Academy: Empowering Scientific Workflows with Federated Agents

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

ParslFest 2025

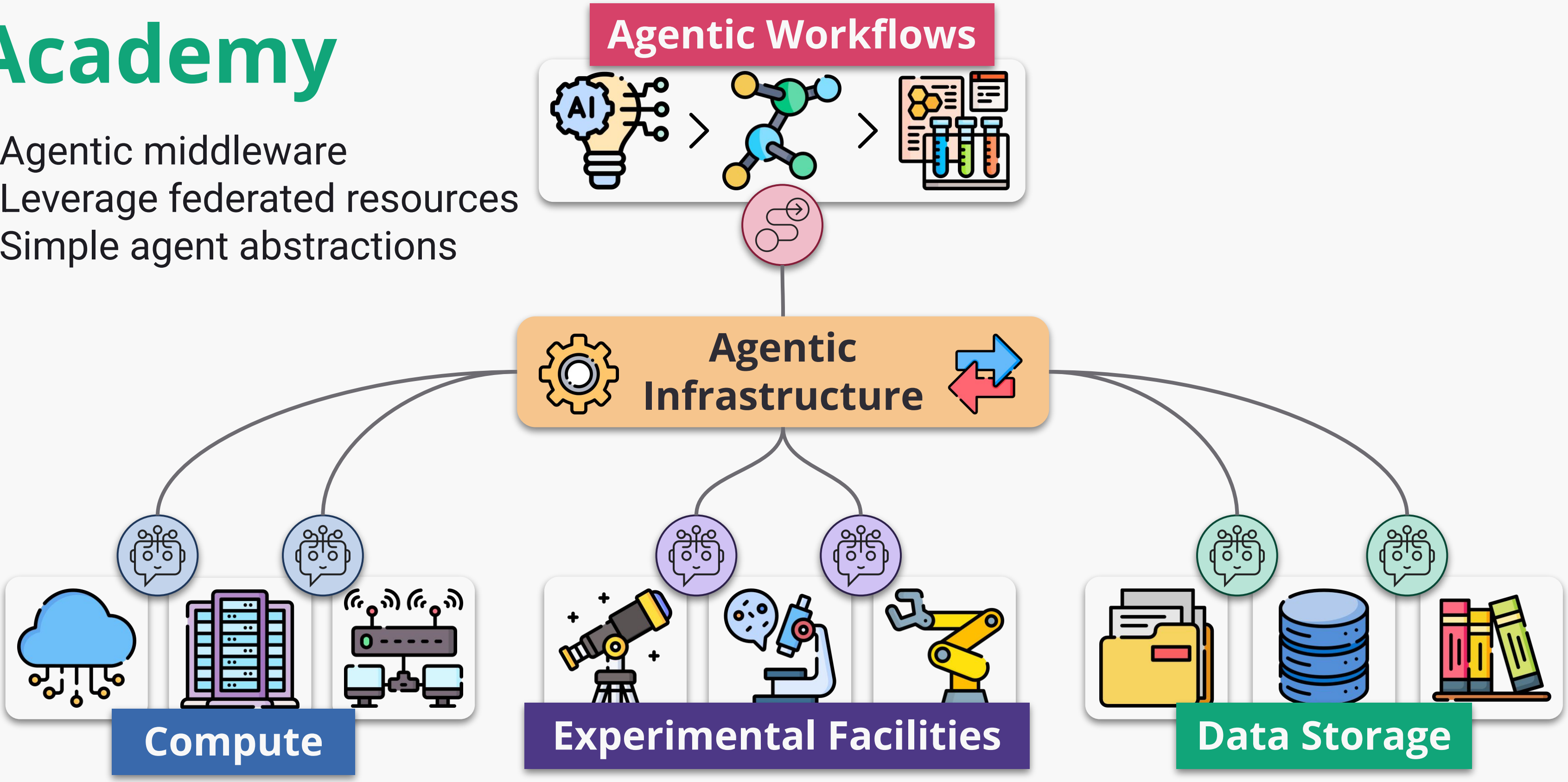
THE UNIVERSITY OF CHICAGO

29 August 2025

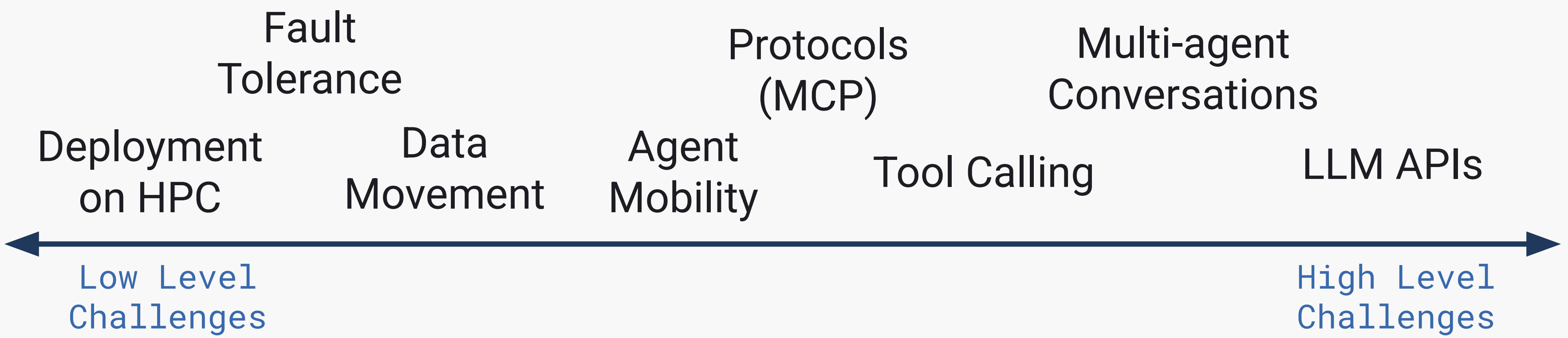
globus labs

Academy

- Agentic middleware
- Leverage federated resources
- Simple agent abstractions



Agentic Middleware: Scope & 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

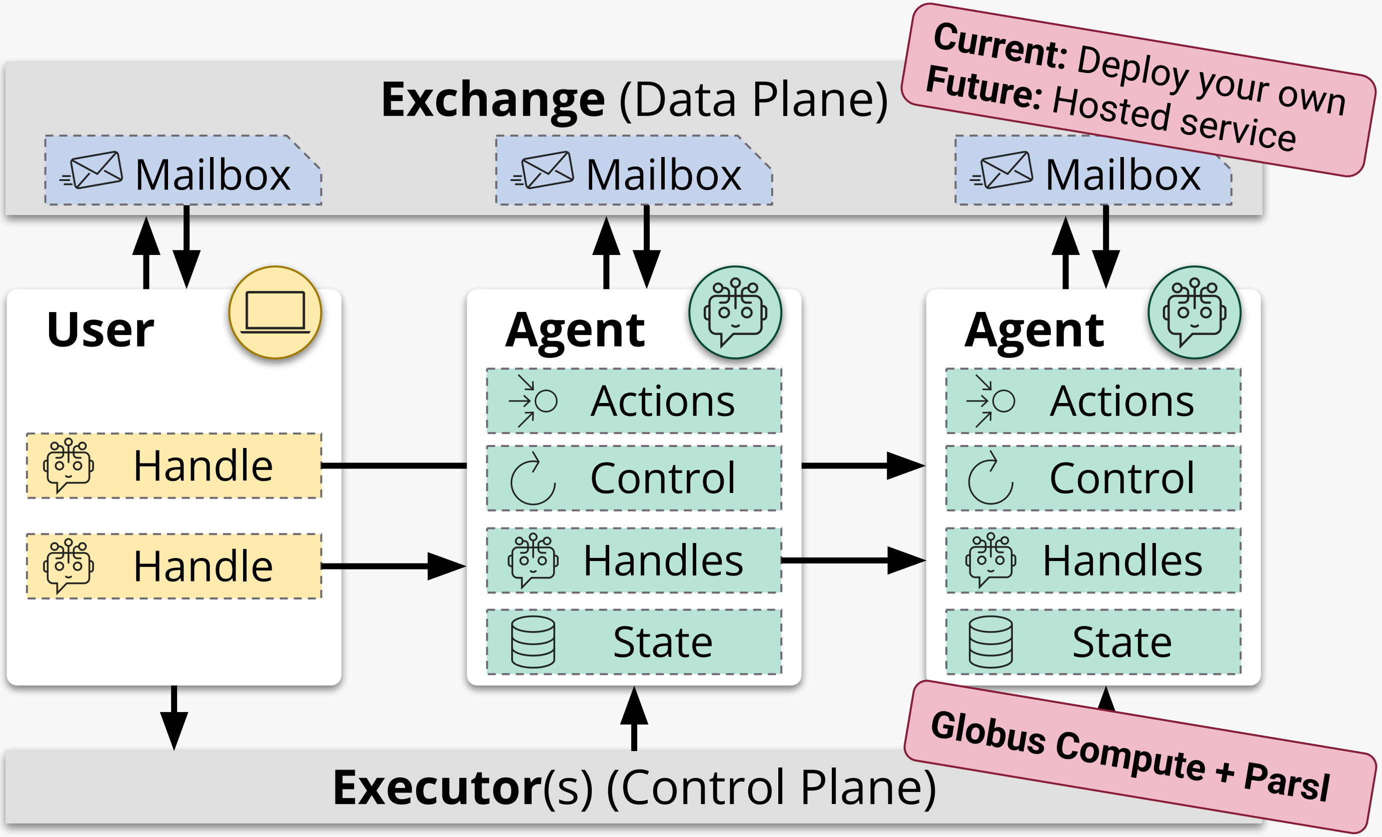
Academy

Design

Focus 3: Coordinate async agent messaging

Focus 1: Program diverse agents and interactions

Focus 2: Deploy agents on federated resources


Exchange (Data Plane)
Current: Deploy your own
Future: Hosted service
Mailbox
Mailbox
Mailbox
User
Handle
Handle
Agent
Actions
Control
Handles
State
Agent
Actions
Control
Handles
State
Globus Compute + Parsl
Executor(s) (Control Plane)

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

Agents defined by a class

Clients & other agents can request actions

Instance of an agent is state

Control loops for autonomous behavior

```python
import asyncio
from academy.agent import Agent, action, loop

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

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

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

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

Single interface for managing your agents

Launch agent and get handle

Interact with agents via handles

```python
from academy.exchange.redis import RedisExchangeFactory
from academy.manager import Manager
from globus_compute_sdk import GlobusComputeExecutor

gce = GlobusComputeExecutor('<UUID>')

async with await Manager.from_exchange_factory(
    factory=RedisExchangeFactory('localhost', 6379),
    executor=gce,
) as manager:
    handle = await manager.launch(Example)

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

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

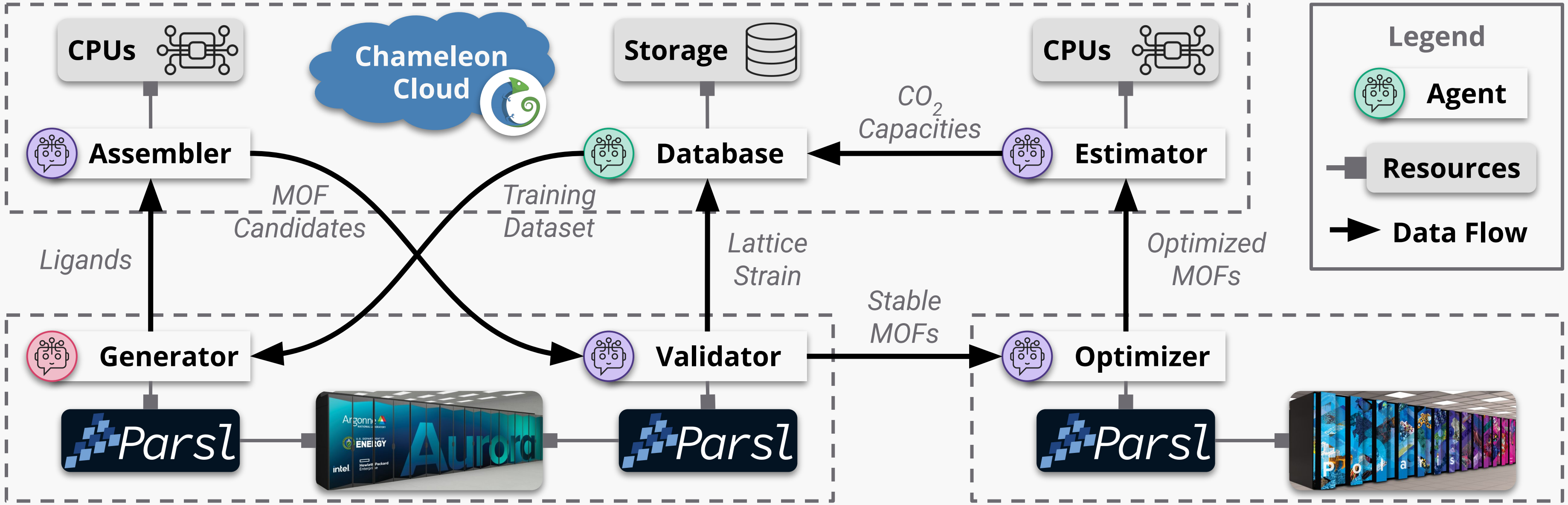
Launch agents via Globus Compute

Pass handles to other agents

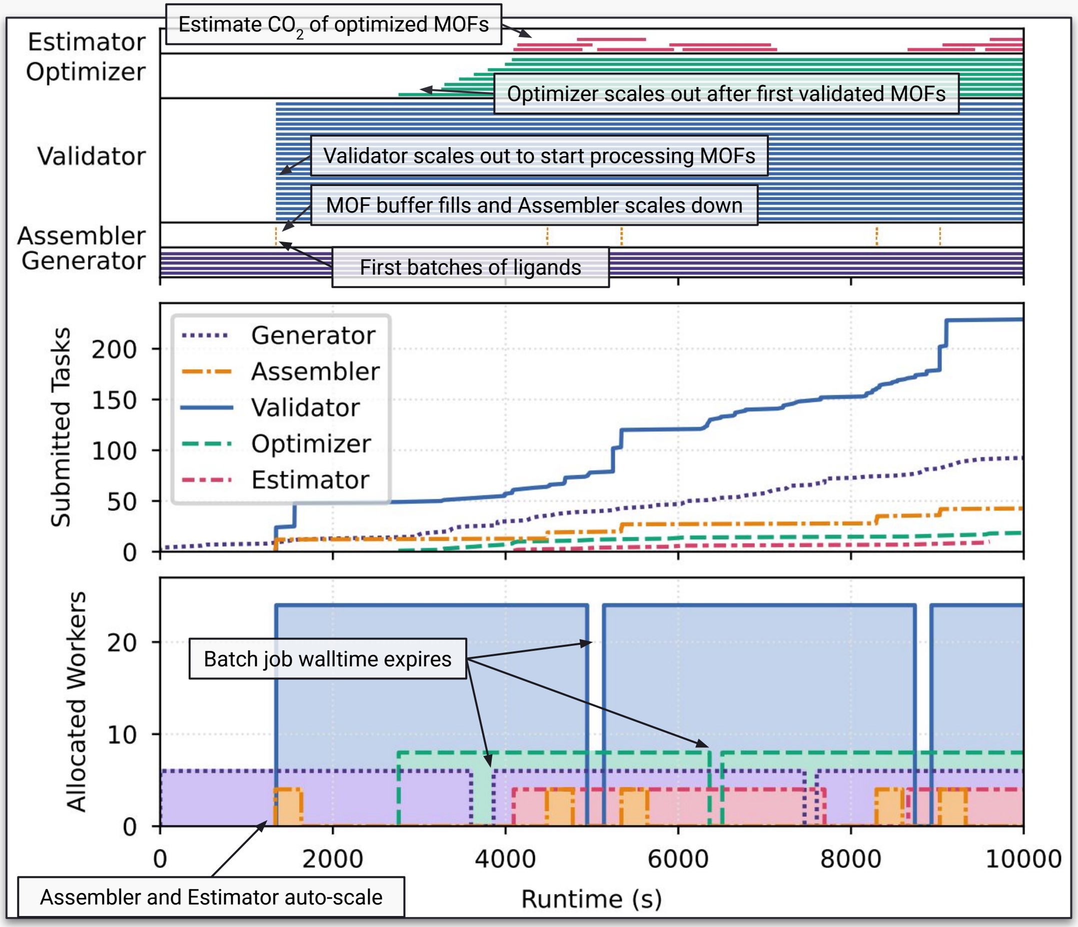
https://academy.proxystore.dev/latest/get-started/

Application

MOF Discovery through Autonomous Agents



Agents executed remotely via Globus Compute



Agentic Workflow 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)

Academy

J. Gregory **Pauloski**

Alok **Kamatar**

Ryan **Chard**

Yadu **Babuji**

Mansi **Sakarvadia**

Kyle **Chard**

Ian **Foster**

Reach out if you are interested:

`chard@uchicago.edu`

Learn more/stay up to date:

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

SC25 St. Louis, MO | hpc ignites.

Attend our tutorial!
Sunday 8:30am-12pm

⭐ Academy on GitHub!