Systems Requirements for Scalable Agentic AI

Ian Foster

THE UNIVERSITY OF CHICAGO

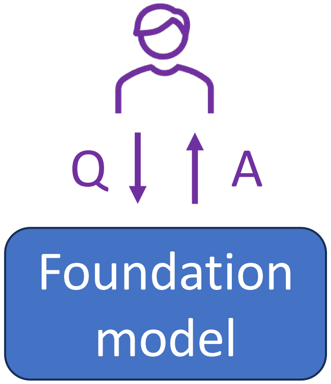
globus labs

Argonne NATIONAL LABORATORY

AI models: To infinity and beyond?

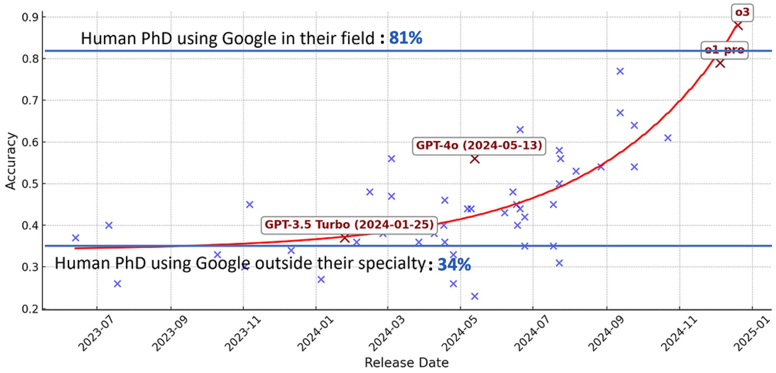
AI progress is commonly framed around models

- Scale, parameters, benchmarks
- Models as stateless inference engines
- Execution assumed to be request–response
- Concerns: Scalable training, inference



Q A

Foundation model



This framing is increasingly incomplete

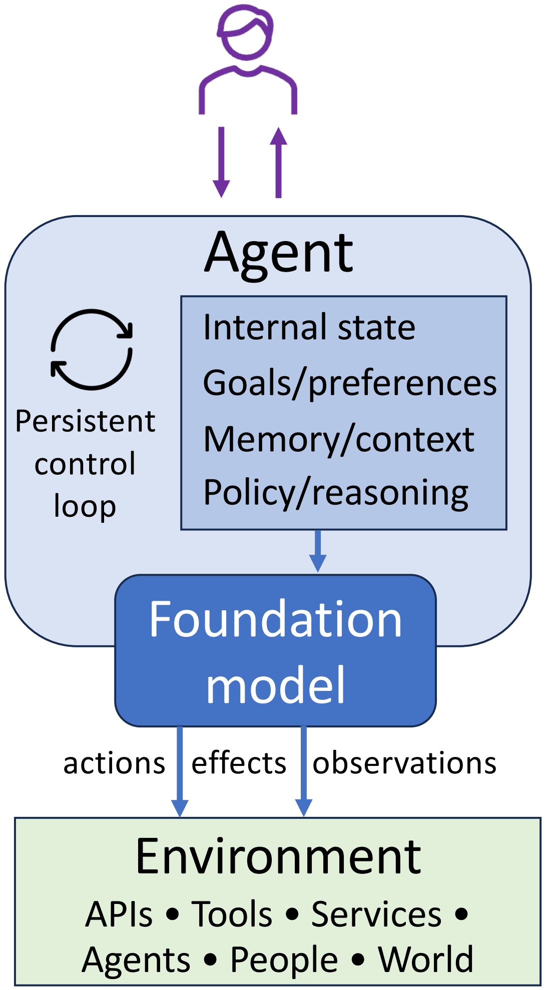
From models to agents

Deployed systems increasingly:

- Persist over time
- Initiate actions autonomously
- Interact continuously with tools, APIs, people
- Accumulate state and context

These systems behave as **agents**

*We need to enable agentic systems to **scale** and to **engage with the science ecosystem***

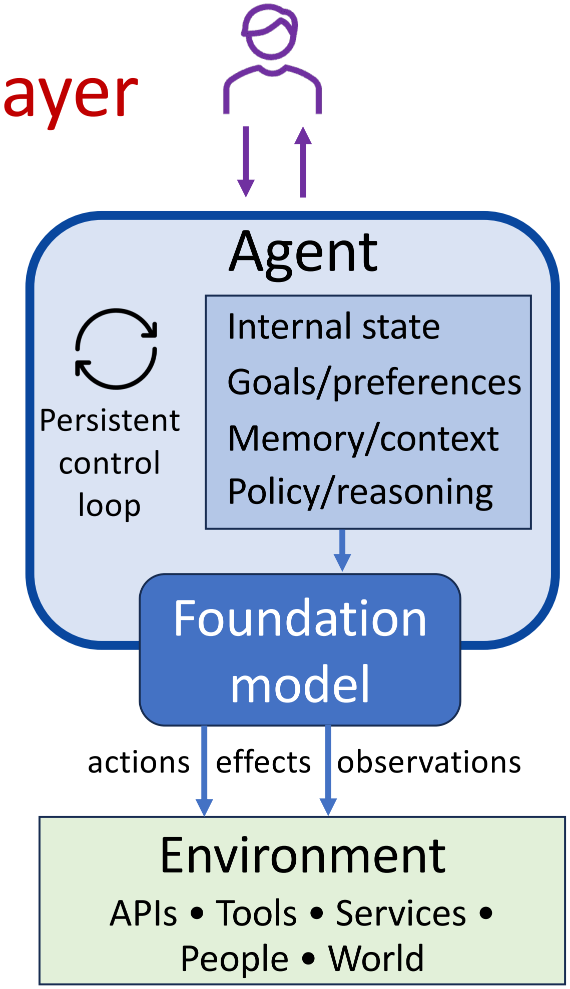


"Agents" are not just an application layer

Agentic systems reorganize computation

- Control flow moves to inside the system
- Responsibility shifts from caller to agent
- Time horizon expands beyond individual executions

This reorganization results in new demands for tools and facilities

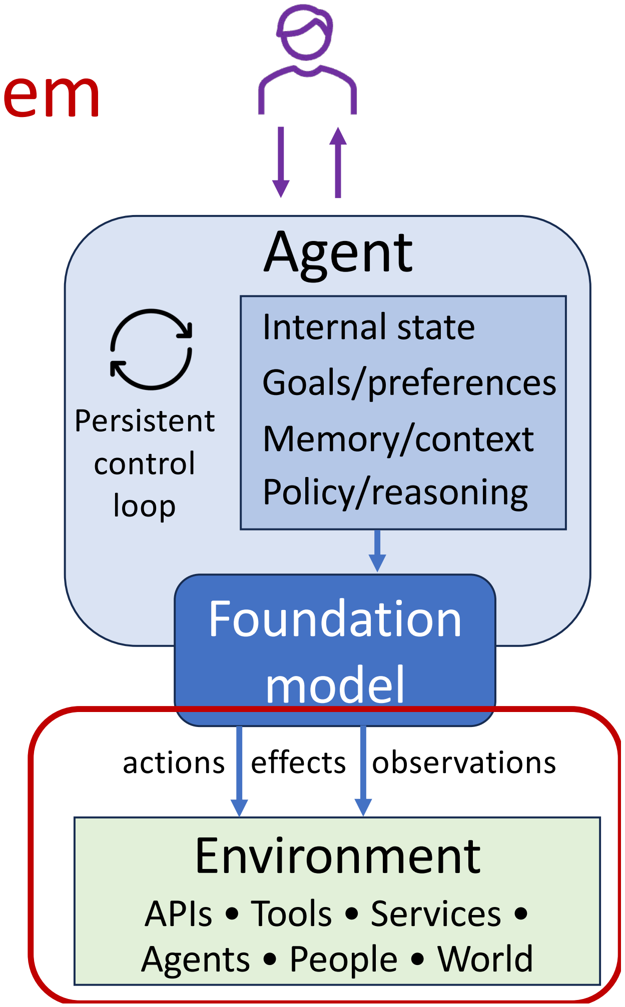


Agents engage with science ecosystem

An agent, like a human researcher:

- must be able to access the diverse elements of the modern scientific ecosystem
- may act as a generator of heterogeneous workloads: LLM calls, HPC jobs, service calls, data transfers, instrument actions, …
- must be managed to avoid excessive use of scarce resources

These are not concerns specific to "intelligence," but AI agents result in new challenges

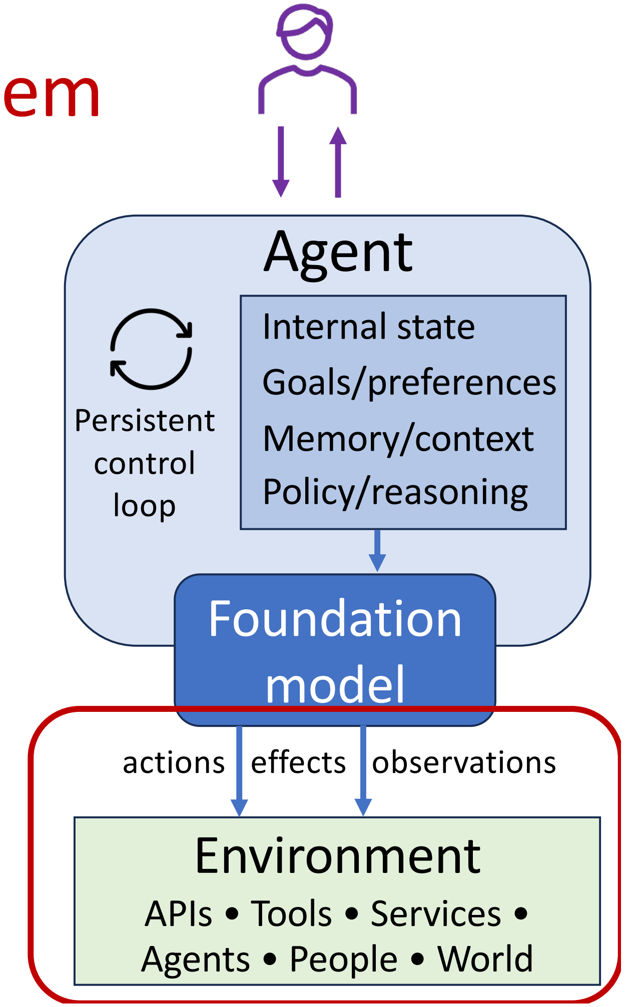


Agents engage with science ecosystem

An agent, like a human researcher:

- **must be able to access the diverse elements of the modern scientific ecosystem**
- may act as a generator of heterogeneous workloads: LLM calls, HPC jobs, service calls, data transfers, instrument actions, …
- must be managed to avoid excessive use of scarce resources

These are not concerns specific to "intelligence," but AI agents result in new challenges



Agentic orchestration: Enabling agent actions

Delegation & identity

Agents act on behalf of scientists, securely and with scoped permissions

Workflow control

Agents run logic-rich flows, with conditionals, retries, parallel tasks

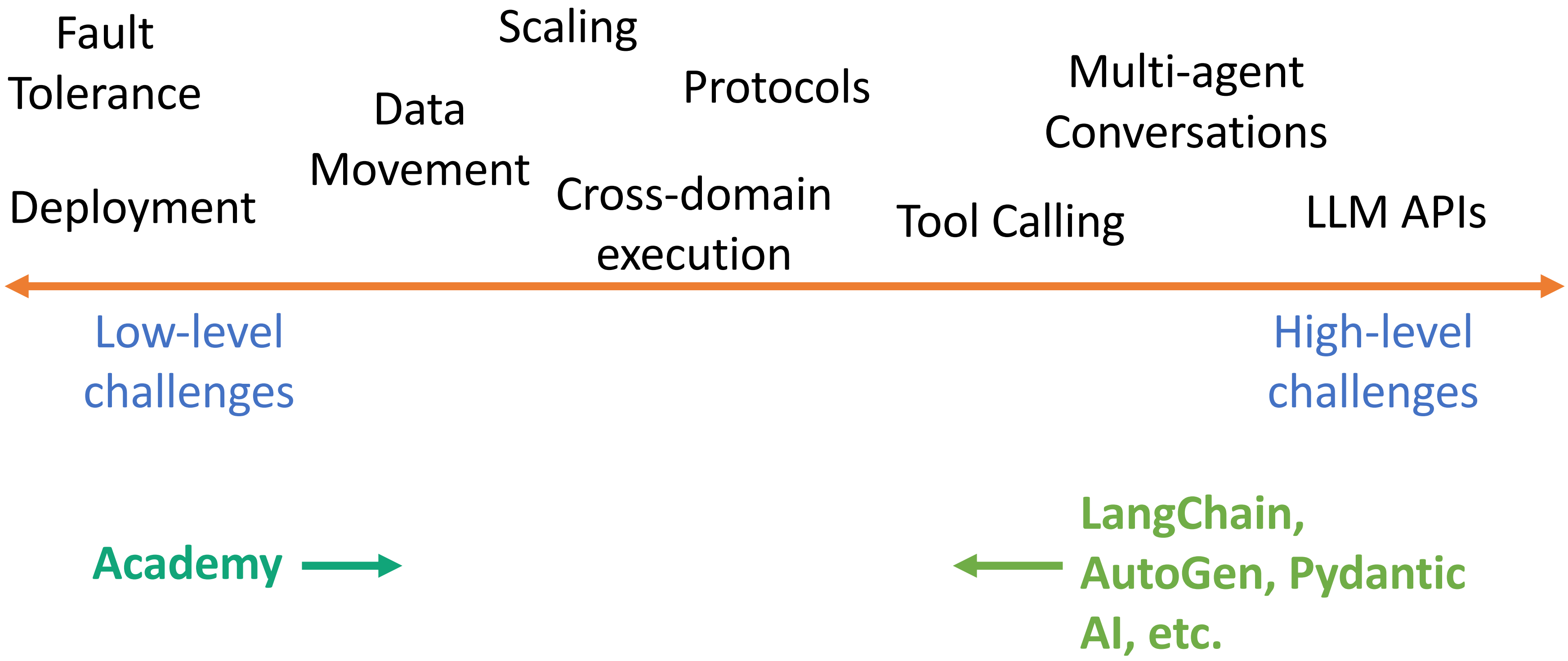
Cross-domain execution

Agents across labs, clouds, and instruments via federated middleware

Audit & policy boundaries

Every action is logged, reversible, and bound by policy (zero-trust)

Agentic middleware: Scope and challenges



Fault Tolerance

Deployment

Data Movement

Scaling

Cross-domain execution

Protocols

Tool Calling

Multi-agent Conversations

LLM APIs

Low-level challenges

High-level challenges

Academy

LangChain, AutoGen, Pydantic AI, etc.

Exploring agentic middleware: **Academy**

Greg Pauloski

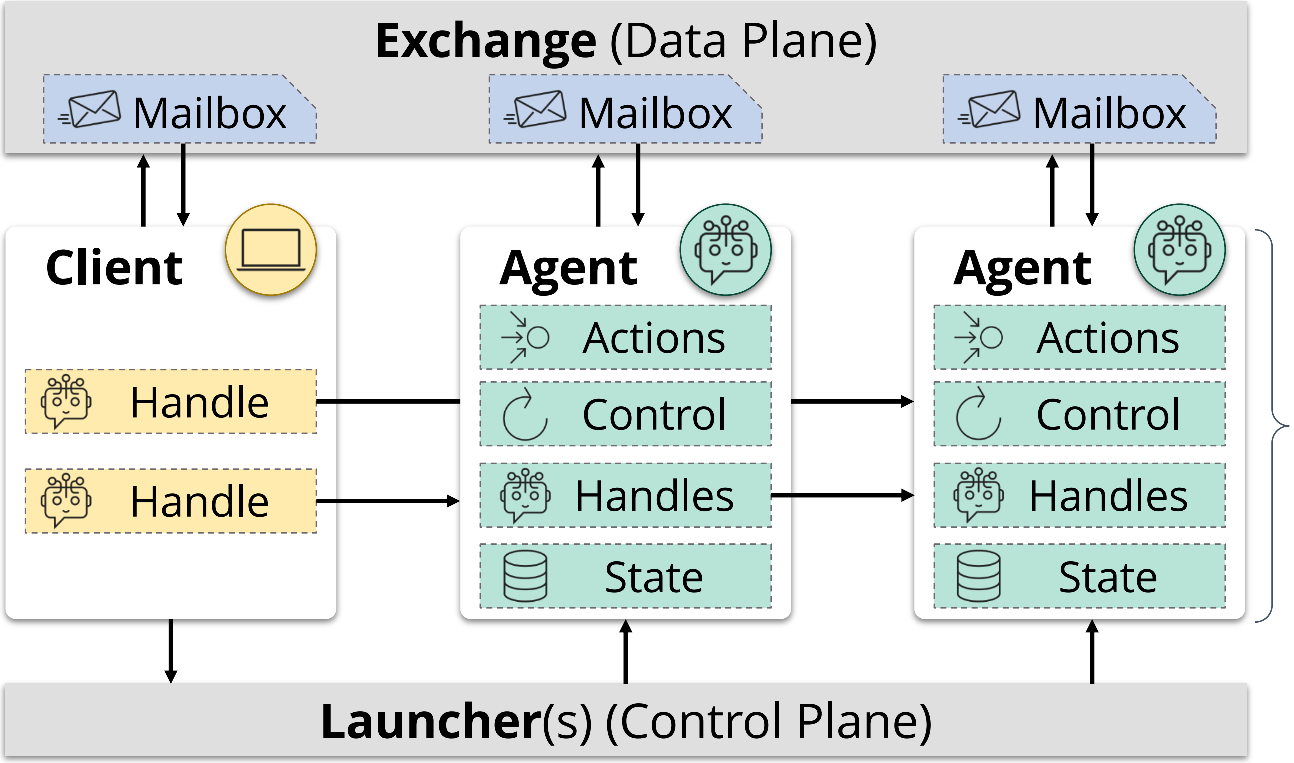
Kyle Chard

Alok Kamatar

Focus 3: How to coordinate async agent messaging

Focus 1: How to program arbitrary agents and their interaction

Focus 2: How to deploy agents on federated resources



Exchange (Data Plane)

Mailbox

Mailbox

Mailbox

Client

Handle

Handle

Agent

Actions

Control

Handles

State

Agent

Actions

Control

Handles

State

Launcher(s) (Control Plane)

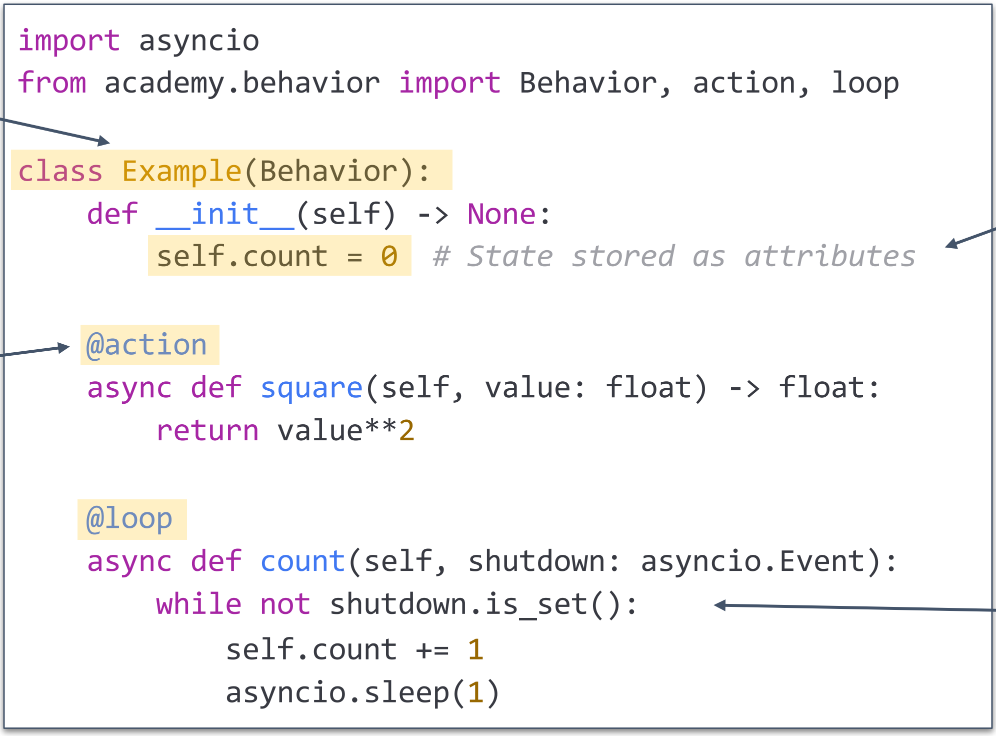
Broad agent definition

Not limited to LLMs / multi-agent conversations

https://academy-agents.org

Agents defined by a **behavior**

Clients & other agents can request **actions**



```python
import asyncio
from academy.behavior import Behavior, action, loop

class Example(Behavior):
    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):
        while not shutdown.is_set():
            self.count += 1
            asyncio.sleep(1)
```

Instance of a behavior is **state**

Control loops for autonomous behavior

https://docs.academy-agents.org/latest/get-started/

Communication and 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

Academy

HPC Centric Capabilities

- Secured with Globus Auth
- Agent coordination across HPC facilities
  - Cloud hosted exchange
- Agents with ability to run tools on HPC
- Agent sharing across users/groups
- Launch 1000s of agents

LLM Centric Capabilities

- Launch custom LLMs as agents
- Integrate agents from multiple frameworks (Langgraph, Pydantic...)
- Wrap science apps for function calling
- Expose apps via MCP
- Implement multi-agent communication patterns

Guides

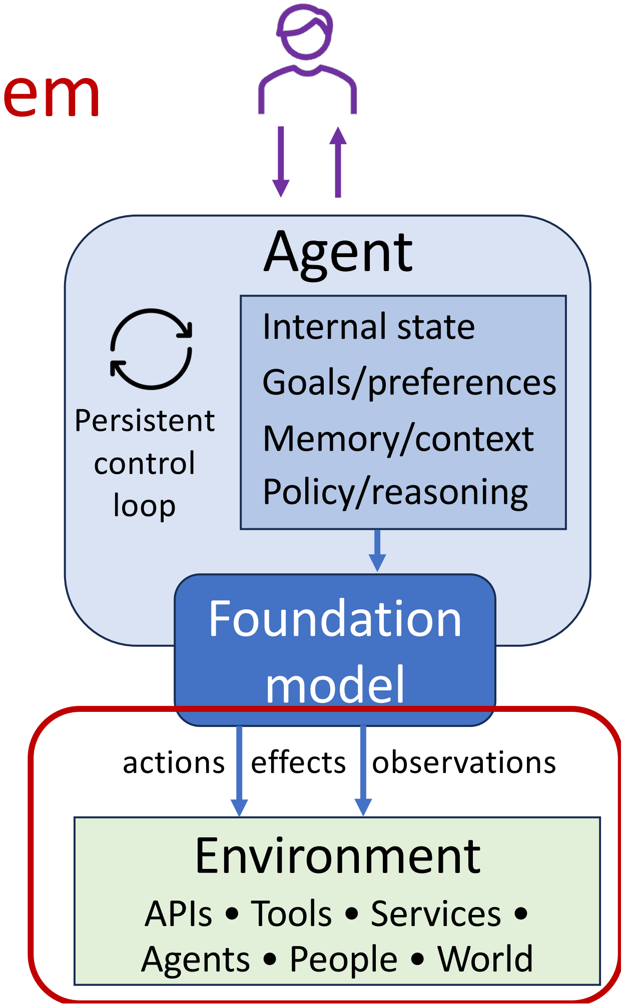
- https://docs.academy-agents.org/main/guides/hpc/
- https://docs.academy-agents.org/main/guides/llm/
- https://academy-agents.org/academy-extensions/latest/guides/mcp/

Agents engage with science ecosystem

An agent, like a human researcher:

- must be able to access the diverse elements of the modern scientific ecosystem
- may act as a generator of heterogeneous workloads: LLM calls, HPC jobs, service calls, data transfers, instrument actions, …
- must be managed to avoid excessive use of scarce resources

These are not concerns specific to "intelligence," but AI agents result in new challenges



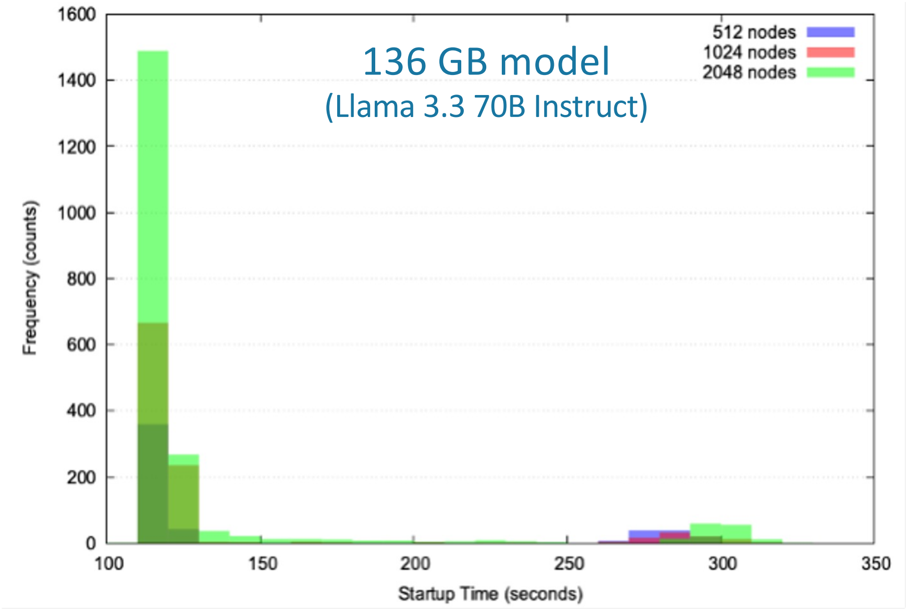
Scaling challenges

- We should anticipate thousands of autonomous agents, each able to (indeed, eager to) generate millions of heterogeneous tool invocations over long time horizons
- **Easy problem**: Scale tool discovery, deployment, invocation, monitoring
- **Hard problem**: Manage this new class of workload
  - Our facilities are designed to support work by humans, with resource use constrained by a mix of policy and human judgment
  - Do we need new abstractions and policies for software entities that decide what to call next?
  - Can this exploding complexity benefit from (or require?) AI?

Early work on the easy problem

Goal: Rapid deployment of LLMs (and LLM-based agents) on DOE supercomputers

Initial results: We leverage parallel I/O methods to **reduce vLLM startup time** on 2048 Aurora nodes from **many hours** to a **few minutes**


136 GB model

(Llama 3.3 70B Instruct)

Scalable token generation:

- Average **89 input tokens/sec/node**, **241 generated tokens/sec/node**
- Generate **1.44 billion tokens** in 35 mins on 2048 Aurora nodes

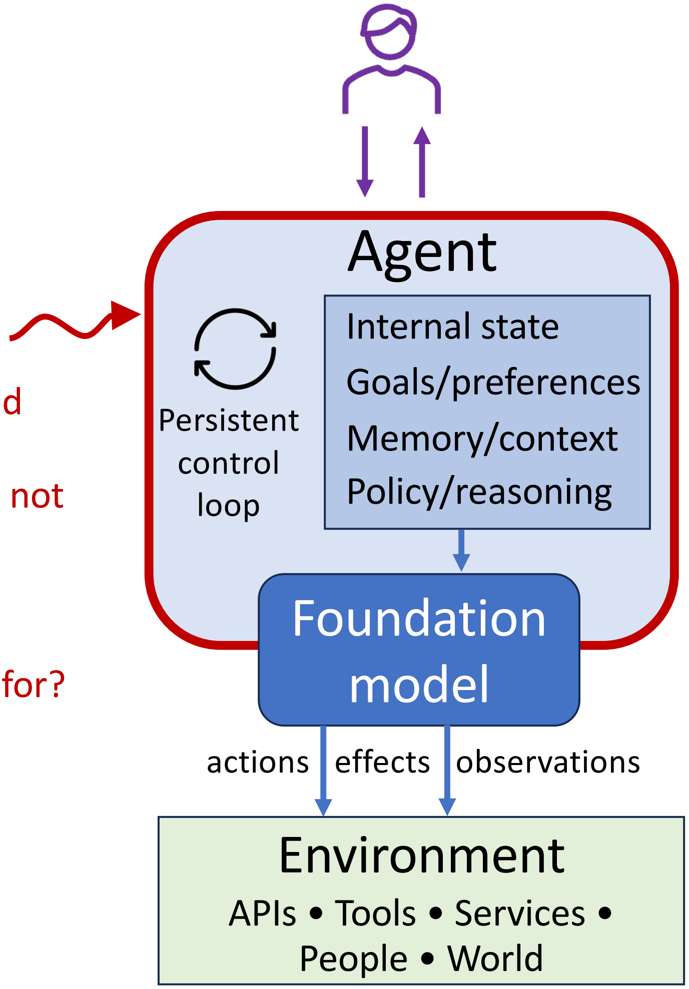
Next steps:

- Frontier and Perlmutter; MPI
- Rapid inter-agent communication
- Tool calling

New research problems

Beyond model capability and alignment

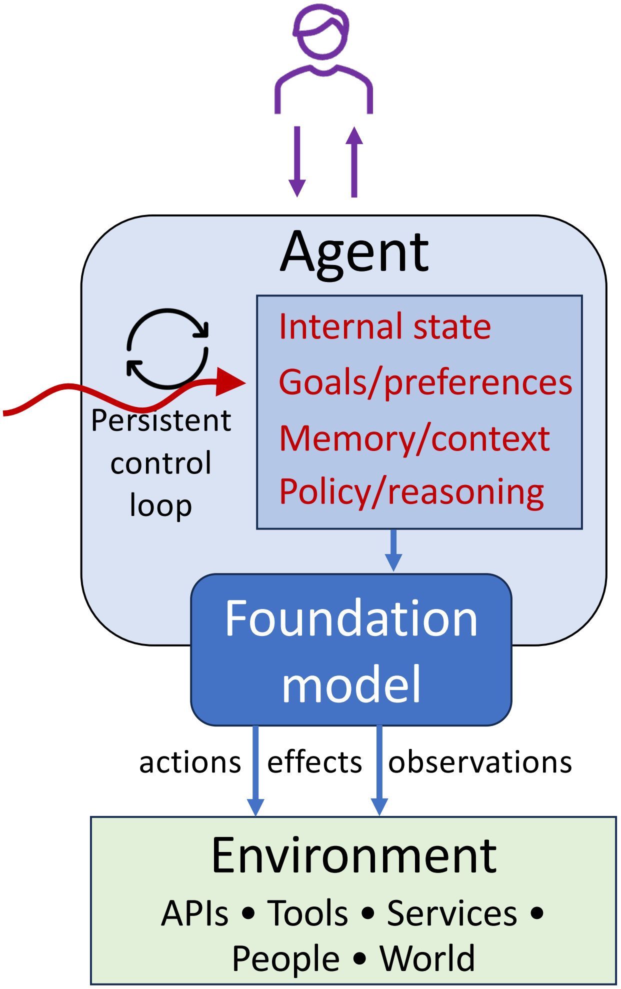
- Execution environments for persistent agents
  - How should resource budgets be expressed for self-initiated processes?
  - How can isolation and sandboxing be enforced proactively, not reactively?
  - What are principled semantics for pausing, checkpointing, migrating, and terminating agents?
  - How should agents that spawn other agents be accounted for?



New research problems

Beyond model capability and alignment

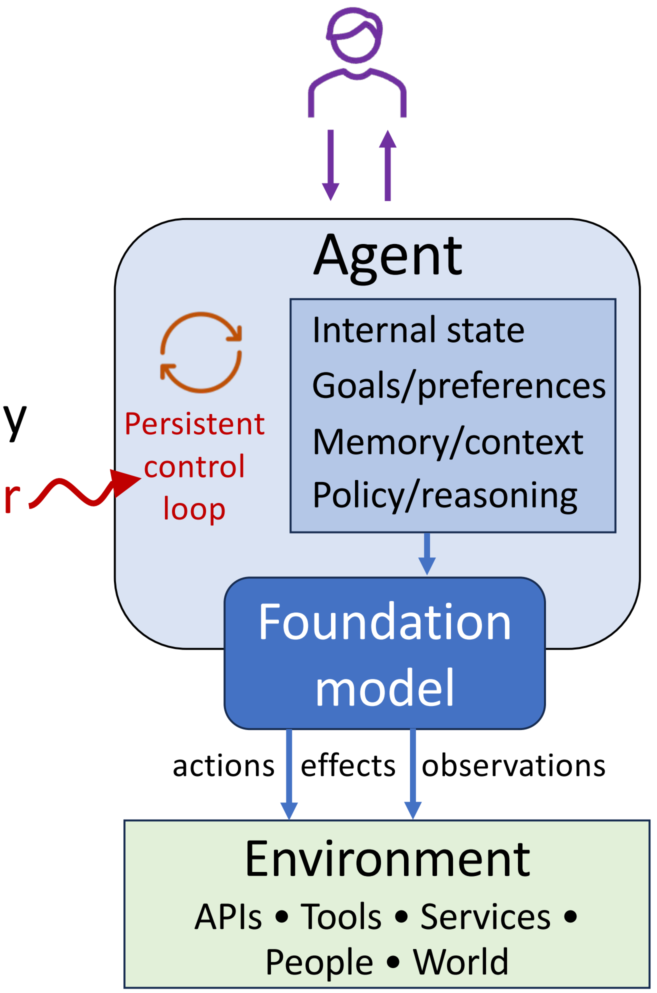
- Execution environments for persistent agents
- Programming models for constrained autonomy
  - How can goals, preferences, and prohibitions be expressed declaratively?
  - How do constraints remain binding as agents adapt strategies?
  - How should conflicts between objectives be resolved and exposed?
  - What is the boundary between agent discretion and system-enforced control?



New research problems

Beyond model capability and alignment

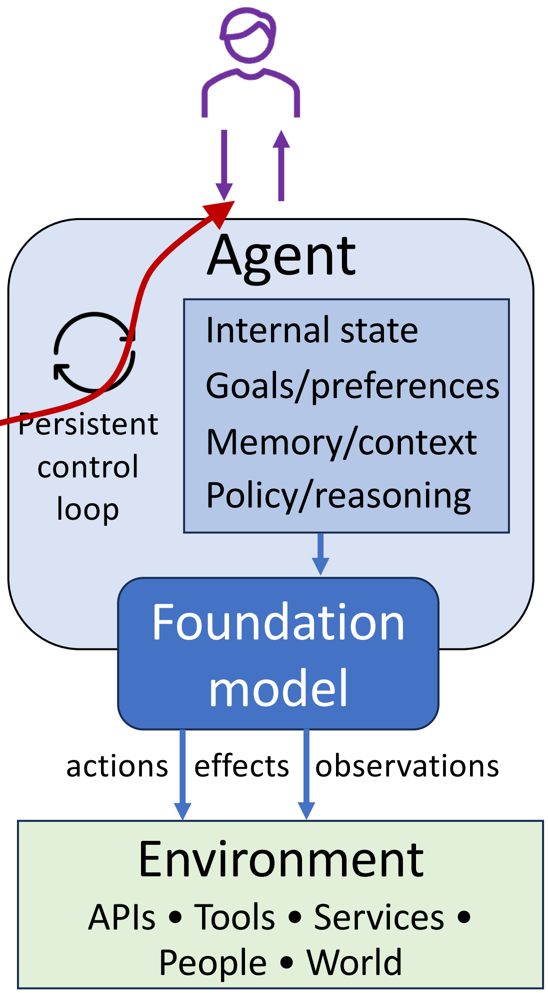
- Execution environments for persistent agents
- Programming models for constrained autonomy
- Correctness for adaptive, long-horizon behavior
  - What does correctness mean when behavior evolves over time?
  - How can safety envelopes or regret bounds replace binary correctness?
  - How do we verify properties over extended reasoning-action loops?
  - How should failure be attributed across long decision sequences?



New research problems

Beyond model capability and alignment

- Execution environments for persistent agents
- Programming models for constrained autonomy
- Correctness for adaptive, long-horizon behavior
- Oversight interfaces for continuous operation
    - How can agent behavior be summarized intelligibly over time?
    - When should agents escalate decisions to humans?
    - How can limited human attention be allocated across many agents?
    - How can systems support intervention without halting operation?

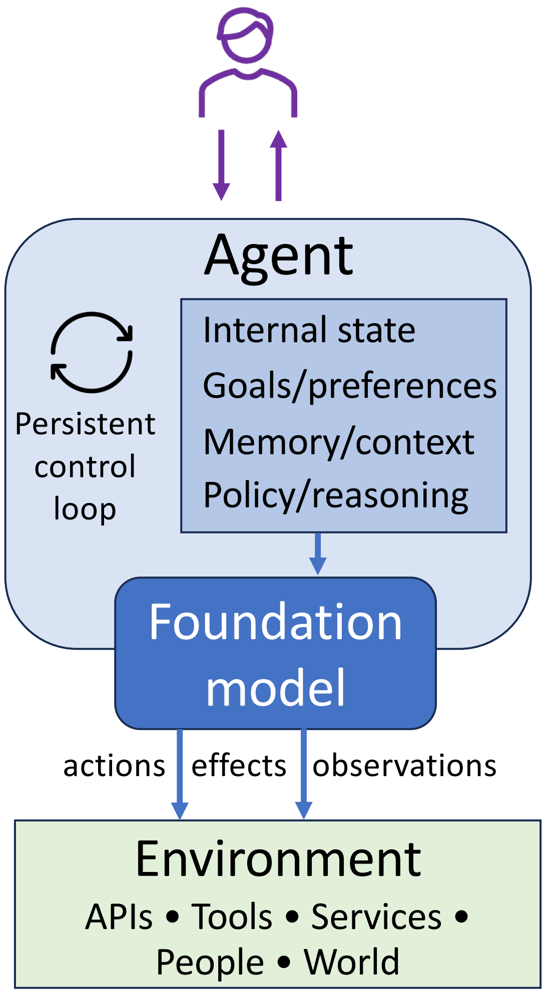


New research problems

Beyond model capability and alignment

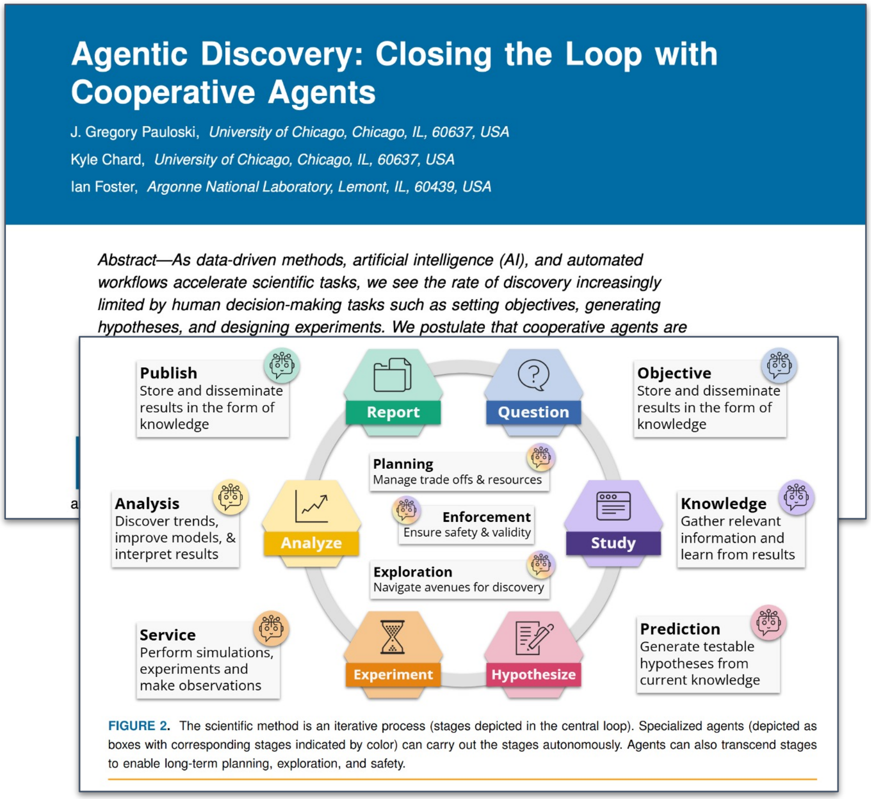
- Execution environments for persistent agents
- Programming models for constrained autonomy
- Correctness for adaptive, long-horizon behavior
- Oversight interfaces for continuous operation

These CS research problems span systems, theory, HCI, etc.—and AI



Summary: Agentic middleware challenges

- Access & privileges
- Agent discovery
- Asynchronous communication
- Fault tolerance
- Interfaces
- Mobility
- Persistent stateful execution
- Provenance
- …



Agentic Discovery: Closing the Loop with Cooperative Agents

J. Gregory Pauloski, *University of Chicago, Chicago, IL, 60637, USA*
Kyle Chard, *University of Chicago, Chicago, IL, 60637, USA*
Ian Foster, *Argonne National Laboratory, Lemont, IL, 60439, USA*

Abstract—As data-driven methods, artificial intelligence (AI), and automated workflows accelerate scientific tasks, we see the rate of discovery increasingly limited by human decision-making tasks such as setting objectives, generating hypotheses, and designing experiments. We postulate that cooperative agents are

Publish
Store and disseminate results in the form of knowledge
Report
Question
Objective
Store and disseminate results in the form of knowledge
Planning
Manage trade offs & resources
Analysis
Discover trends, improve models, & interpret results
Analyze
Enforcement
Ensure safety & validity
Study
Knowledge
Gather relevant information and learn from results
Exploration
Navigate avenues for discovery
Service
Perform simulations, experiments and make observations
Experiment
Hypothesize
Prediction
Generate testable hypotheses from current knowledge

FIGURE 2. The scientific method is an iterative process (stages depicted in the central loop). Specialized agents (depicted as boxes with corresponding stages indicated by color) can carry out the stages autonomously. Agents can also transcend stages to enable long-term planning, exploration, and safety.

Pauloski et al., IEEE Computer
https://arxiv.org/pdf/2510.13081

Summary: Opportunities for TPC

- Define agentic workloads as a class
- Establish benchmarks beyond throughput
- Define interfaces for execution control
- Collaborate on open source software for scalable orchestration
- Share agent implementations
- Coordinate cross-site experiments
- Align model, systems, and facilities communities