AI Agents for Science

Lecture 6, October 15: HPC Systems and Self-Driving Labs

Instructor: Ian Foster
TA: Alok Kamatar

Crescat scientia; vita excolatur

CMSC 35370 -- https://agents4science.github.io
https://canvas.uchicago.edu/courses/67079

The joys and pains of context stuffing

- RAG and MCP both stick lots of stuff in the LLM context, and then rely on the LLM/RM to sort out which of that stuff is relevant to a query
  - Both trust the model's internal attention mechanisms to select what's relevant
- How well does that work in practice?
  - Better than simply relying on parametric knowledge for knowledge-intensive tasks (i.e., tasks that humans could not reasonably be expected to perform without access to an external knowledge source)
  - But with well-known limitations
- Are there better ways?
  - Yes sometimes!

When does it work reasonably well?

- **Small, coherent retrievals**: If you retrieve a few highly relevant documents (say 3–5 short chunks), the model's self-attention can effectively "spot" the relevant spans
- **Tightly phrased queries**: When the prompt or question aligns semantically with retrieved text, models exhibit high lexical-semantic precision
- **High-end reasoning models** (e.g. GPT-4o, Claude-Opus, Gemini 1.5 Pro) can maintain 50–100 KB effective context with surprisingly good relevance discrimination

→ In these conditions, simple context injection can yield strong results: 70–90% of correct answer coverage in many RAG benchmarks

Diverse tasks from six domains: **Office** (e.g., spreadsheet analysis), **Lifestyle** (e.g., news retrieval), **Leisure** (e.g., video game inquiries), **Finance** (e.g., stock price monitoring), **Travel** (e.g., ticket search), **Shopping** (e.g., product recommendations). All:

(1) **Time-varying**: Tasks exhibit time-sensitive outcomes;
(2) **Long-horizon**: Tasks require multiple tools to complete;
(3) **Genuine utility**: Tasks address authentic user needs.

Model	Office	Leisure	Travel	Lifestyle	Finance	Shopping	Overall (%)
Claude-Sonnet-4-20250514	90.32	64.29	75.00	80.00	78.57	66.67	78.95
Claude-Opus-4-20250514	80.65	64.29	66.67	86.67	64.29	33.33	70.53
DeepSeek-R1-0528	41.94	50.00	58.33	46.67	50.00	55.56	48.42
Qwen3-235B-A22B	54.84	35.71	41.67	53.33	50.00	44.44	48.42
GPT-4.1-Mini	45.16	50.00	50.00	46.67	42.86	22.22	44.21
Qwen2.5-72B-Instruct	35.48	35.71	50.00	40.00	57.14	55.56	43.16
DeepSeek-V3-0324	41.94	42.86	50.00	40.00	28.57	55.56	42.11
Gemini-2.5-Pro	48.39	28.57	16.67	60.00	57.14	11.11	41.05
GPT-4.1	51.61	28.57	25.00	46.67	35.71	22.22	38.95
Qwen3-32B	29.03	14.29	25.00	53.33	28.57	33.33	30.53

https://arxiv.org/pdf/2508.01780

Table 2: Task success rate results for the frontier models. Evaluation using Deepseek-V3.

Number of example queries

Table 1: Statistics of Dataset Instance MCP Tools Count and Tokens

Category	Number Instance	MCP Tool Count	Tokens Per Tool	Total Tokens
Browser	187	32	107.4	3.4 K
File System	241	11	143.8	1.6 K
Search	181	5	555.6	2.8 K
Map	500	32	401.3	13 K
Finance	90	1	505.0	0.5 K
Pay	310	6	656.5	3.9 K
Total	1509	87	288.3	25 K

https://arxiv.org/pdf/2508.07575

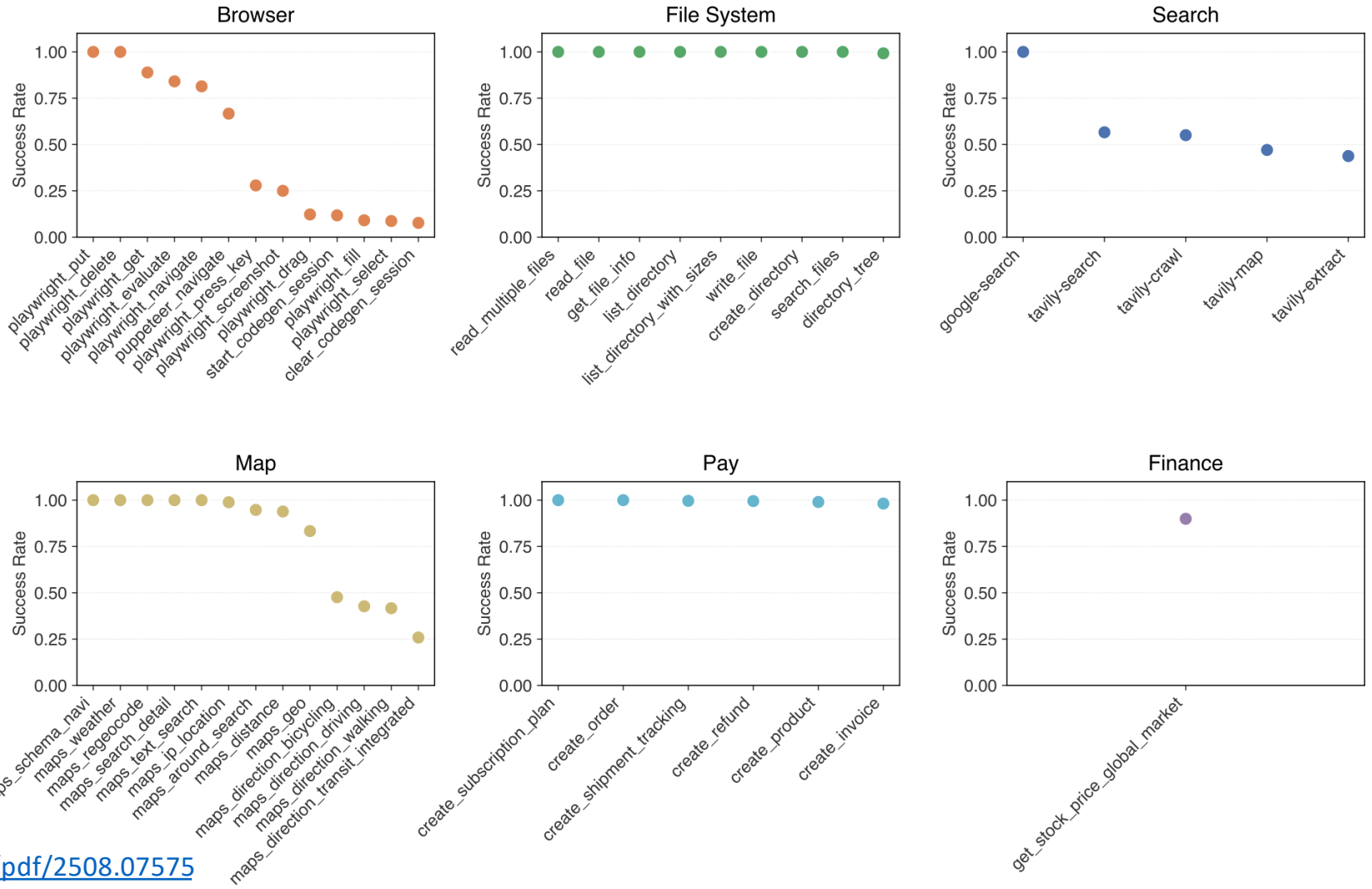
Calls correct tool with correct arguments

Gets correct answer

Table 2: Evaluation of MCP Tool Call Abstract Syntax Tree (AST) and Pass@K

	Browser		File System		Search	
Model	AST	Pass@1	AST	Pass@1	AST	Pass@1
GPT-4o	0.6524	0.2182	0.8863	0.8232	0.5200	0.4720
Qwen2.5-max	0.7262	0.2749	0.9419	0.8871	0.6280	0.4600
Claude-3.7-Sonnet	0.6503	0.1840	0.8415	0.8183	0.7280	0.6200
Kimi-K2-Instruct	0.8182	0.2524	0.9062	0.8772	0.7320	0.3680
Qwen3-coder	0.8866	0.2925	0.9080	0.8680	0.7180	0.5227
	Map		Pay		Finance	
Model	AST	Pass@1	AST	Pass@1	AST	Pass@1
GPT-4o	0.6120	0.3616	0.7077	0.5742	0.7200	0.2889
Qwen2.5-max	0.7372	0.2272	0.6684	0.5277	0.7511	0.2556
Claude-3.7-Sonnet	0.5820	0.2748	0.7058	0.5574	0.7400	0.2311
Kimi-K2-Instruct	0.6088	0.2008	0.8071	0.6761	0.7156	0.2378
Qwen3-coder	0.7830	0.3054	0.7240	0.5440	0.7320	0.2860

https://arxiv.org/pdf/2508.07575



https://arxiv.org/pdf/2508.07575

When does it break down?

- **Context overflow and dilution**
  - As the number of retrieved or tool-provided items grows, the model's attention becomes diffuse. Even if the context window is large (1 M tokens or more), relevance doesn't scale linearly: the model struggles to "ignore" irrelevant snippets.
- **No structured grounding**
  - The model may **hallucinate** relationships between documents that were merely co-located in context. There's no formal representation of provenance or logic
- **Implicit reasoning load**
  - The LLM must both decide what's relevant and perform the reasoning in one pass, which can be cognitively challenging
- **Token efficiency and cost**
  - For large tool ecosystems (like MCP servers with dozens of resource schemas), naive context-stuffing becomes computationally and financially expensive

Why do RAG and MCP still do it?

- It is **simple and flexible**:
  - The model's attention acts as an implicit *information retrieval and fusion layer*
  - You don't need special index structures or symbolic reasoning.
  - It is model-agnostic: any LLM can "read" context text.
- That simplicity is powerful for prototyping or for small-scale pipelines (like MCP servers serving schema-validated JSONs into context)
- But for scientific or multi-agent systems, limits become visible quickly

An example refinement: Structured retrieval and routing

Instead of throwing everything into context, use a *retrieval router*:

- Perform retrieval and scoring per schema or tool
- Let an **agentic planner** choose which retrieved results to pass downstream
- Adapt routing algorithm based on results over time

Phase	**Component**	**Function**
① Query reception	Router	Performs & scores *retrieval per schema/tool*
② Planning	Agentic planner	Chooses which results/tools to use, and how
③ Execution	Executor(s)	Runs chosen tools or models
④ Feedback	System logs, verifier	Reports success, latency, quality
⑤ Adaptation	Router	Updates scoring weights and schemas via feedback

Scientific example

User: “Estimate adsorption energy of CO on Pt(111) at 300 K.”

- **Retrieval Router**
  - Finds candidate tools and contexts: CatalystDB (data), LAMMPS (simulation)
  - Returns scored contexts to planner
- **Agentic Planner**
  - Decides: “Retrieve existing data from CatalystDB. If uncertainty > 0.2 eV, run simulation via LAMMPS”; constructs that two-step plan
- **Executor**
  - Executes the plan and reports success and performance
- **Feedback**
  - Router updates tool weights (LAMMPS success +1; CatalystDB coverage −0.1)
  - Planner logs that simulation accuracy improved overall plan quality

A few other ideas

- **Evaluation and Feedback Loops**
  - **Human-in-the-loop correction:** When routing confidence is low, ask user to confirm or clarify ("Did you mean the *LAMMPS simulator* or *QuantumEspresso*?")
  - **Confidence scoring and logging:** Store embeddings of queries and their matched tools; analyze misroutes to refine descriptions or retrain the matcher

- **Long-Term Improvements**
  - **Tool descriptions co-evolve with usage:** Auto-summarize past successful invocations into richer tool documentation
  - **Federated resource directories:** Shared registries across labs or institutions, with consistent tagging, so that routers can learn from aggregate usage

Curriculum

1) Why AI agents for science?

AI agents and the sense-plan-act-learn loop. Scientific Discovery Platforms (SDPs): AI-native systems that connect reasoning models with scientific resources.

2) Frontiers of Language Models

Surveys frontier reasoning models: general-purpose LLMs (GPT, Claude), domain-specific foundation models (materials, bio, weather), and hybrids. Covers techniques for eliciting better reasoning: prompting, chain-of-thought, retrieval-augmented generation (RAG), fine-tuning, and tool-augmented reasoning.

3) Systems for Agents

Discusses architectures and frameworks for building multi-agent systems, with emphasis on inter-agent communication, orchestration, and lifecycle management.

4) Retrieval Augmented Generation (RAG) and Vector Databases

Covers how to augment reasoning models with external knowledge bases, vector search, and hybrid retrieval methods.

Curriculum

5) Tool Calling

Introduces methods for invoking external tools from reasoning models. Focus on model context protocol (MCP), schema design, and execution management.

6) HPC Systems and Self Driving Labs

How SDPs connect to HPC workflows and experimental labs. Covers distributed coordination, robotics, and federated agents.

7) Human–AI Workflows

Explores how scientists and agents collaborate: trust boundaries, interaction design, and debugging.

8) Benchmarking and Evaluation

Frameworks for assessing agents and SDPs: robustness, validity, and relevance.

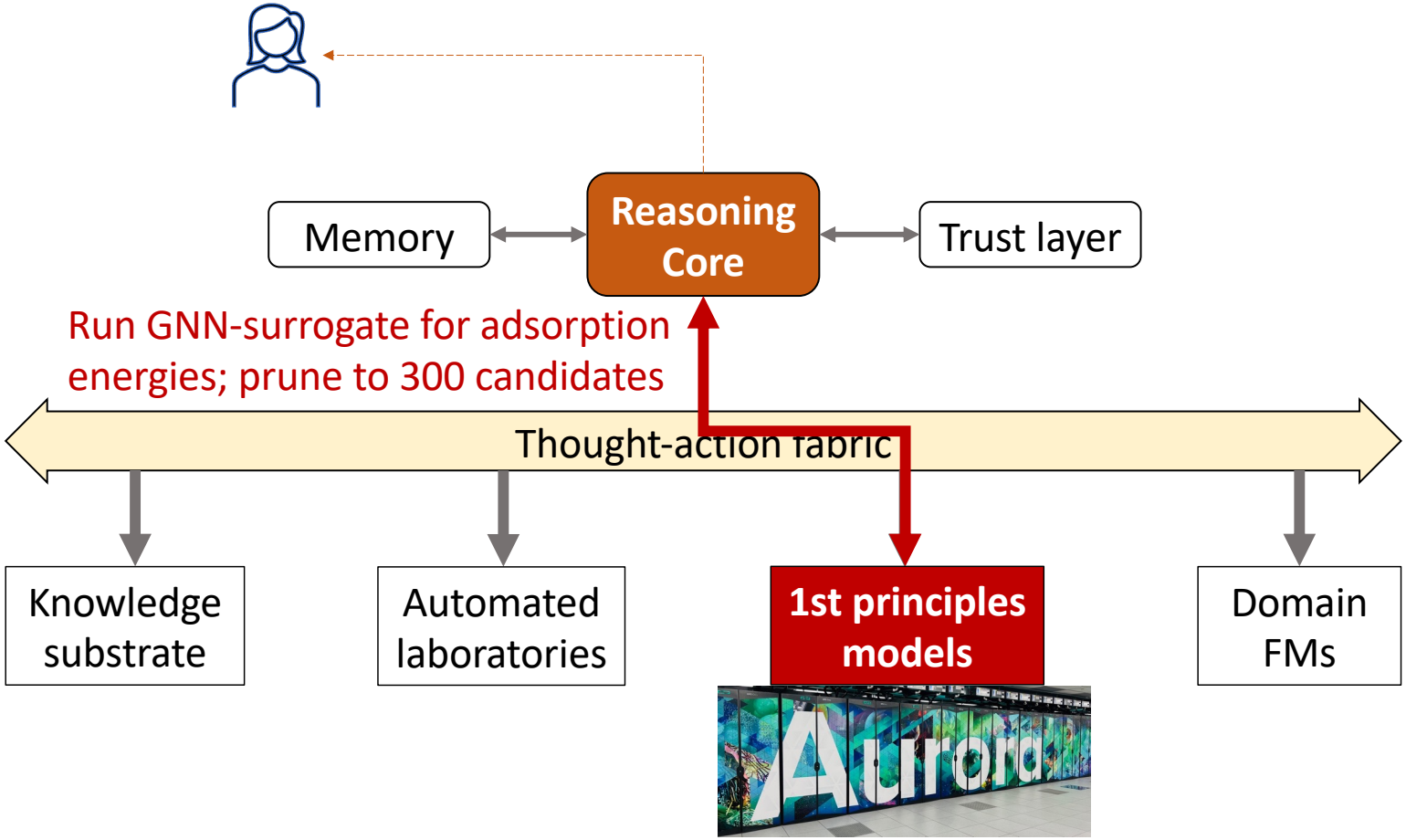
Readings and assignment

- *Self-Driving Laboratories for Chemistry and Materials Science*, *Chemical Reviews*

- *Empowering Scientific Workflows with Federated Agents*

Agents and HPC

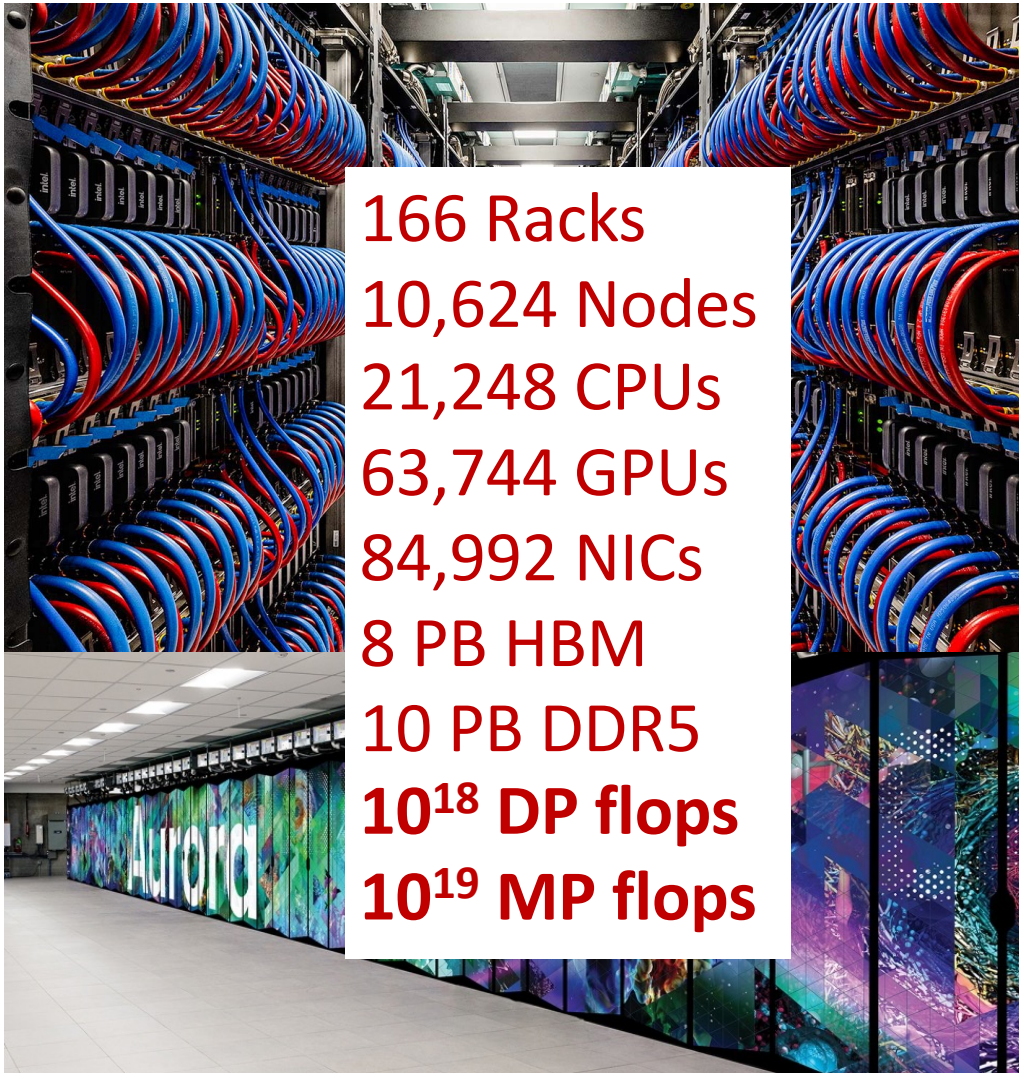
AI-native Scientific Discovery Platform



High-performance computing

- A high-performance computing (HPC) system is designed to perform large-scale, complex, or time-critical computations much faster than a typical personal computer or server
- It achieves this by combining many powerful processors, large amounts of memory, high-speed networking, and efficient software to act as a single unified machine

E.g., Aurora at Argonne →



Argonne AI testbed

GroqRack (Available for Allocation Requests)

GroqRack Inference

System Size: 72 Accelerators (9 nodes x 8 Accelerators per node)

Compute Units per Accelerator: 5120 vector ALUs

Performance of a single accelerator (TFlops): >188 (FP16) >750 (INT8)

Software Stack Support: GroqWare SDK, ONNX

Interconnect: RealScale TM

Cerebras CS-2 (Available for Allocation Requests)

Cerebras CS-2 Wafer-Scale Cluster WSE-2

System Size: 2 Nodes (each with a Wafer scale engine) including Memory-X and Swarm-X

Compute Units per Accelerator: 850,000 Cores

Performance of a single accelerator (TFlops): >5780 (FP16)

Software Stack Support: Cerebras SDK, Tensorflow, Pytorch

Interconnect: Ethernet-based

SambaNova Dataflow (Available for Allocation Requests)

SambaNova DataScale SN30

System Size: 64 Accelerators (8 nodes and 8 accelerators per node)

Compute Units per Accelerator: 1280 Programmable compute units

Performance of a single accelerator (TFlops): >660 (BF16)

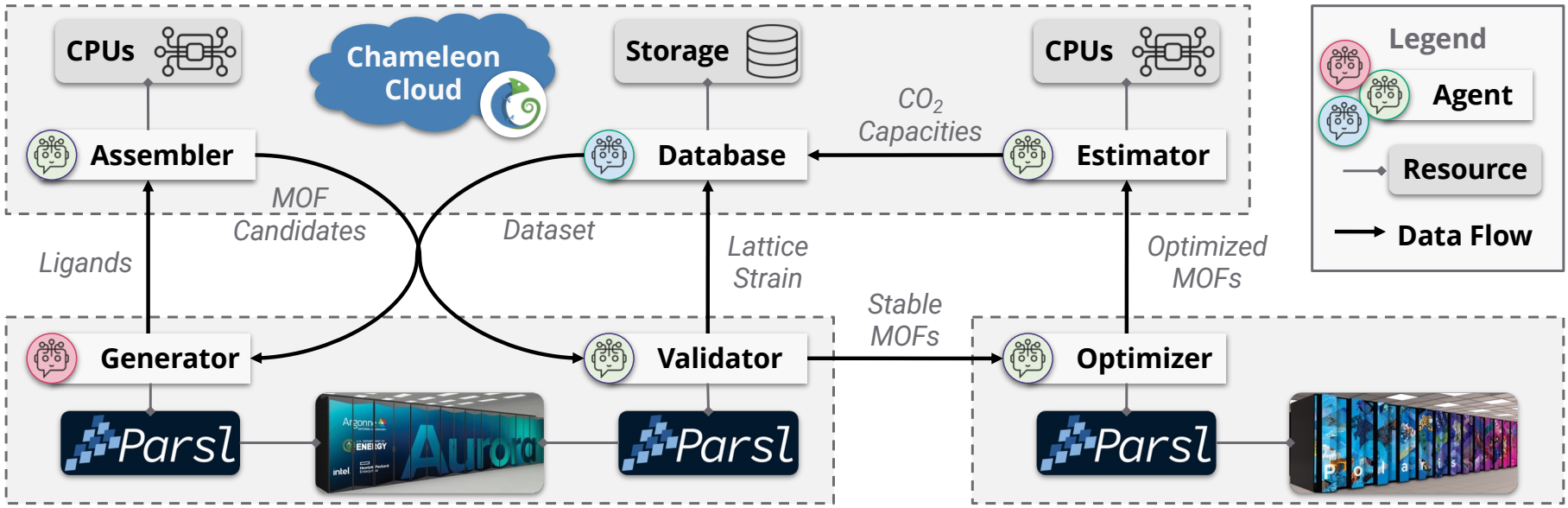
Software Stack Support: SambaFlow, Pytorch

Interconnect: Ethernet-based

Why HPC in a scientific discovery platform?

- Run detailed computational simulations of physical phenomena
  - Individual simulations to study distinct systems in great detail
  - Many simulations to characterize many different systems
- Train and fine-tune LLMs and other foundation models
- Large-scale inference with LLMs, RMs, other foundation models
- Analyze enormous quantities of data
- Run specialized agents

Run specialized agents: e.g., MOFA



HPC systems have specialized environments

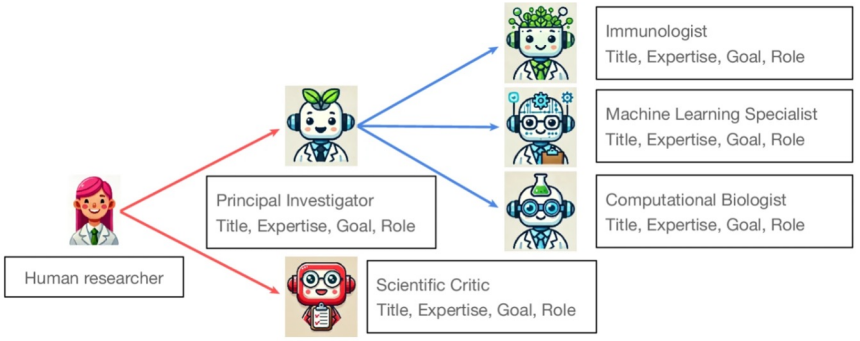
- Batch schedulers
- Globus Compute

Academy: Empowering Scientific Workflows with Federated Agents

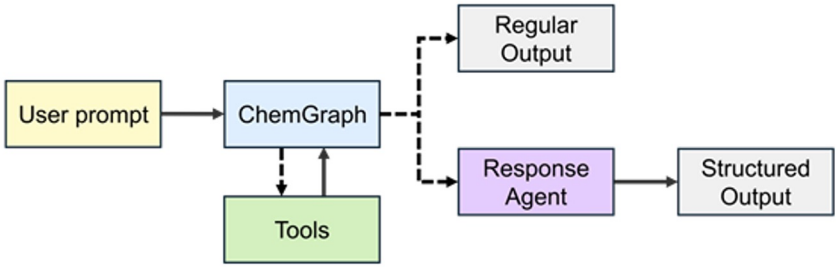
Greg Pauloski, **Alok Kamatar**, Yadu Babuji, Ryan Chard, Mansi Sakarvadia,
Kyle Chard, Ian Foster

Diaspora

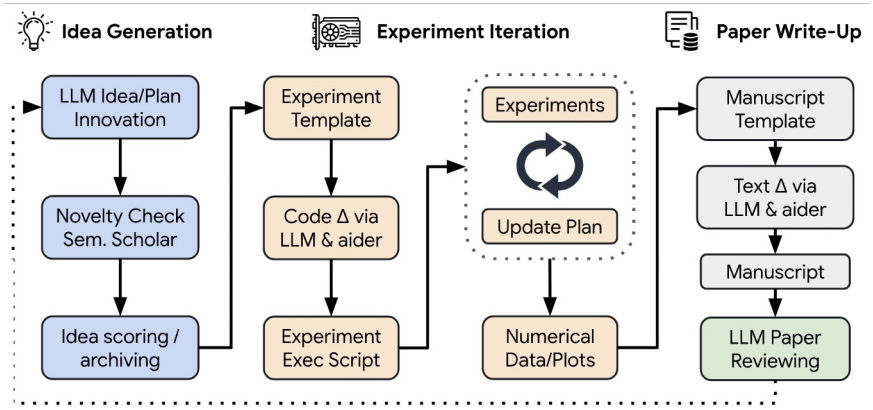
Multi-Agent Systems



From: The Virtual Lab of AI agents designs new SARS-CoV-2 nanobodies (Nature, 2025)



From: ChemGraph: An Agentic Framework for Computational Chemistry Applications (2506.06363)



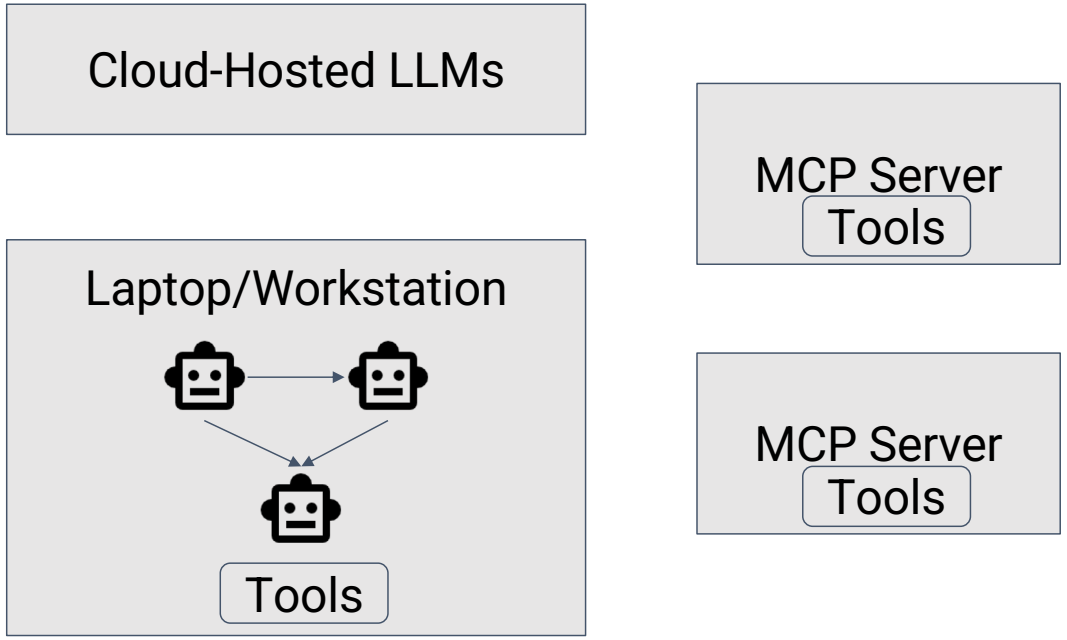
From: The AI-Scientist V2: Workshop Level Automated Discovery via Agentic Tree Search (2504.08066)

Centralized Model of Multi-Agent Systems

"Agents" are typically LLMs with separate context

All run in a centralized location

Use tools either locally or through MCP servers



LangChain Under the Hood

Multi-agent systems define a **graph!**

Pregel: Large-Scale Graph Processing

- Divide computation into “supersteps”
- At each node:
  - Process incoming message
  - (optionally) send message to neighbor
  - (optionally) vote to halt

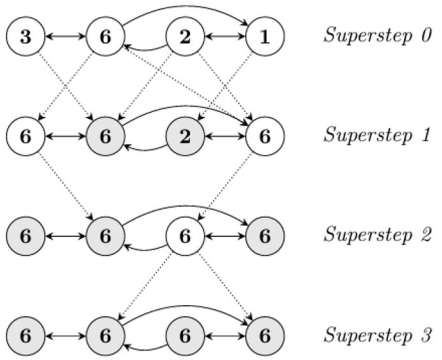


Figure 2: Maximum Value Example. Dotted lines are messages. Shaded vertices have voted to halt.

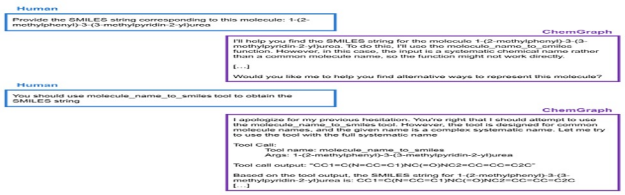
Actor Systems

More generally, Agents are built on **Actor Systems**

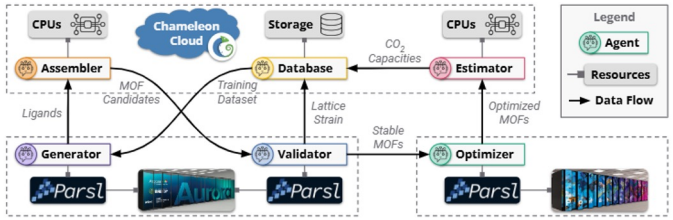
- Actors: computational entities that communicate through message passing
- On receiving a message an actor can:
  - Send a finite number of messages
  - Create a finite number of actors
  - Update their state
- No shared state
- Inherently concurrent

Scientific Agents Beyond LLMs

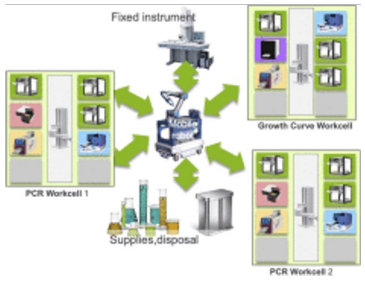
Conversational Assistants



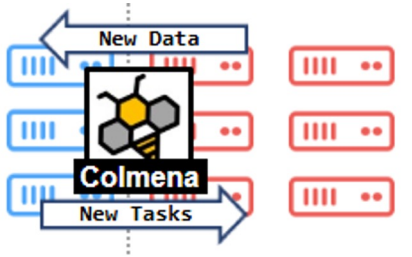
Multi-Site Workflows



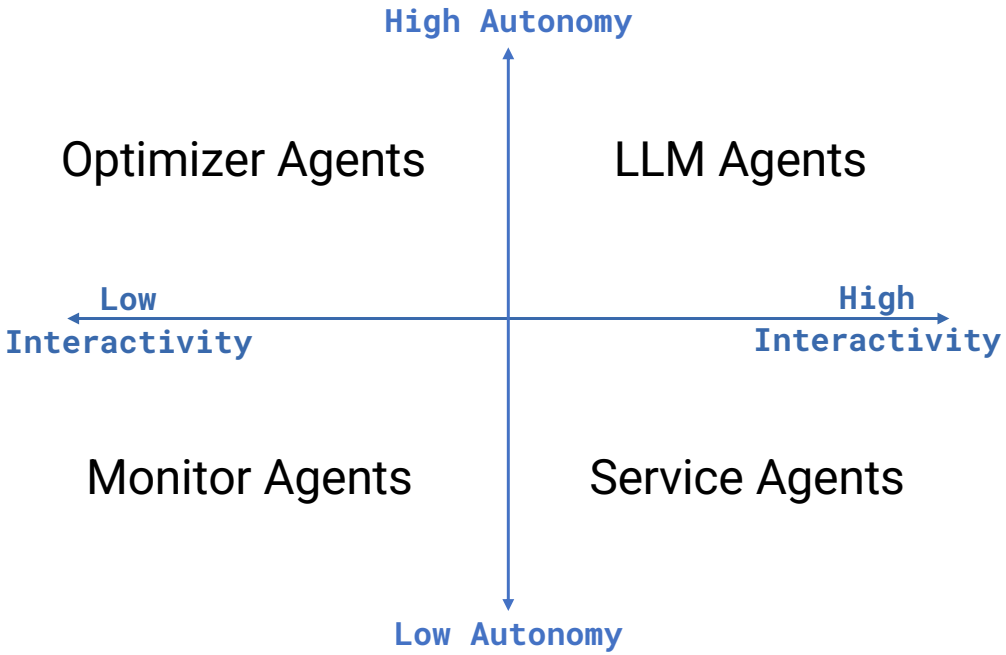
Integrating Experiments/Observations



Steering Workflows



Scientific Agents: Agent Behaviors



Other defining aspects:

- Persistent vs ephemeral
- General vs narrow purpose
- Embodiment

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

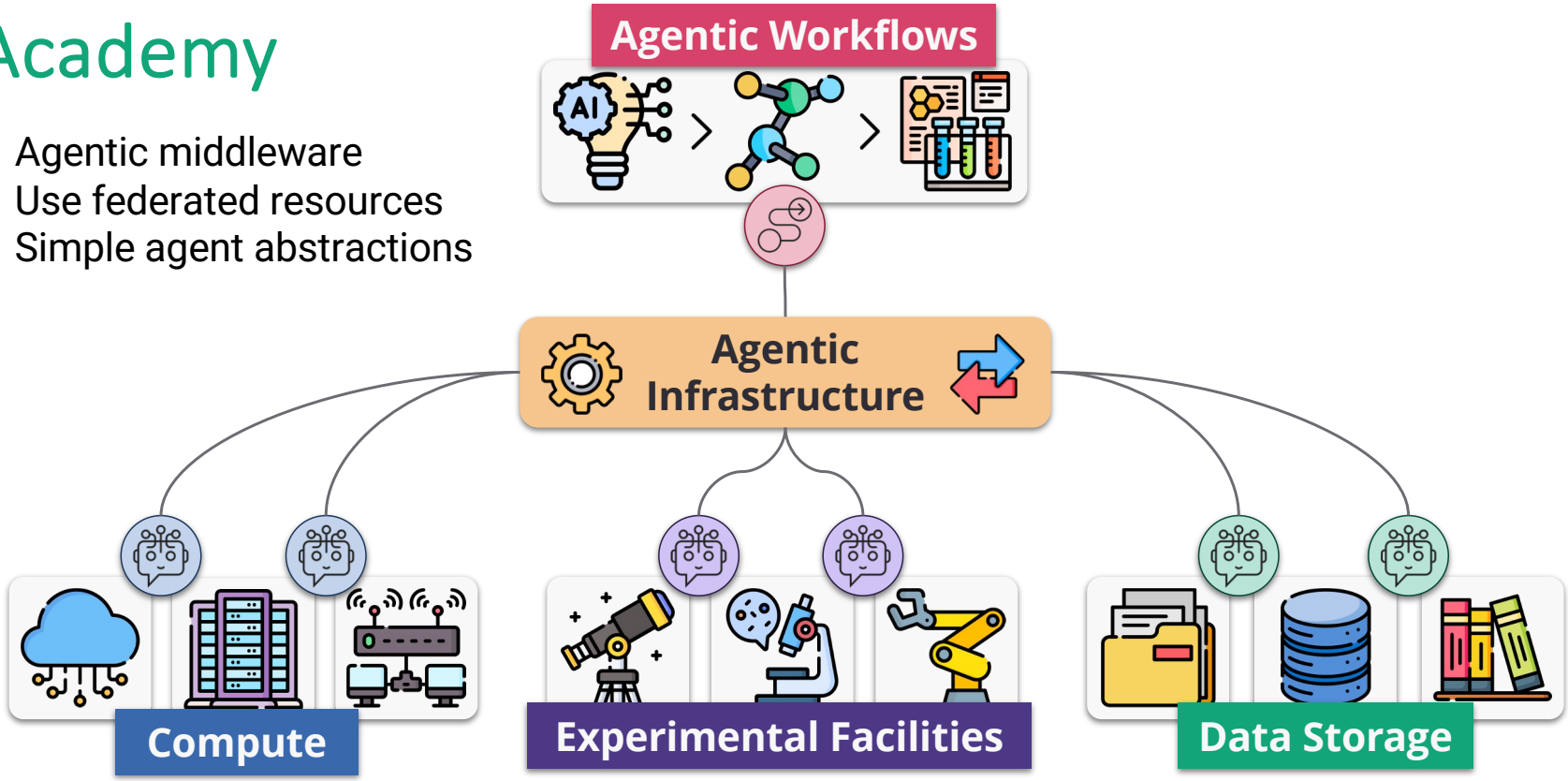
Academy primitives support the creation diverse agent types.

Additional Challenges

- Models beyond LLMs
  - Protein Language Models
  - Generative Chemistry Models
  - Surrogate Models
  - Control Models
- Diverse, Distributed Resources
- Differing messaging/communication requirements
- Varying availability
- Federated authentication
- Failures and Resilience

Academy

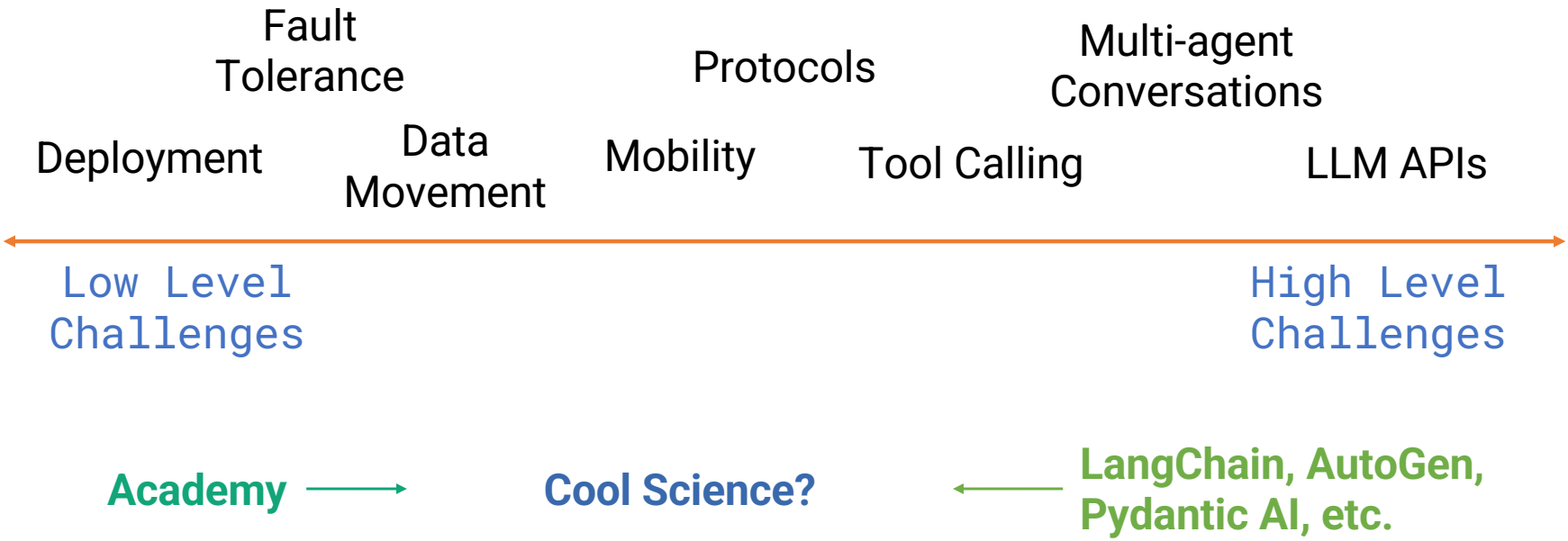
- Agentic middleware
- Use federated resources
- Simple agent abstractions



Agentic Middleware

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

Agentic Middleware: Scope & Challenges



Agentic Middleware: Using Research Infrastructure

Centralized

- Agents co-located (workstation, cloud)
- Research infrastructure available via APIs (REST, SDKs, MCP Servers, …)
- 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

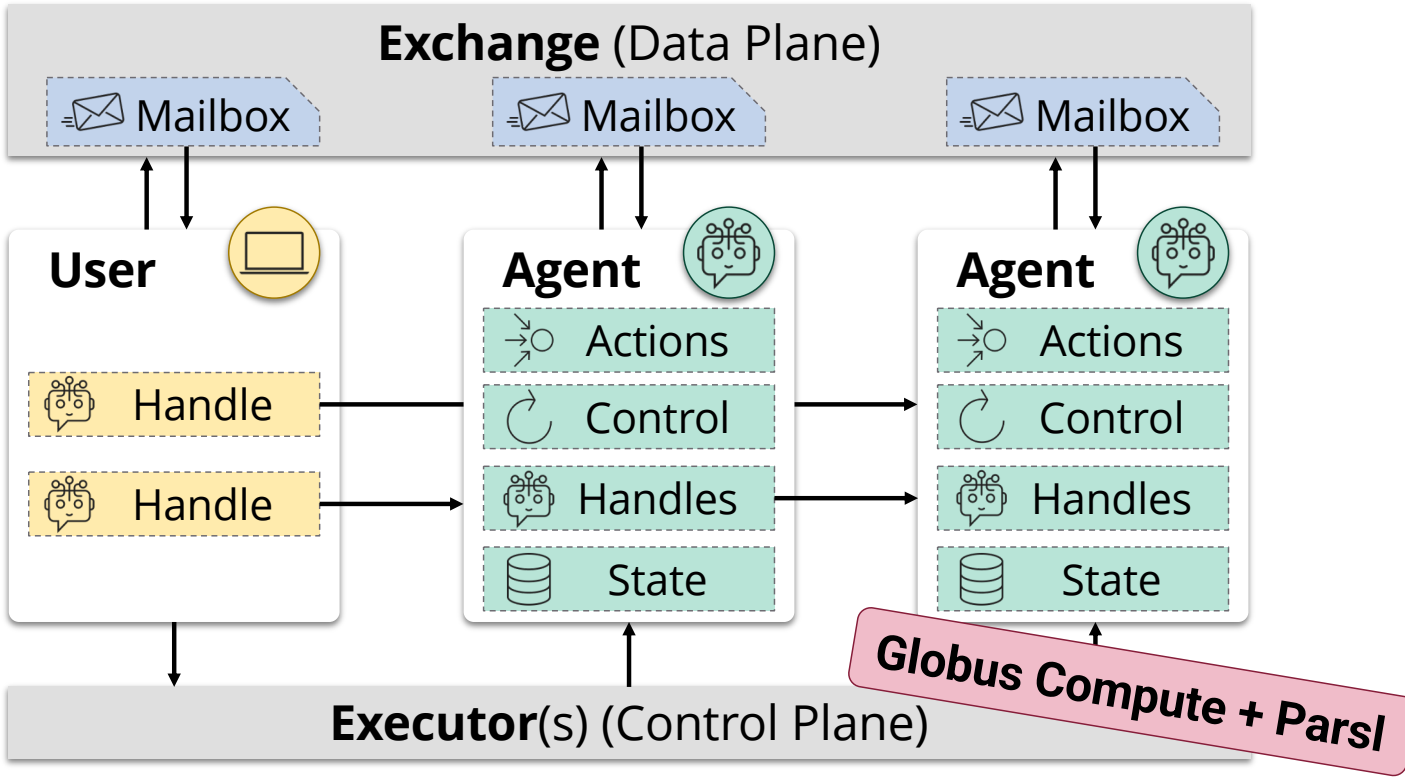
Academy

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

Focus 3: Coordinate async agent messaging

Focus 1: Program diverse agents and interactions

Focus 2: Deploy agents on federated resources



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

Exchange Design

Spatially Decoupled

- Messages are passed through a mediated communication mechanism
- Agents do not have to live in the same location
- Relies on *outbound* communication only

Temporally Decoupled

- Messages are stored in mailboxes
- Agents do not have to be online at the same time

Execution Options

- **Local Execution**
  - Threads or Processes
  - Communication can happen through in-memory data-structures
- **Cluster/HPC Execution**
  - Parsl: A pure python framework for managing workflows on HPC resources
  - Configurable executor/launcher for interfacing with scheduler (i.e. sbatch and srun)
  - Can communicate over high-speed interconnects
- **Remote Execution**
  - Globus Compute: A federated function as a service
  - Turns any compute resource into an endpoint that can run remote functions
  - Communicate through authenticated cloud-hosted exchange (exchange.academy-agents.org)

Writing Apps in Academy

Agents defined by a behavior

Clients & other agents can request actions

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

Instance of a behavior is state

Control loops for autonomous behavior

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

gce = GlobusComputeExecutor('<UUID>')

async with await Manager.from_exchange_factory(
        factory=LocalExchangeFactory(),
        executors=gce,
) as manager:
    behavior = Example()  # From the prior slide
    handle = await manager.launch(behavior)

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

    await handle.shutdown()
```

Launch agents via Globus Compute

Pass handles to other agents

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

Using Research Infrastructure through Tool Calling

```python
from academy.handle import Handle
from langchain_core.tools import tool

def make_sim_tool(handle: Handle[Simulator]):
    @tool
    async def compute_property(smiles: str) -> float:
        """Compute molecule property."""
        return await handle.compute_property(smiles)
    return compute_property

tool = make_tool(agent_handle)
print(tool.args_schema.model_json_schema())
```

Turn agent handles into LLM framework “tools”

Comparison to Alternatives

Tool Servers (MCP)
Rely on externally reachable endpoints that are blocked by facility policies. Requires user to manage services, infrastructure, and VPNs

Func-as-a-Service (Globus Compute)
Easier remote execution (no VPN, infrastructure management) but tools must be stateless, short-running tasks

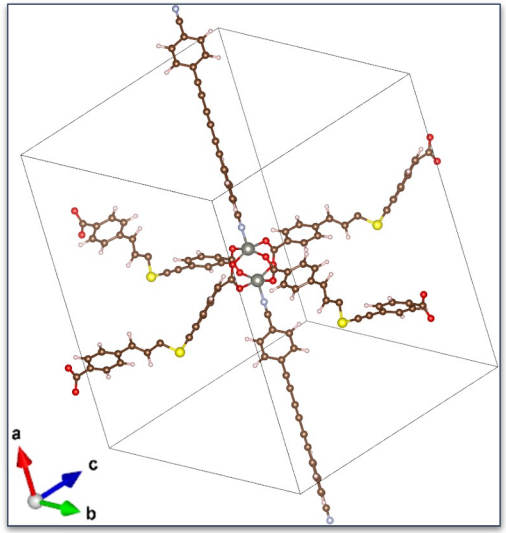
Case Studies

Use Case: MOF Discovery

Metal Organic Frameworks (MOF)

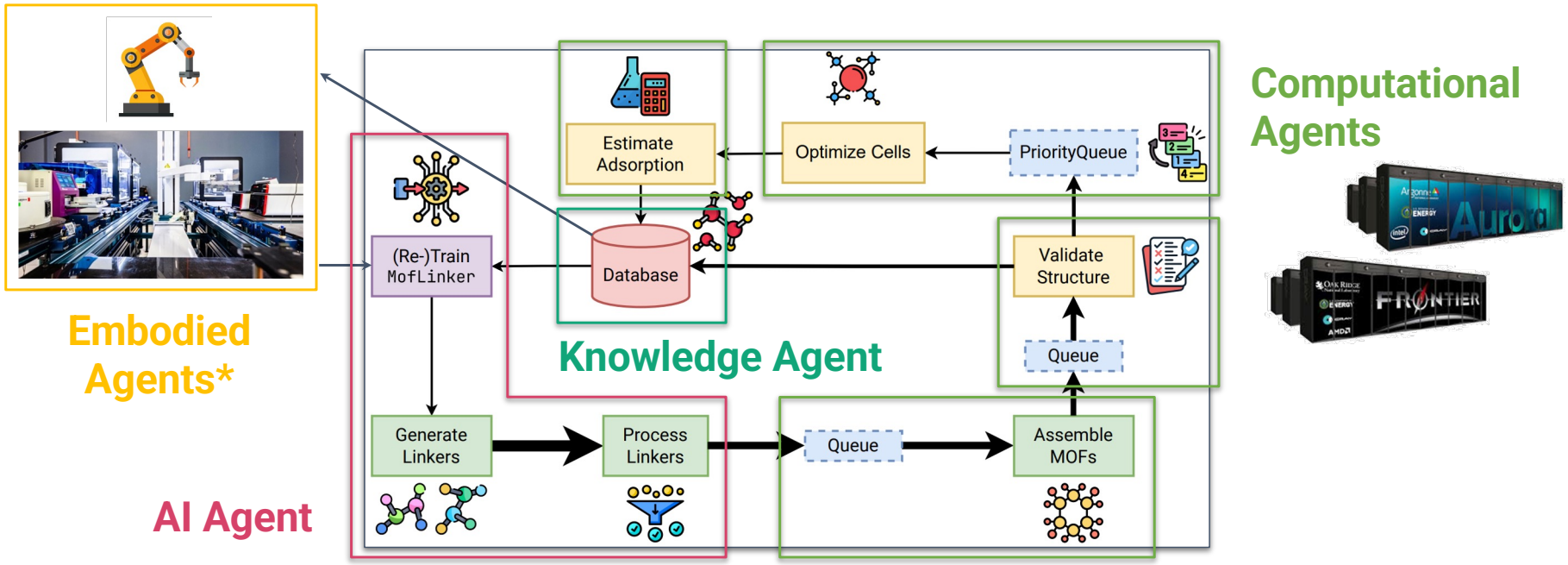
- ➔ Composed of organic molecules (ligands) and inorganic metals (nodes)
- ➔ The sponges of materials science!
- ➔ Porous structures that adsorb and store gases
- ➔ Topologies can be optimized for targeted gas storage → **Carbon Capture**

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



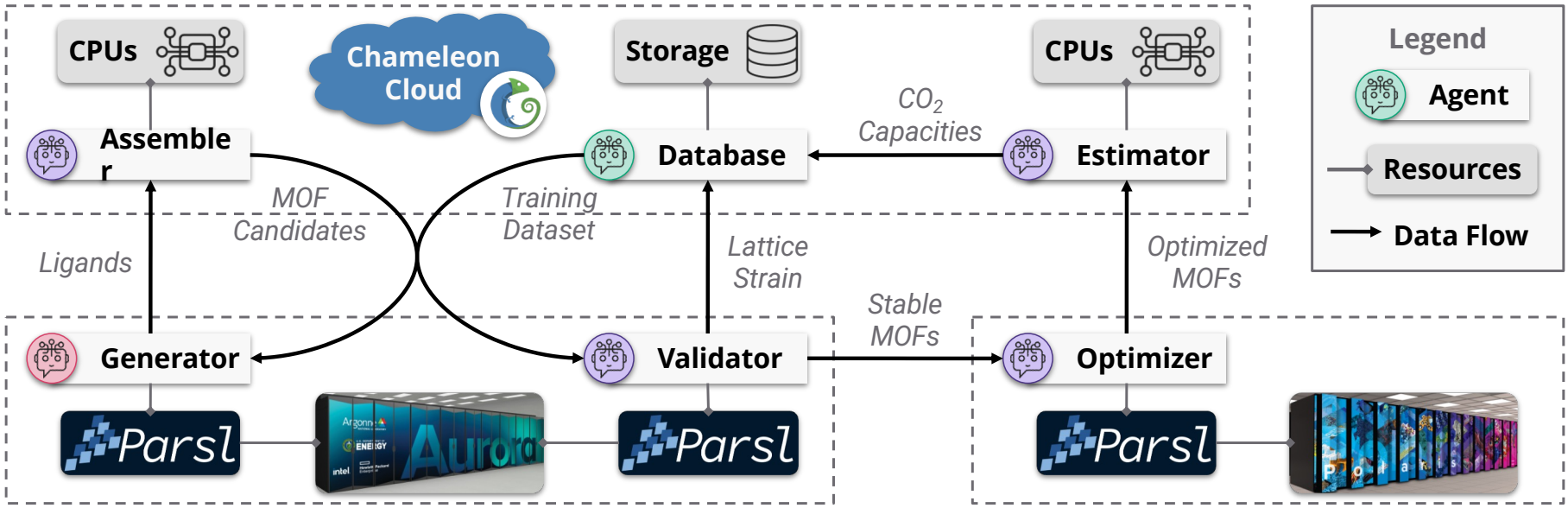
Intractable search space of ligand, node, & geometry combinations

MOFA: Online learning + GenAI + Simulation

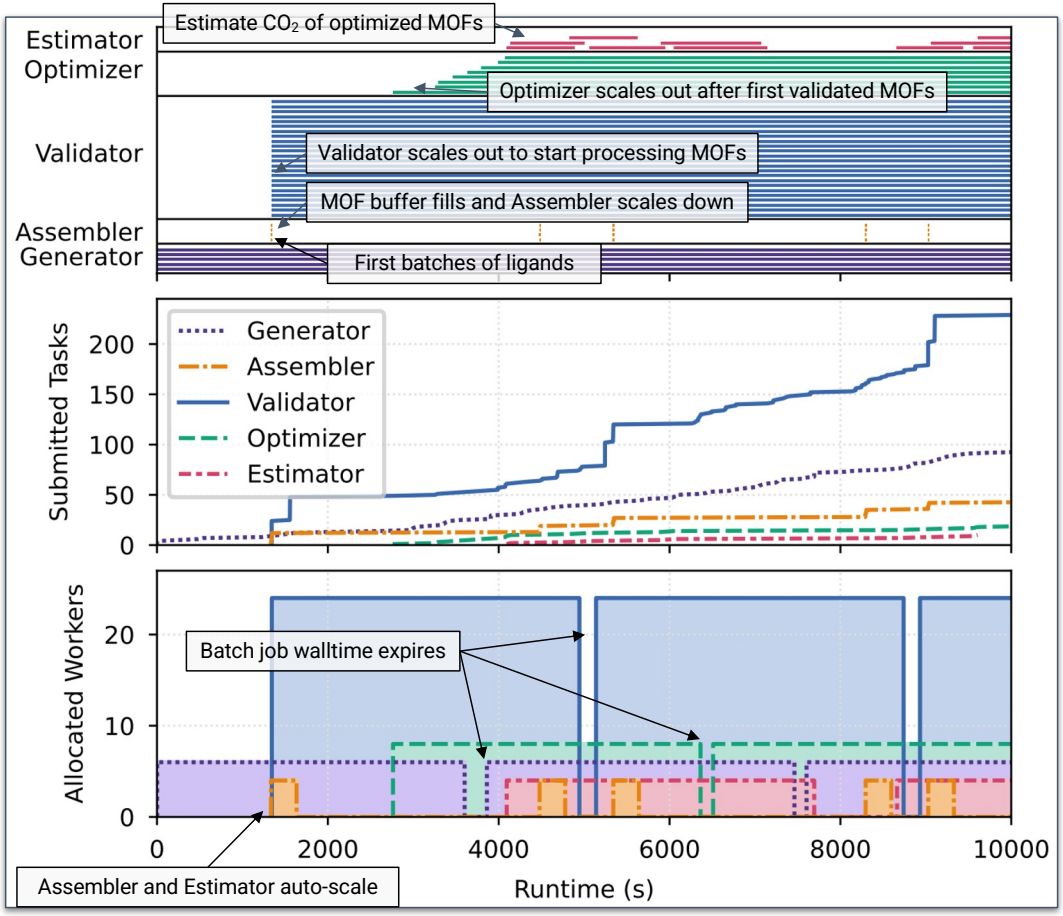


Yan et al., "**MOFA: Discovering Materials for Carbon Capture with a GenAI- and Simulation-Based Workflow**" (Under Review)

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:
`alokvk2@uchicago.edu`

Please Star the Repo and Join the Slack!

- github.com/academy-agents
- academy-agents.org

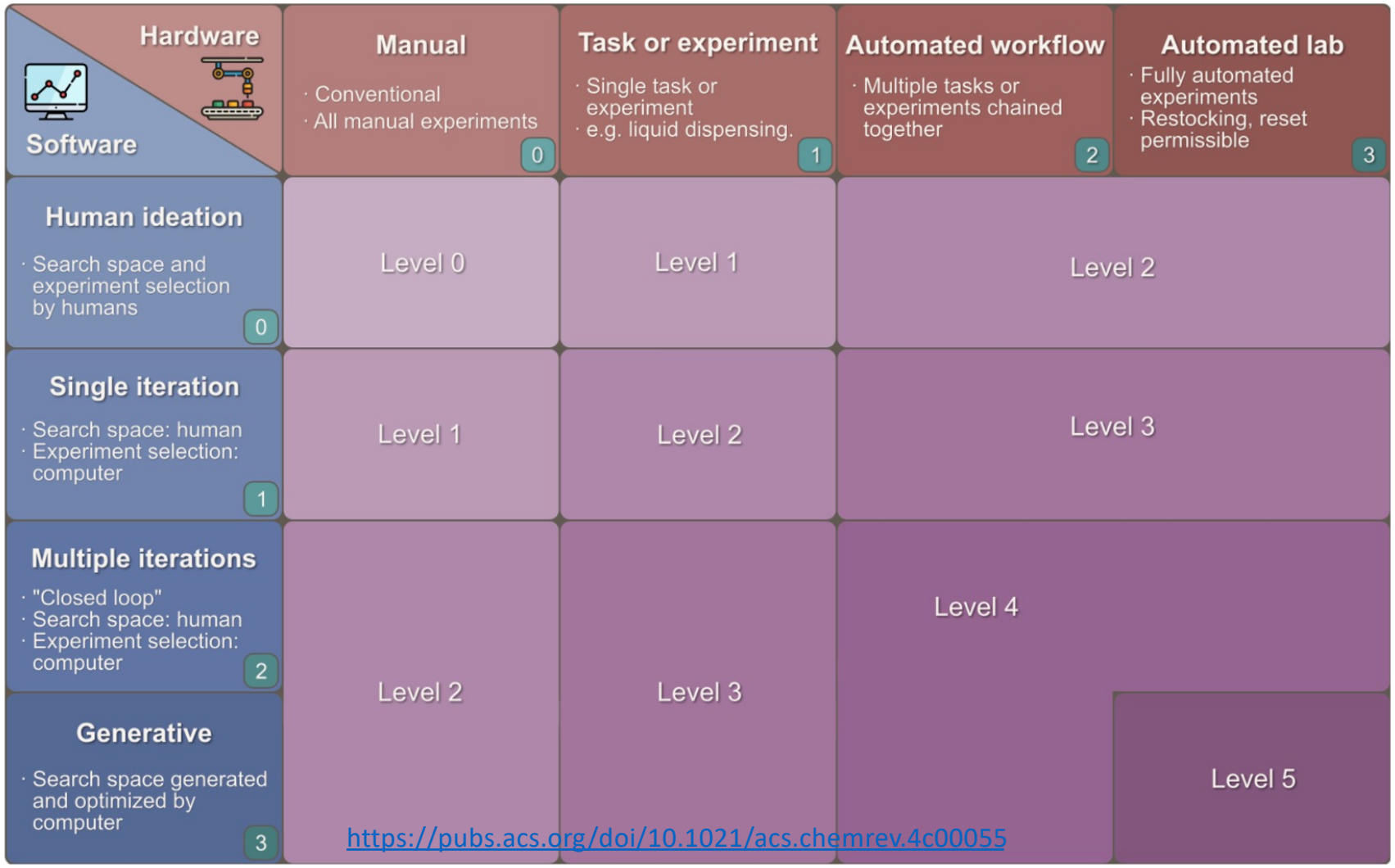
⭐ Academy on GitHub!

Self-driving laboratories

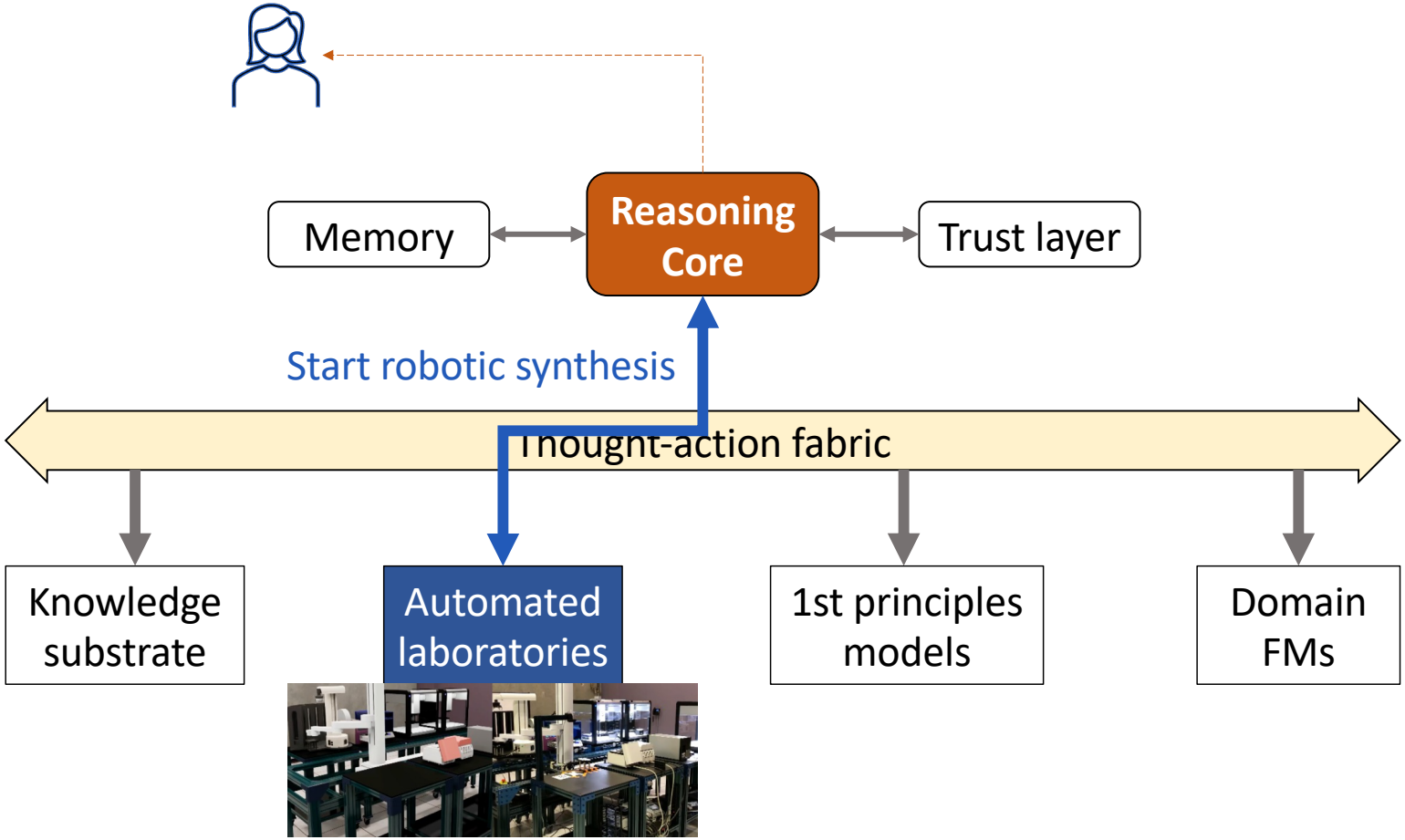
Self-driving laboratories

- "A self-driving lab integrates automation, computation, and artificial intelligence to iteratively design, execute, and analyze experiments without human supervision"
  - Automation can enable greater use of scarce/expensive resources and also enhance reliability and reproducibility
  - Computation and artificial intelligence can enable more efficient use of scarce/expensive resources
  - Democratization due to reduced resource and skill requirements

- An "SDL" is a form of Scientific Discovery Platform
- Here we focus on the mechanics of accessing experimental apparatus


Hardware
Software
Manual
· Conventional
· All manual experiments
0
Task or experiment
· Single task or experiment
· e.g. liquid dispensing.
1
Automated workflow
· Multiple tasks or experiments chained together
2
Automated lab
· Fully automated experiments
· Restocking, reset permissible
3
Human ideation
· Search space and experiment selection by humans
0
Level 0
Level 1
Level 2
Single iteration
· Search space: human
· Experiment selection: computer
1
Level 1
Level 2
Level 3
Multiple iterations
· "Closed loop"
· Search space: human
· Experiment selection: computer
2
Level 4
Generative
· Search space generated and optimized by computer
3
Level 2
Level 3
Level 5
https://pubs.acs.org/doi/10.1021/acs.chemrev.4c00055

AI-native Scientific Discovery Platform



Instruments as Scientific Discovery Platform nodes

- A reasoning model structured requests to orchestrate actions across simulations, databases, and instruments:
  - **Simulation tools**: For predictive or exploratory modeling
  - **Laboratory apparatus**: For physical validation or synthesis
  - **Data stores & knowledge graphs**: For retrieval and hypothesis grounding

- All communicate via a shared **context and schema**, enabling the reasoning model to:
  - Generate hypotheses
  - Choose the best execution resource (e.g., simulation vs. lab)
  - Interpret results and refine hypotheses

Challenges specific to physical equipment

Interface & Control

- Labs expose diverse, human-oriented control layers (GUIs, scripts)
- Reasoning model discovers instrument capabilities via APIs or MCP schemas
- Actions have real-world latency and uncertain observability

Safety and Robustness

- Avoid unsafe states (thermal, chemical, mechanical)
- Handle ambiguous or missing sensor feedback
- Detect and recover from hardware or reasoning errors

Reasoning & Planning

- Translate high-level scientific goals into valid, safe instrument commands
- Coordinate lab actions with simulations and data queries
- Anticipate time-coupled tasks and downstream data dependencies

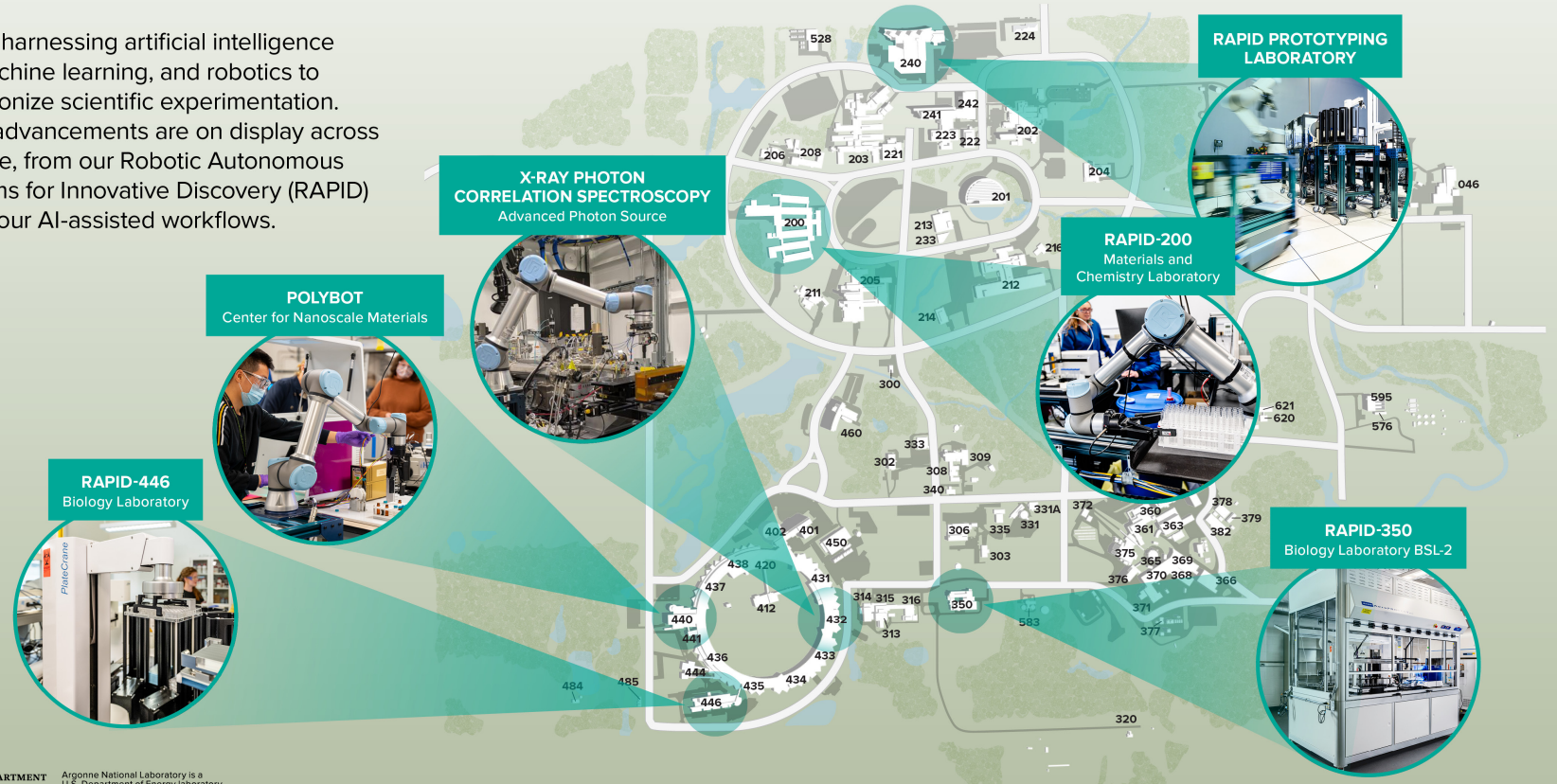
Knowledge and Feedback

- Fuse experimental data with simulation predictions
- Maintain provenance across heterogeneous sources
- Adapt future reasoning based on performance and uncertainty metrics

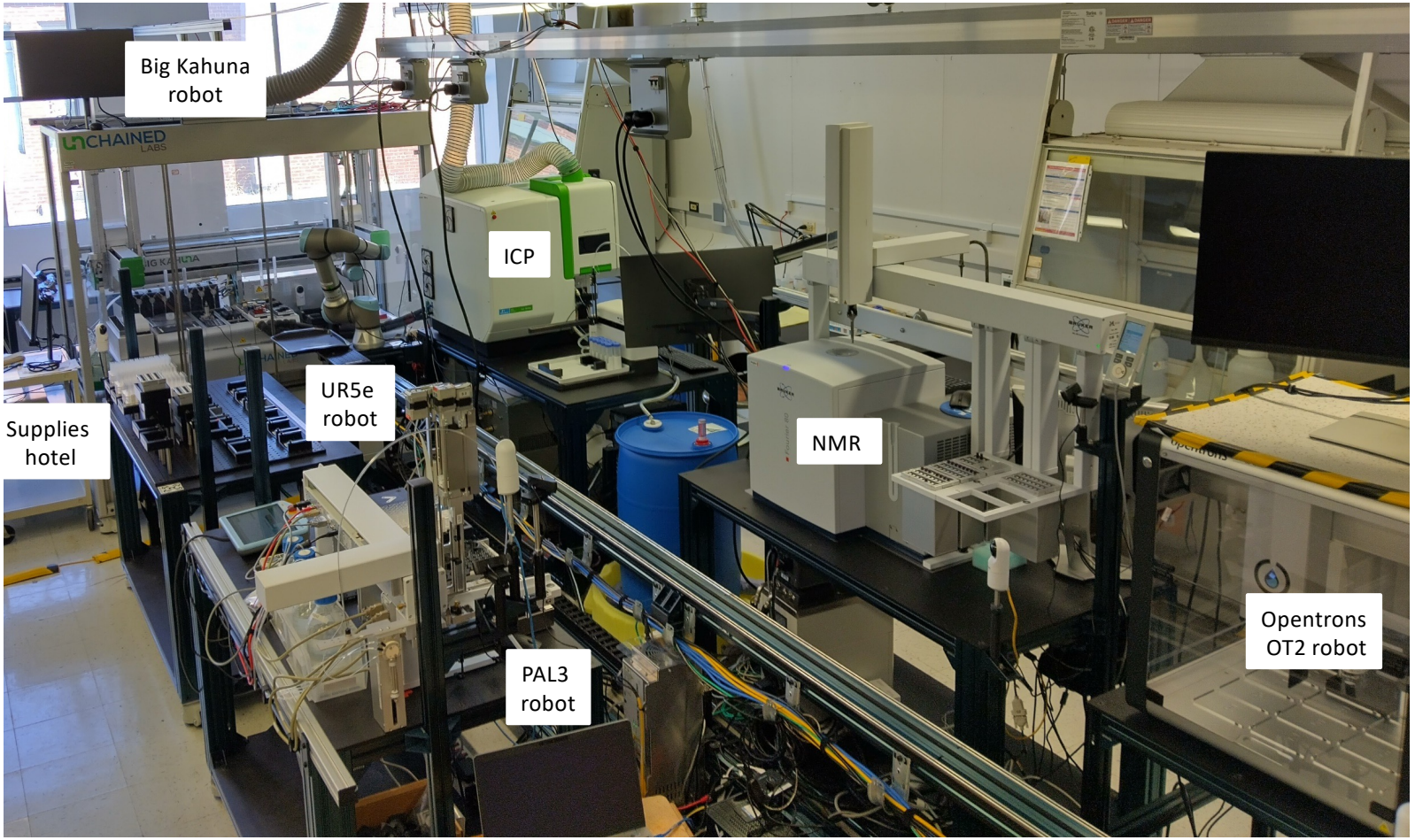
AUTONOMOUS LABORATORIES
ACCELERATING SCIENTIFIC DISCOVERY

Argonne NATIONAL LABORATORY

We are harnessing artificial intelligence (AI), machine learning, and robotics to revolutionize scientific experimentation. These advancements are on display across Argonne, from our Robotic Autonomous Platforms for Innovative Discovery (RAPID) labs to our AI-assisted workflows.

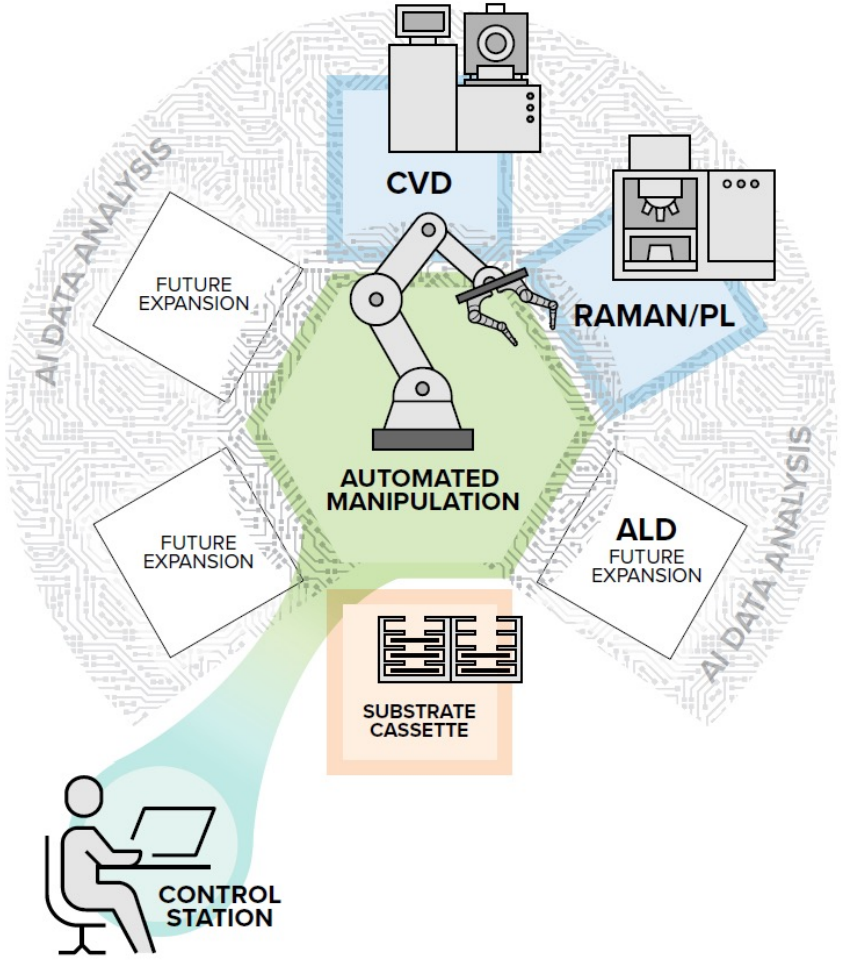


RAPID-200: “Benchtop” experiments in air

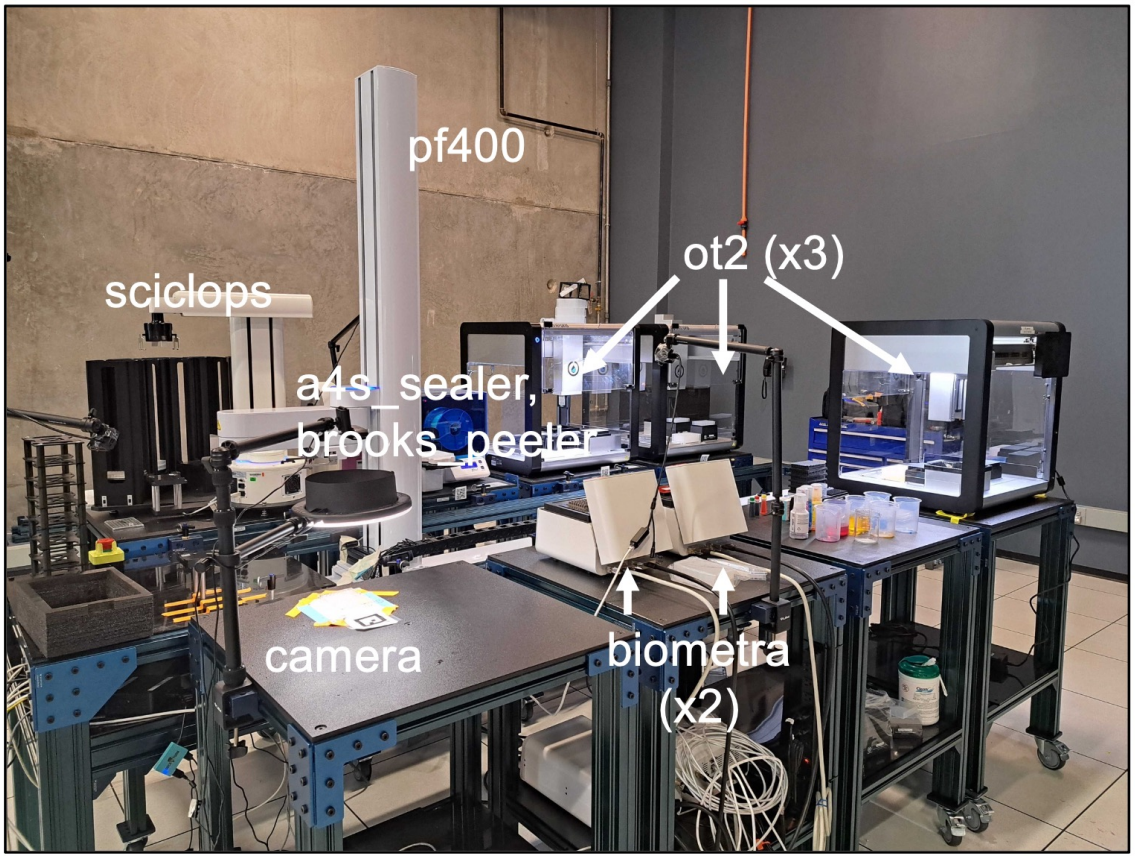


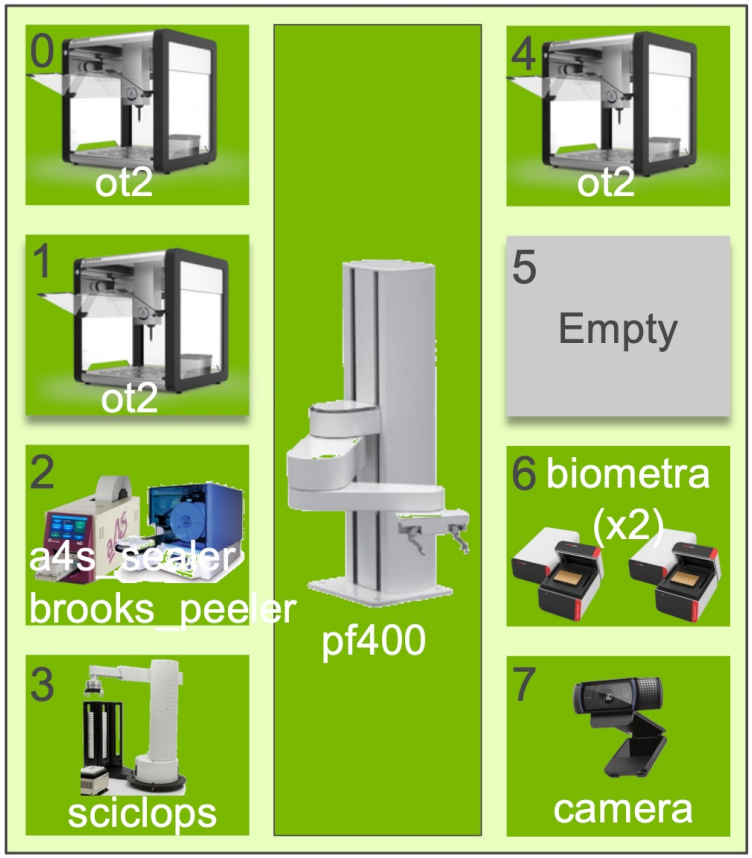
RAPID-200: Experiments in glovebox for air sensitive materials

RAPID-200: Experiments in vacuum for solid-state growth and characterization



Modular design of robotic instruments

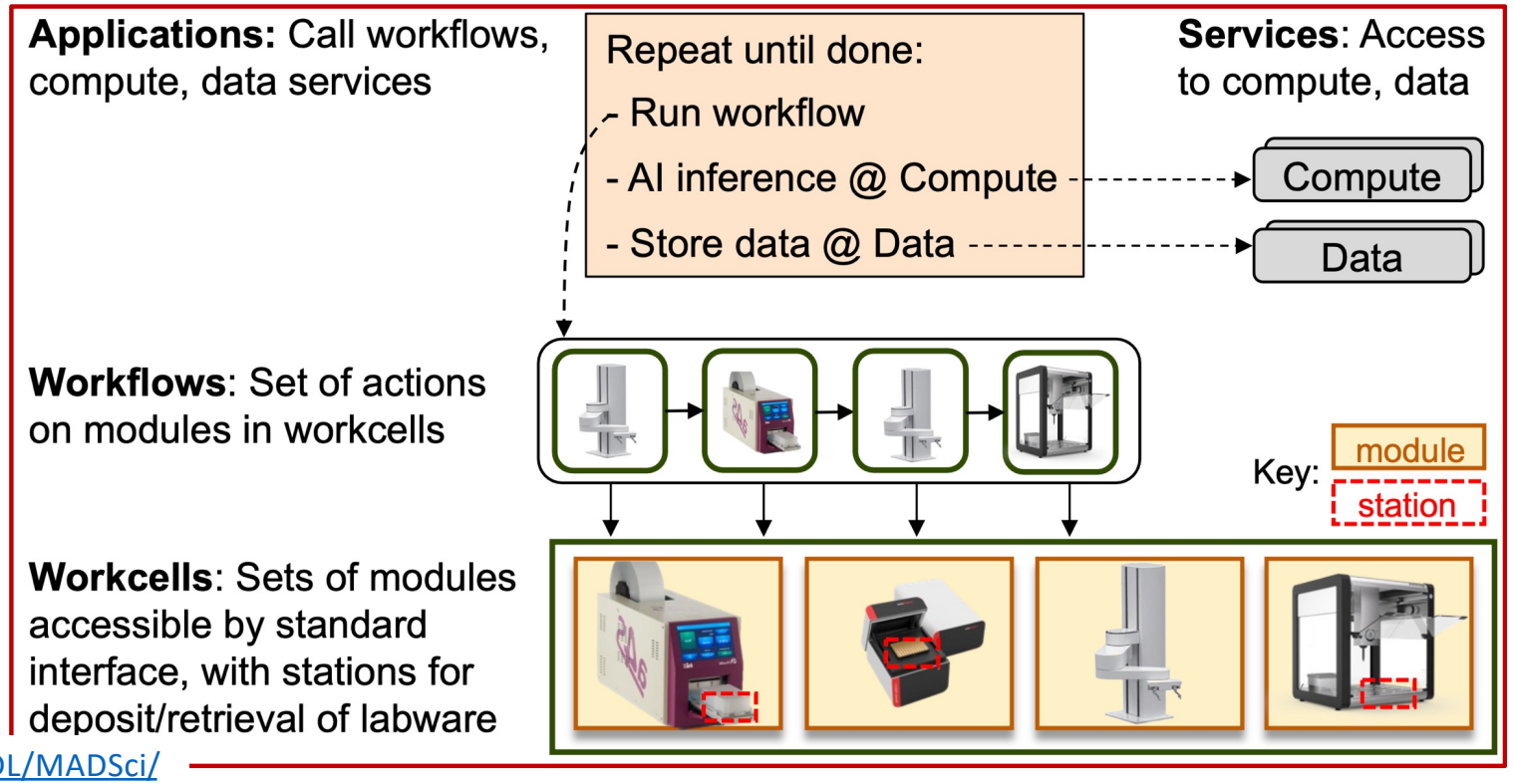




https://arxiv.org/abs/2308.09793

Modular design of robotic instruments

Hierarchical API with modularity and abstraction;
domain protocols compiled to robotic commands

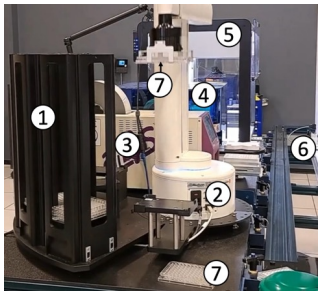


arxiv.org/abs/2308.09793

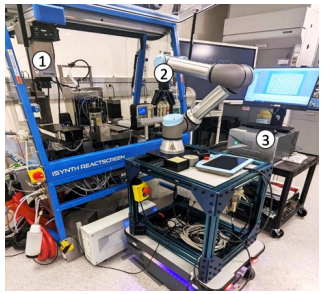
Modular Autonomous Discovery for Science (MADSci) Framework

https://github.com/AD-SDL/MADSci/

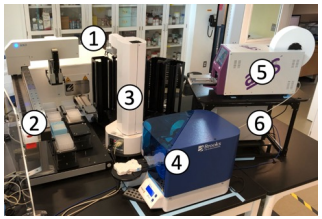
Modular design of robotic instruments



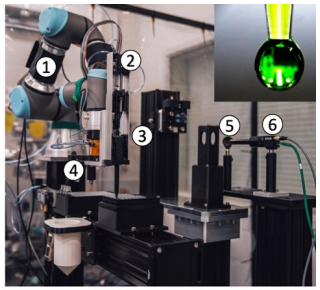
RPL workcell



CNM workcell



BIO workcell



8ID workcell

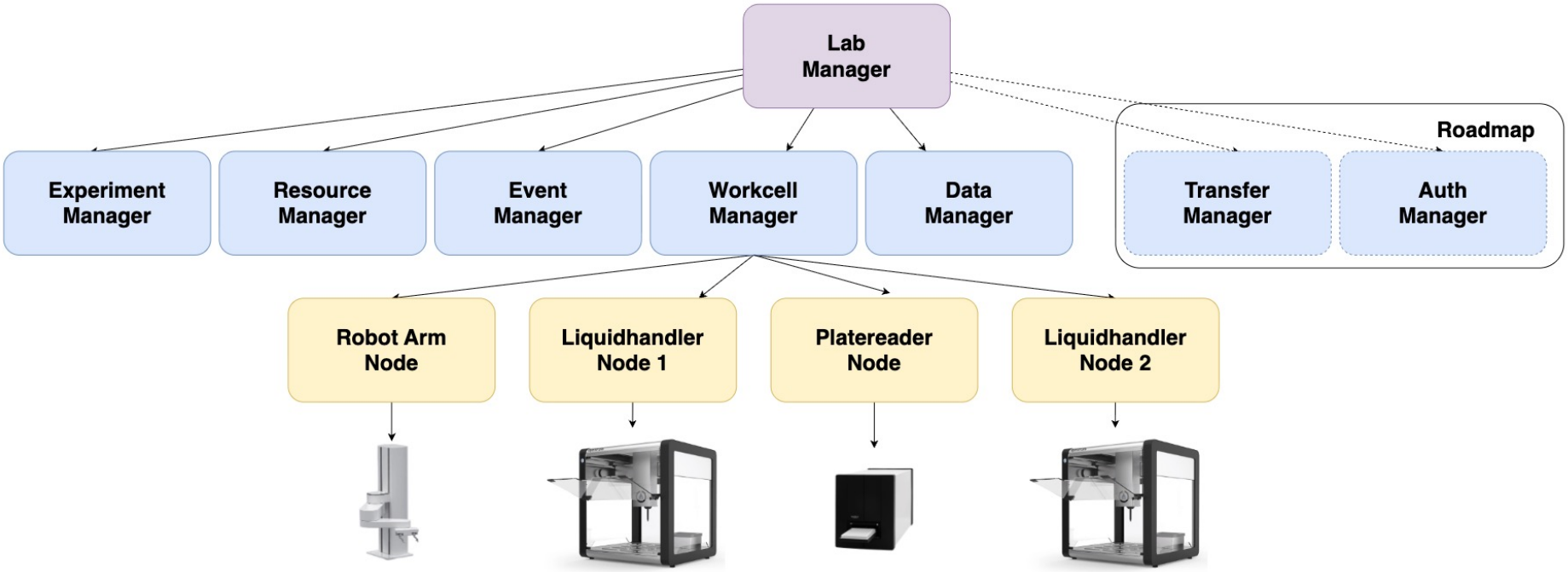
Same module, different applications

Same application, different workcells

Same action (transfer), different modules

arxiv.org/abs/2308.09793

		Application / Workcell					
		Color picker/RPL	PCR/RPL	Growth assay/RPL	Growth assay/BIO	Electrochromic/CNM	Pendant/8ID
Class	**Module**						
Synthesis	ot2	✓	✓	✓			
	solo				✓		
	chemspeed					✓	
Plate prep	a4s_sealer		✓	✓	✓		
	brooks_peeler		✓	✓	✓		
Heat	biometra		✓				
	liconic				✓		
Measure	camera	✓	✓				
	hidex			(✓)	✓		
	tecan					✓	
Manipulate	platecrane				✓		
	pf400	✓	✓	✓			
	ur					✓	✓
	sciclops	✓	✓				
Mobility	mir	(✓)	(✓)			(✓)	
Compute, data	Globus Flows	✓	✓	✓	✓	✓	(✓)
	Globus Search	✓	✓	✓	✓	(✓)	(✓)



https://github.com/AD-SDL/MADSci/

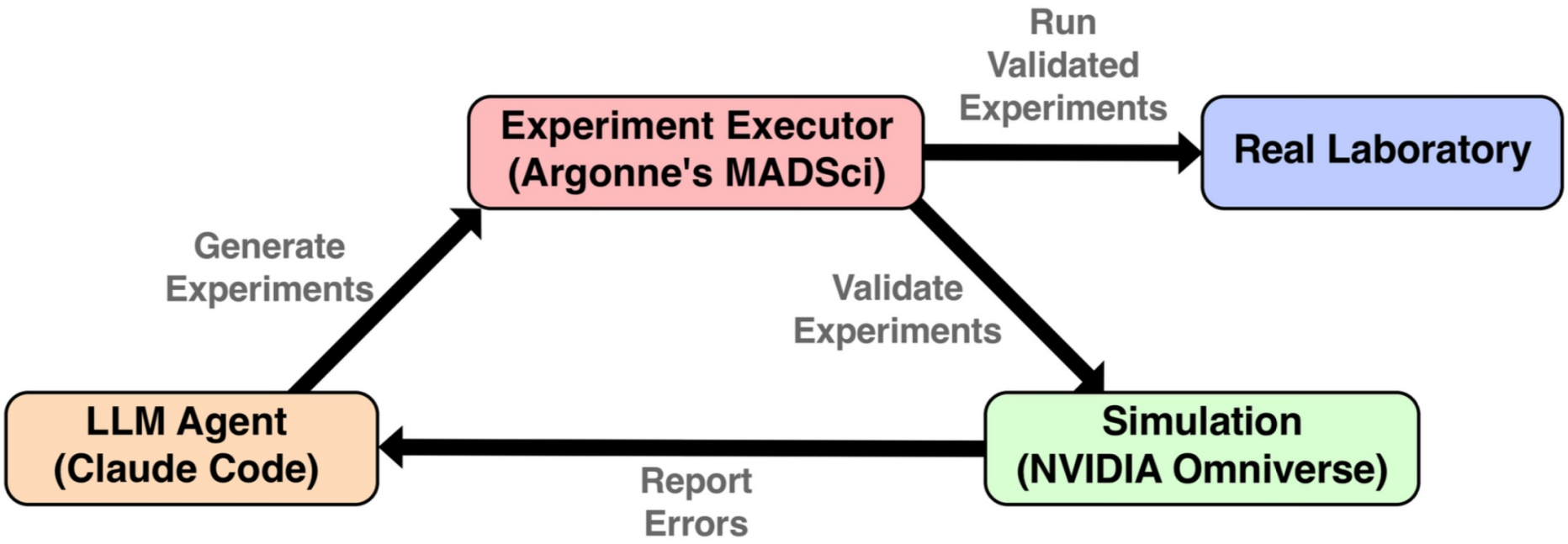
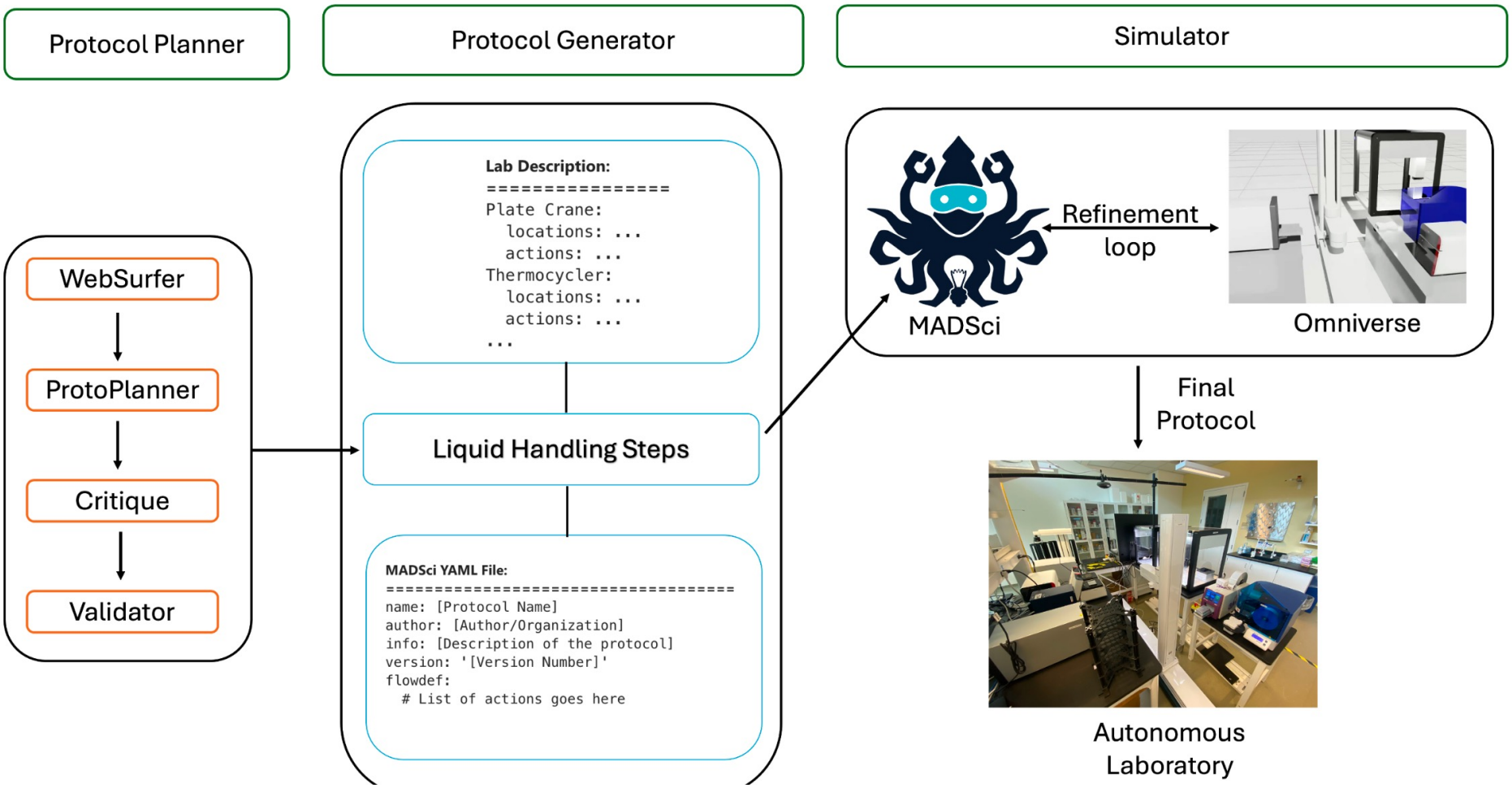


Fig. 1. Overview of simulation integration into agentic workflows. Agents such as Claude Code generate experimental protocols and format them into robot-level commands using large language models. The MADSci orchestration framework, developed at Argonne National Laboratory, sends these commands to our robotic simulation system for analysis before physical execution. If errors are detected, detailed feedback is returned to the agent for protocol improvement. Once protocols pass simulation analysis, the same orchestration software can execute the refined protocols directly on real laboratory equipment without modification.

Rory Butler



Brian Hsu, Priyanka Setty, Rory Butler, et al.

PCR Workflow Sequence Overview

This document describes the high-level workflow phases for automated PCR processing, based on the provided YAML workflow file.

Steps

- Execute the liquid handling protocol on the OT-2 robot (file `payload.ot2_protocol`)
- Transfer the destination plate from OT-2 to sealer
- Seal the plate using the sealer
- Transfer the sealed plate from sealer to thermocycler
- Run the thermocycling program (program #5)
- Transfer the plate from thermocycler to peeler
- Peel the plate using the peeler
- Transfer the plate from peeler to hidex
- Run the assay protocol ("PCR_Final_Results")
- Open the plate reader lid after analysis completion

```yaml
name: PCR Amplification Workflow
author: Autoprotocol
info: Automated PCR workflow with OT-2 liquid handling, sealing, thermocycling,
peeling, and Hidex assay
version: '1.0'

flowdef:

- action: run_protocol
  name: Execute OT-2 liquid handling protocol
  module: ot2bioalpha
  files:
    protocol: payload.ot2_protocol

- action: transfer
  name: Transfer plate from OT-2 (wide) to exchange (wide)
  module: biopf400
  args:
    source: ot2bioalpha_deck1_wide
    source_approach: safe_path_ot2bioalpha
    source_plate_rotation: wide
    target: exchange_deck_high_wide
    target_approach: safe_path_exchange
    target_plate_rotation: wide

- action: transfer
  name: Transfer plate from exchange (narrow) to sealer (narrow)
  module: biopf400
  args:
    source: exchange_deck_high_narrow
    source_approach: safe_path_exchange
    source_plate_rotation: narrow
    target: sealer_nest
    target_approach: safe_path_sealer
    target_plate_rotation: narrow

- action: seal
  name: Seal the reaction plate
  module: bio_sealer
  args: {}

- action: open
  name: Open thermocycler lid
  module: bio_biometra3_96
  args: {}

- action: transfer
  name: Transfer sealed plate from sealer (narrow) to exchange (narrow)
  module: biopf400
  args:
    source: sealer_nest
    source_approach: safe_path_sealer
    source_plate_rotation: narrow
    target: exchange_deck_high_narrow
    target_approach: safe_path_exchange
    target_plate_rotation: narrow

- action: transfer
  name: Transfer sealed plate from exchange (wide) to thermocycler (wide)
  module: biopf400
  args:
    source: exchange_deck_high_wide
    source_approach: safe_path_exchange
    source_plate_rotation: wide
    target: bio_biometra3_nest
    target_approach: safe_path_biometra3
    target_plate_rotation: wide

- action: close
  name: Close thermocycler lid
  module: bio_biometra3_96
  args: {}
```

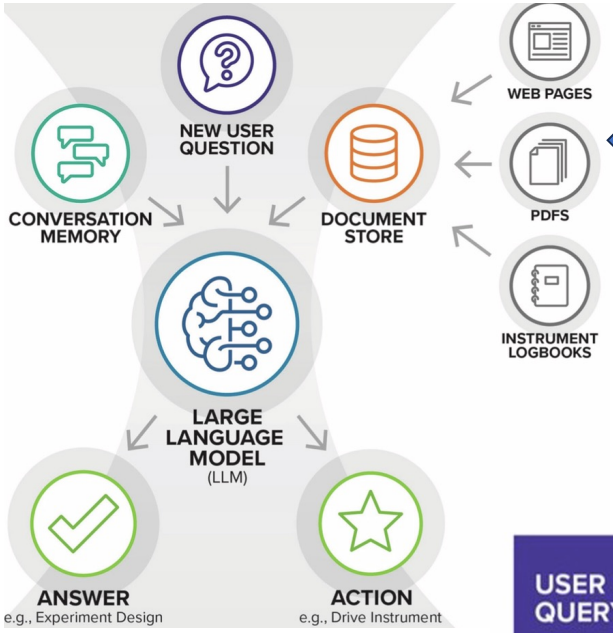
```yaml
- action: run_program
  name: Run thermocycling program #5
  module: bio_biometra3_96
  args:
    program_number: 5

- action: open
  name: Open thermocycler lid to unload
  module: bio_biometra3_96
  args: {}

- action: transfer
  name: Transfer plate from thermocycler (wide) to exchange (wide)
  module: biopf400
  args:
    source: bio_biometra3_nest
    source_approach: safe_path_biometra3
    source_plate_rotation: wide
    target: exchange_deck_high_wide
    target_approach: safe_path_exchange
    target_plate_rotation: wide

- action: transfer
  name: Transfer plate from exchange (narrow) to peeler (narrow)
  module: biopf400
  args:
    source: exchange_deck_high_narrow
    source_approach: safe_path_exchange
    source_plate_rotation: narrow
    target: peeler_nest
    target_approach: safe_path_peeler
    target_plate_rotation: narrow

- action: peel
  name: Peel the plate seal
  module: bio_peeler
  args: {}
```

```yaml
- action: open
  name: Open Hidex plate reader lid
  module: hidex_geraldine
  args: {}

- action: transfer
  name: Transfer plate from peeler (narrow) to Hidex (narrow)
  module: biopf400
  args:
    source: peeler_nest
    source_approach: safe_path_peeler
    source_plate_rotation: narrow
    target: hidex_geraldine_high_nest
    target_approach: safe_path_hidex
    target_plate_rotation: narrow

- action: close
  name: Close Hidex lid
  module: hidex_geraldine
  args: {}

- action: run_assay
  name: Run assay: PCR_Final_Results
  module: hidex_geraldine
  args:
    assay_name: "PCR_Final_Results"

- action: open
  name: Open Hidex lid after analysis
  module: hidex_geraldine
  args: {}
```

Task	Model	Cycles	Failure modes	Outcome
PCR	Mistral	5 (maxed out)	Volume errors; step formatting	Failed
	Claude Opus 4	1	Extra thermocycling step (4 vs 3)	Success
	GPT-4o	1	Missing thermocycling instructions	Success
Cell Painting	Mistral	5 (maxed out)	No source/destination mapping; invalid logic	Failed
	Claude Opus 4	3	Volumes wrong (2/51); vague repeats (11); missing pipette type (15)	Partial success
	GPT-4o	1	Minor formatting issues	Success

Brian Hsu, Priyanka Setty, Rory Butler, et al.

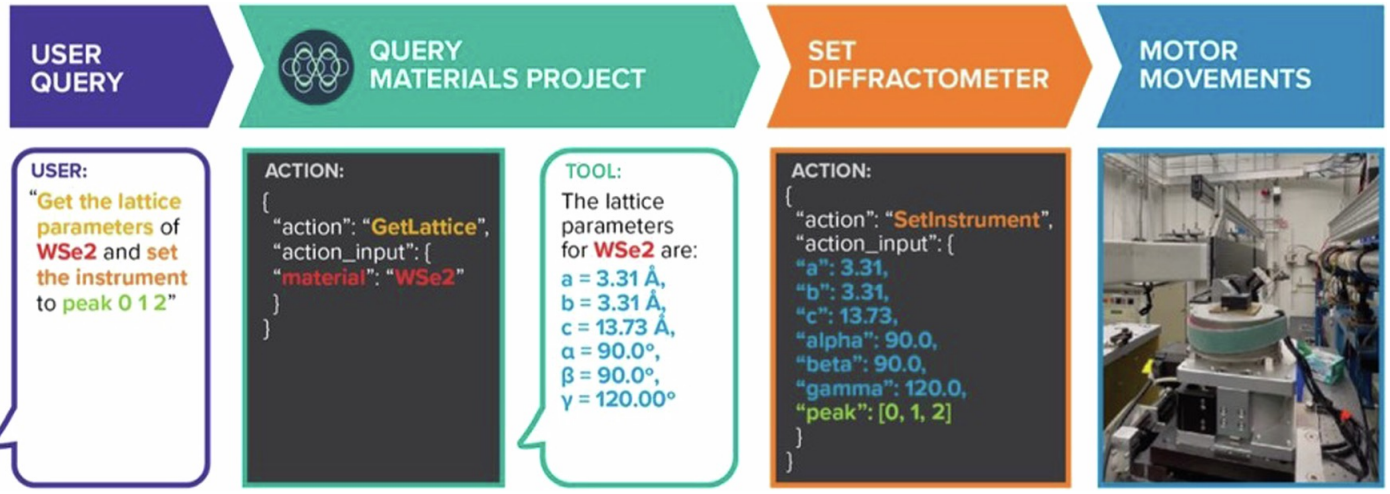
Further experiments with LLMs



CALMS: Context-Aware Language Model for Science

Execution of a robot motor move based on a user query to CALMS

CALMS parses the user query to extract the material and Bragg peak, queries Materials Project for the lattice constants, and then calculates the position of and moves the beamline diffractometer to the requested peak



Prince, Chan, Vriza, Zhou, Sastry, Luo, Dearing, Harder, Vasudevan, & Cherukara, *Opportunities for retrieval and tool augmented large language models in scientific facilities* Npj Comp. Materials, 2024

https://doi.org/10.1038/s41524-024-01423-2