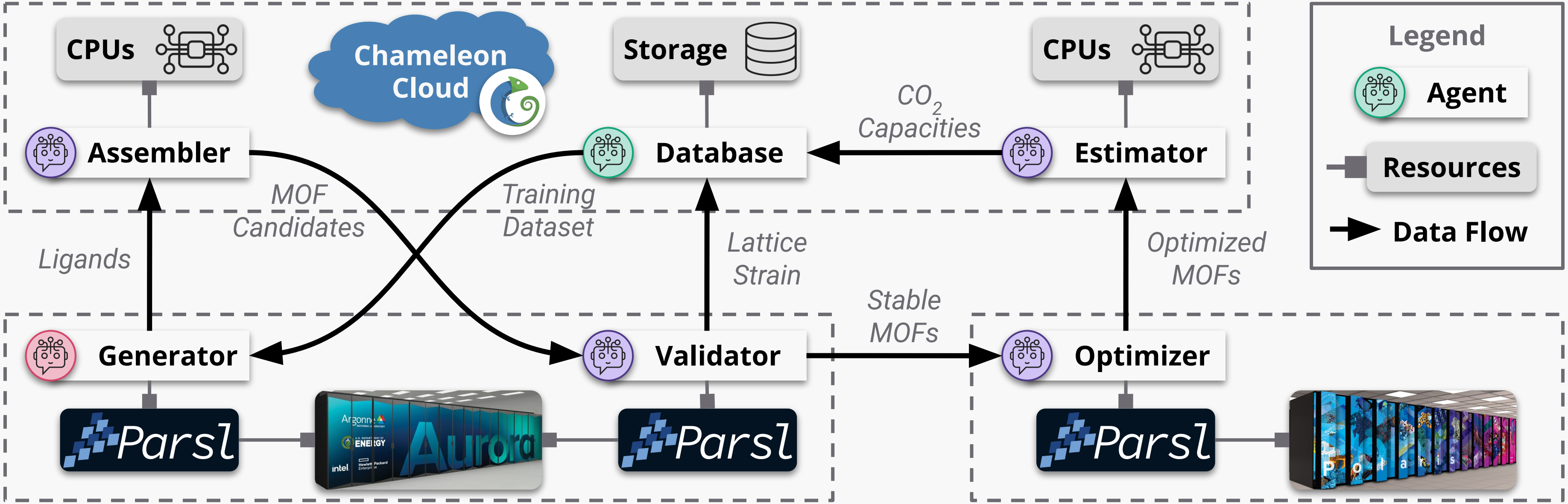
MOFA: Online learning + GenAI + Simulation



Computational Agents

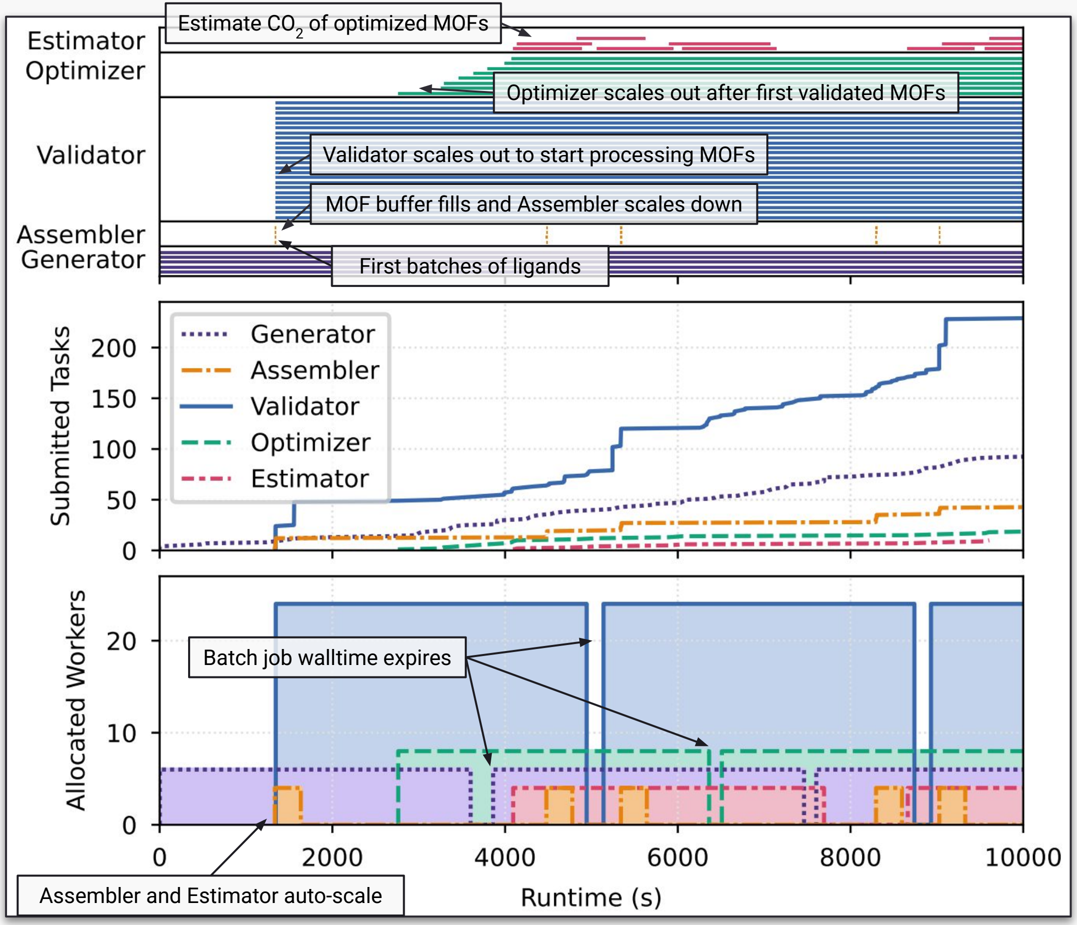
Yan et al., "**MOFA: Discovering Materials for Carbon Capture with a GenAI- and Simulation-Based Workflow**" (Under Review)

MOFA through Autonomous Agents



Agents executed remotely via Globus Compute

MOFA Agents Trace



Why is this agentic model better?

- ➔ **Placement:** Move agents to resources
- ➔ **Separation of concerns:** Resource acquisition and scaling based on local workload
- ➔ **Loose coupling:** Swap agents or integrate new agents (e.g., SDL)
- ➔ **Shared agents:** Multiple workflows can share agents (microservice-like)

Questions?

J. Gregory
Pauloski

Yadu
Babuji

Ryan
Chard

Alok
Kamatar

Mansi
Sakarvadia

Kyle
Chard

Ian
Foster

Reach out if you are interested:

`jgpauloski@uchicago.edu`

Learn more/stay up to date:

- arxiv.org/abs/2505.05428
- github.com/proxystore/academy
- academy.proxystore.dev

⭐ Academy on GitHub!