

Agentic AI

Mark Zhao

CSCI 5942

Fall 2026

College of Engineering & Applied Science
UNIVERSITY OF COLORADO **BOULDER**

Announcements

- A6 due today, quiz in class
- A7 release, due 11/17

Where are we?

- Inference-time compute (CoT, self-consistency, routing)
  - *How do we use more tokens / LLM calls to answer a question?*
- Continuous Batching
  - *How do we increase concurrent decodes?*
- PagedAttention
  - *How do we effectively utilize GPU memory for KV cache?*
- Chunked Prefill, PD Disaggregation
  - *How do we reduce interference between prefill and decode?*
- Prefix caching, KV-centric serving
  - *How do we reuse precomputed KV cache between requests?*

Today: Agentic AI

- What is an ”agent”?
  - autonomy (intelligence)
  - tool use (external environment beyond the model)
  - specialization (customize for different tasks)
  - non-determinism

What is an “agent”?

- Some intelligent system that interacts with some environment
  - Physical world: robotics, autonomous cars
  - Digital world: Go, StarCraft, …
  - Human world: chatbot

- How we define “intelligent” and “environment” -> how we define “agent”

What is an LLM agent?

context c_t
Task, tool specs, (action, observation) pairs

→

LLM π(a_t | c_t)
One inference request

→

action a_t
Tool call, code, thought, message, …

→

environment
Runs a_t, returns observation o_t

Loop: c_t+1 = c_t + a_t + o_t until finish or budget is up. Each new iteration is a new request with more context in the prompt.

Generative Agents: Interactive Simulacra of Human Behavior

Joon Sung Park
Stanford University
Stanford, USA
joonspk@stanford.edu

Joseph C. O'Brien
Stanford University
Stanford, USA
jobrien3@stanford.edu

Carrie J. Cai
Google Research
Mountain View, CA, USA
cjcai@google.com

Meredith Ringel Morris
Google DeepMind
Seattle, WA, USA
merrie@google.com

Percy Liang
Stanford University
Stanford, USA
pliang@cs.stanford.edu

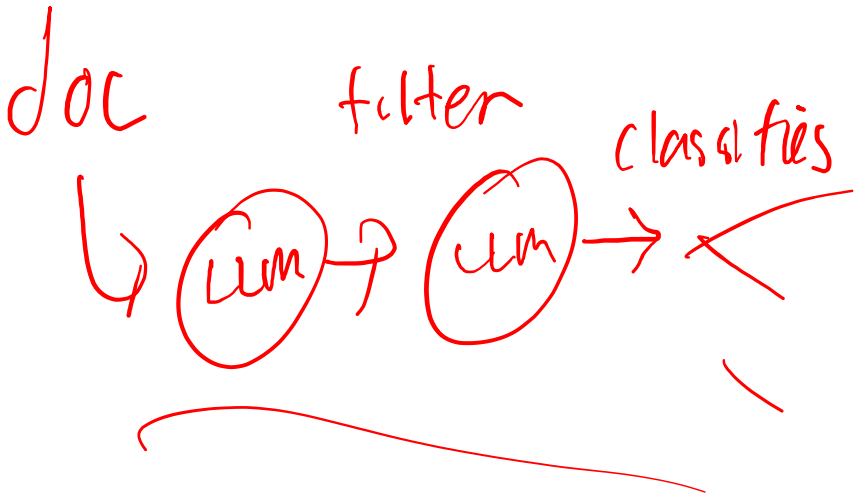
Michael S. Bernstein
Stanford University
Stanford, USA
msb@cs.stanford.edu



Workflows/Chains vs Agents?

Chain

- fixed-function calls to LLMs



Agent

- Dynamically decide next steps based on context
- more general purpose
- Planning, Tools, Memory

A history of agents

- ELIZA (1966): Text agent via pattern matching and substitution

```
  Welcome to
                EEEEEE  LL      IIII   ZZZZZZ   AAAAA
                EE      LL       II       ZZ   AA   AA
                EEEEE   LL       II     ZZZ    AAAAAAA
                EE      LL       II    ZZ      AA   AA
                EEEEEE  LLLLLL  IIII ZZZZZZ    AA   AA

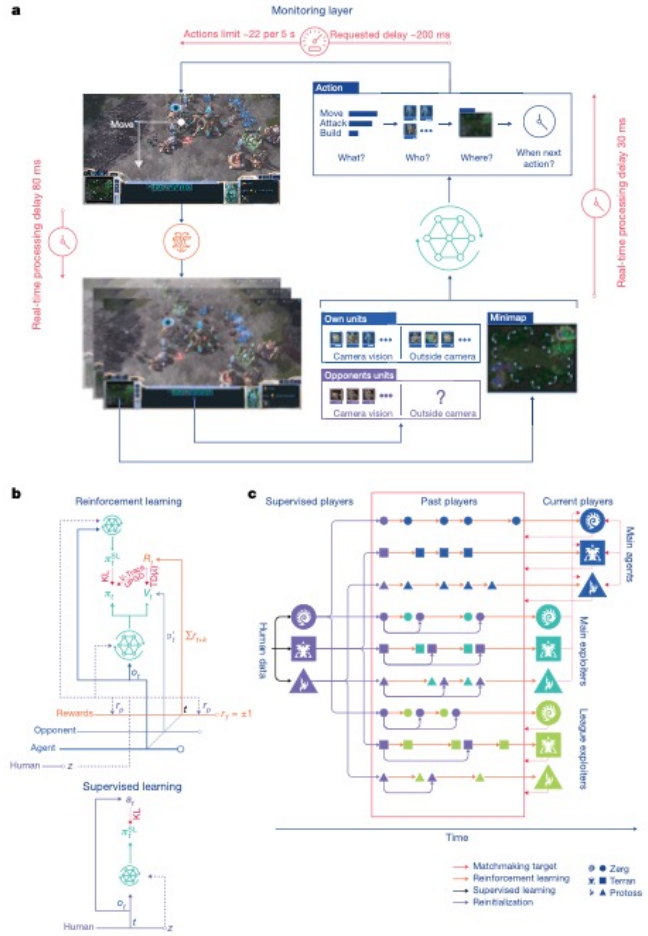
  Eliza is a mock Rogerian psychotherapist.
  The original program was described by Joseph Weizenbaum in 1966.
  This implementation by Norbert Landsteiner 2005.


ELIZA: Is something troubling you ?
YOU:   Men are all alike.
ELIZA: What is the connection, do you suppose ?
YOU:   They're always bugging us about something or other.
ELIZA: Can you think of a specific example ?
YOU:   Well, my boyfriend made me come here.
ELIZA: Is it important to you that your boyfriend made you come here ?
YOU:   He says I'm depressed much of the time.
ELIZA: I am sorry to hear that you are depressed.
YOU:   It's true. I am unhappy.
ELIZA: Can you explain what made you unhappy ?
YOU:
```

Weizenbaum, “ELIZA,” CACM 1966

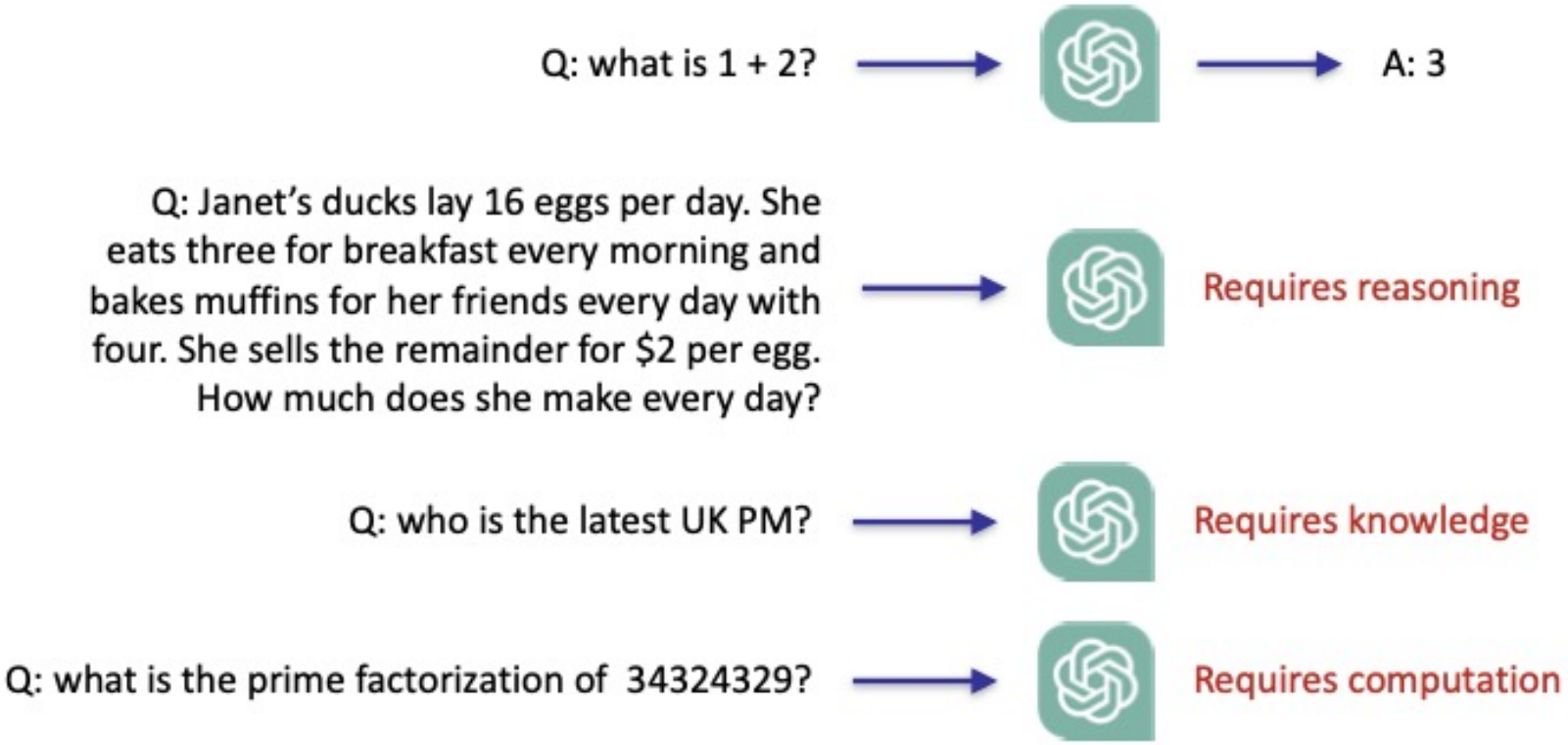
A history of agents

- Use deep reinforcement learning (e.g., AlphaGo, AlphaStar) to learn policies from reward



Vinyals et al., "Grandmaster level in StarCraft II using multi-agent reinforcement learning," Nature 575 (2019)

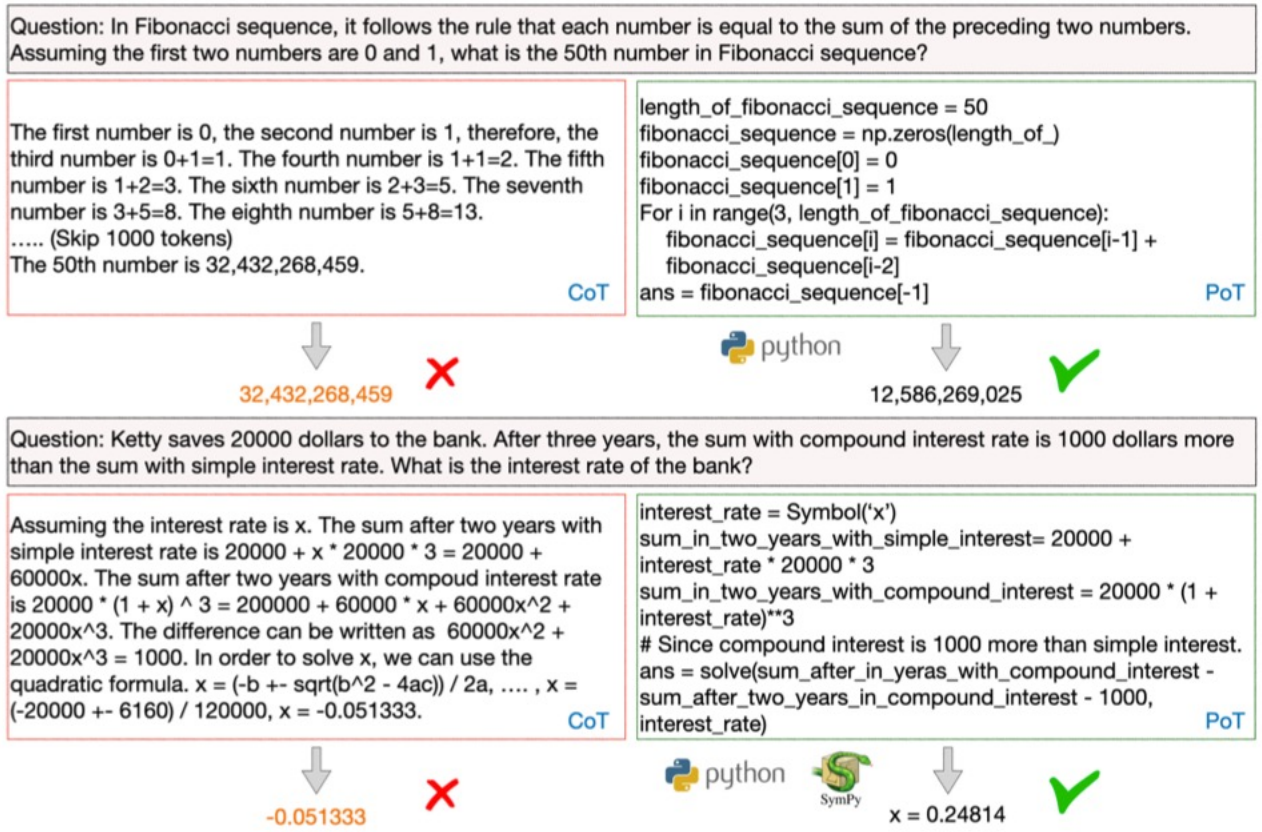
What about LLMs?



Adapted from Shunyu Yao

Zoo of specialized solutions

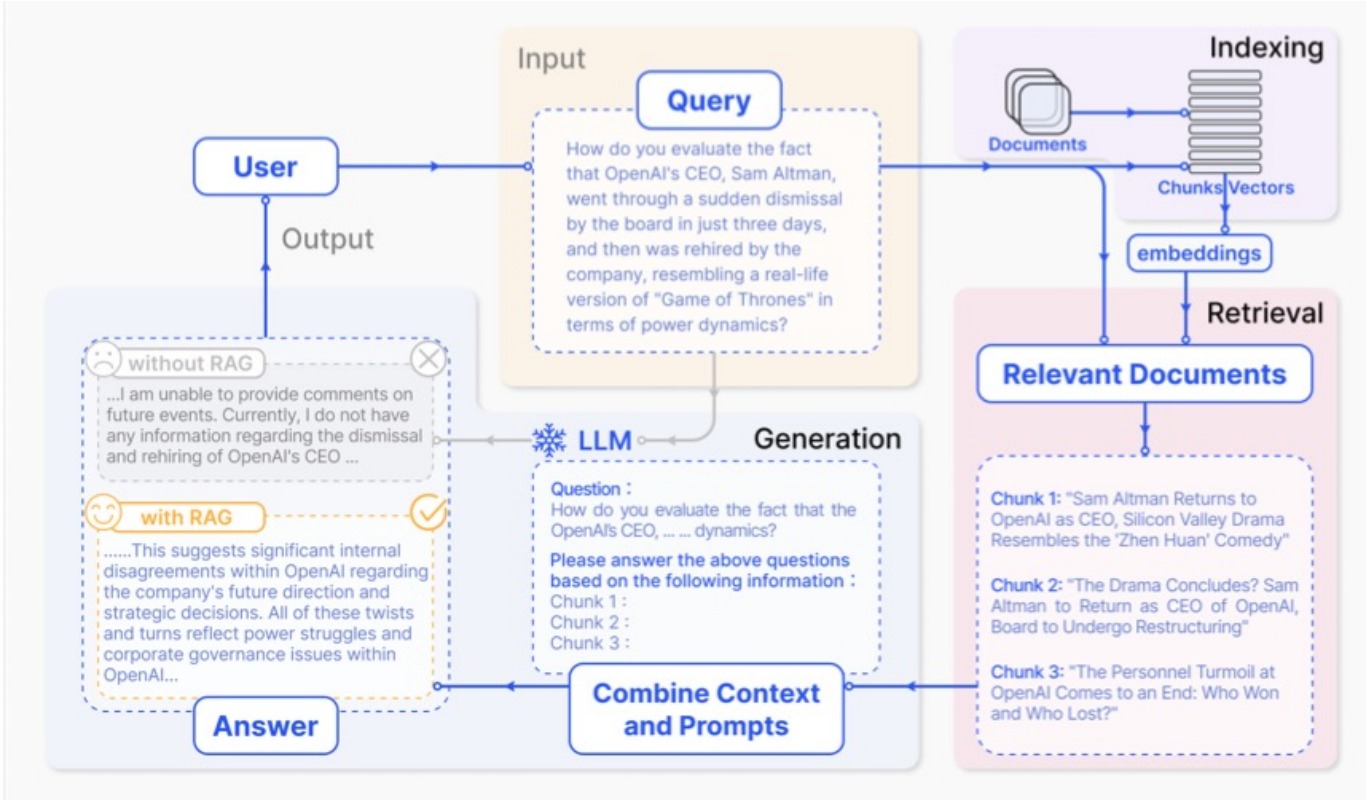
- Example: Prompt model to separate reasoning from computation by allowing external interpreter to handle computation



Chen et al., "Program of Thoughts Prompting: Disentangling Computation from Reasoning for Numerical Reasoning Tasks," TMLR 2023

Zoo of specialized solutions

- Answer knowledge-intensive questions by referencing external corpora



Gao et al., "Retrieval-Augmented Generation for Large Language Models: A Survey," arXiv:2312.10997

Zoo of specialized solutions

- Introduce special tokens or fine-tune model to invoke tool calls
  - E.g., Search engine, calculator, APIs

The New England Journal of Medicine is a registered trademark of **[QA("Who is the publisher of The New England Journal of Medicine?") → Massachusetts Medical Society]** the MMS.

Out of 1400 participants, 400 (or **[Calculator(400 / 1400) → 0.29]** 29%) passed the test.

The name derives from "la tortuga", the Spanish word for **[MT("tortuga") → turtle]** turtle.

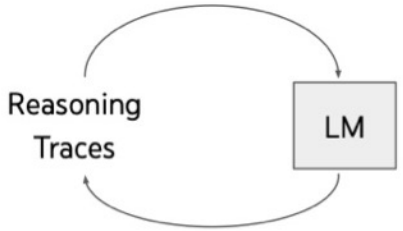
The Brown Act is California's law **[WikiSearch("Brown Act") → The Ralph M. Brown Act is an act of the California State Legislature that guarantees the public's right to attend and participate in meetings of local legislative bodies.]** that requires legislative bodies, like city councils, to hold their meetings open to the public.

Schick et al., "Toolformer: Language Models Can Teach Themselves to Use Tools," NeurIPS 2023

Zoo of specialized solutions

- Needed to specialize how you use an LLM to answer a question
  - Reasoning? CoT
  - Knowledge? RAG
  - Computation? Code/Tools

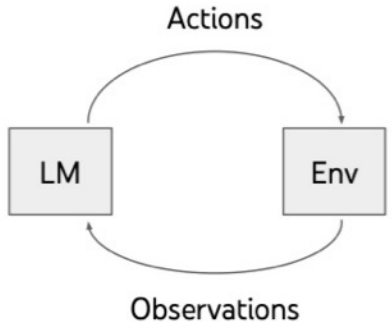
CoT



Flexible and general to augment test-time compute

Lack of external knowledge and tools

RAG/Retrieval/Code/Tool use



Retrieval
Search engine
Calculator
Weather API
Python
.....

Lack of reasoning

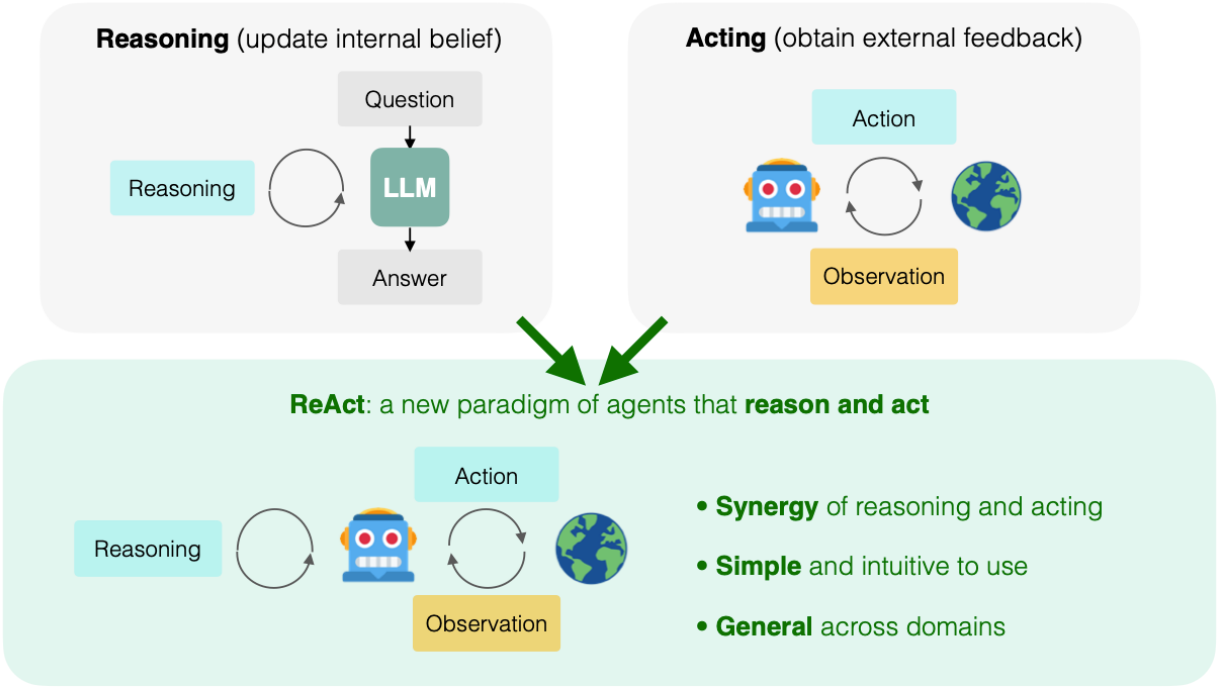
Flexible and general to augment knowledge, computation, feedback, etc.

Adapted from Shunyu Yao

ReAct: Reasoning and Action

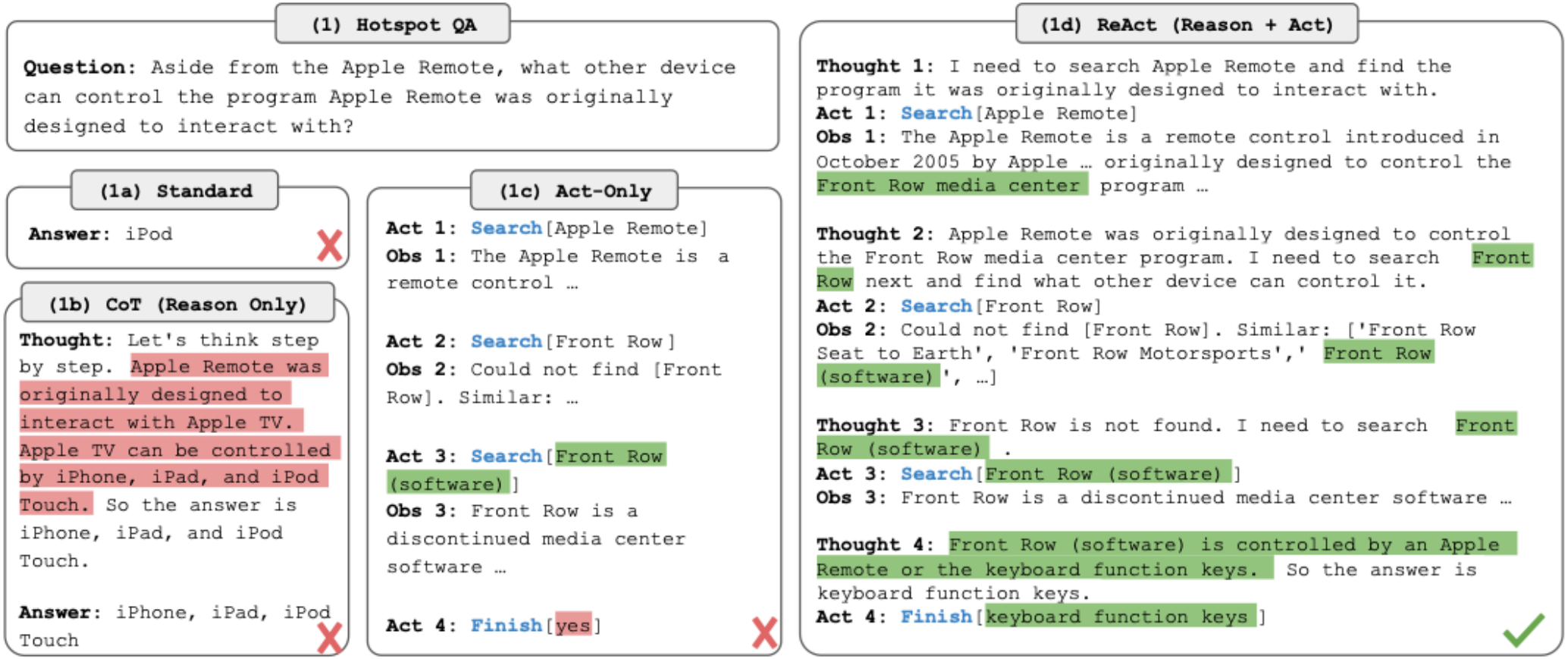
- ReAct: Unified abstraction for reasoning and acting

- **$\hat{A} = A \cup L$**
  - A: Environment actions
    - Search, lookup, retrieve, …
  - L: "Thought" action
    - Does not affect environment (no observation feedback)
    - Compose useful information, update the context: c_t+1 = (c_t, â_t)



Yao et al., "ReAct: Synergizing Reasoning and Acting in Language Models," ICLR 2023

A ReAct trajectory

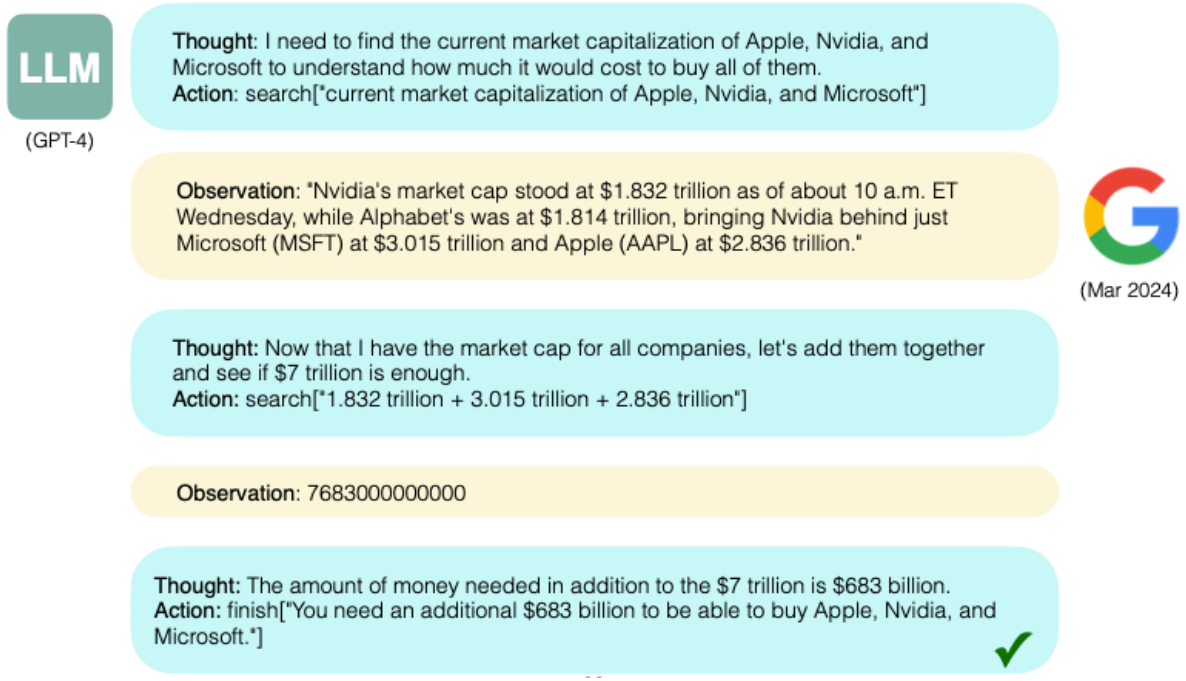


A zero-shot ReAct prompt

You are an agent that answers questions by using two actions:
- search[query]: Google search the query. You can also use it to calculate math.
- finish[answer]: return the answer

Your generation should have the following format:
Thought: to analyze the observation and inform the action.
Action: your action

Question: If I have seven trillion dollars today, can I buy Apple, Nvidia, and Microsoft? If not, how much more money do I need?



Adapted from Shunyu Yao

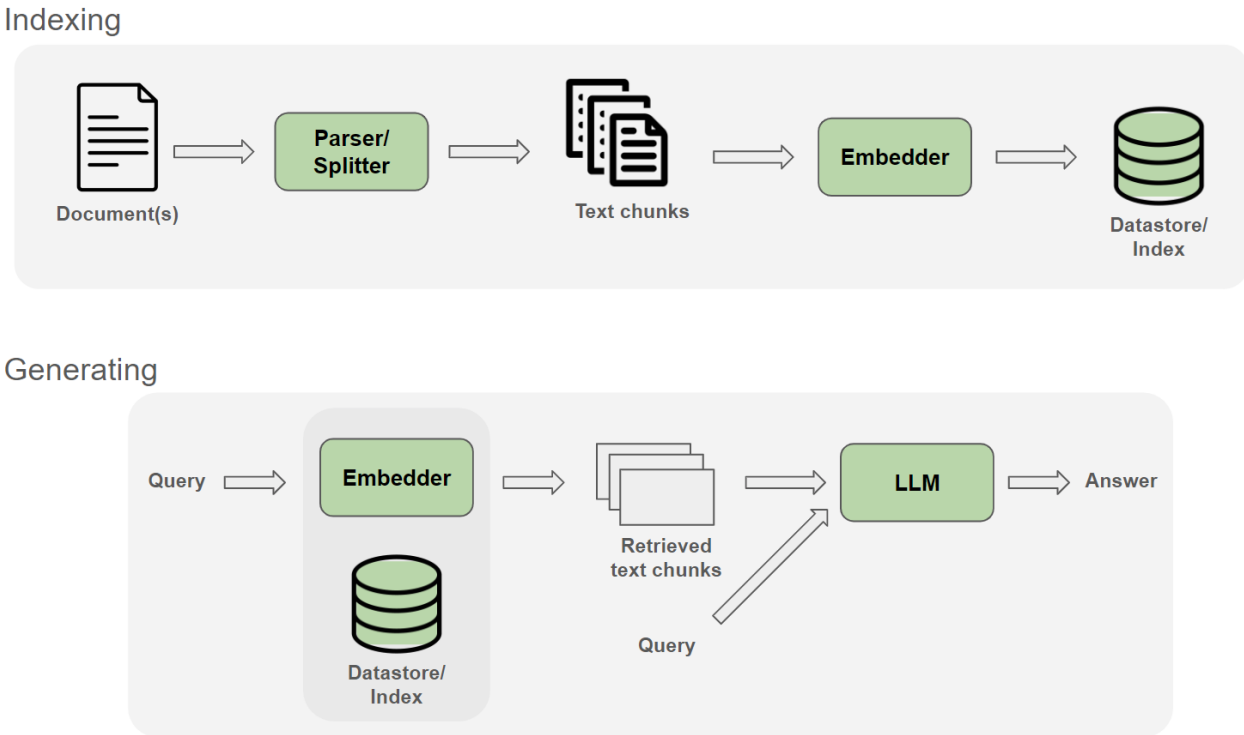
Retrieval Augmented Generation

Why would we want to augment generation with external docs?

- Knowledge is frozen
  - Weights updated until cutoff date, but need information from later
- Private data
  - Company docs, code, user data never used in pretraining
- Grounded answers
  - Model reads sources it can cite instead of hallucinating facts from weights

RAG Pipeline

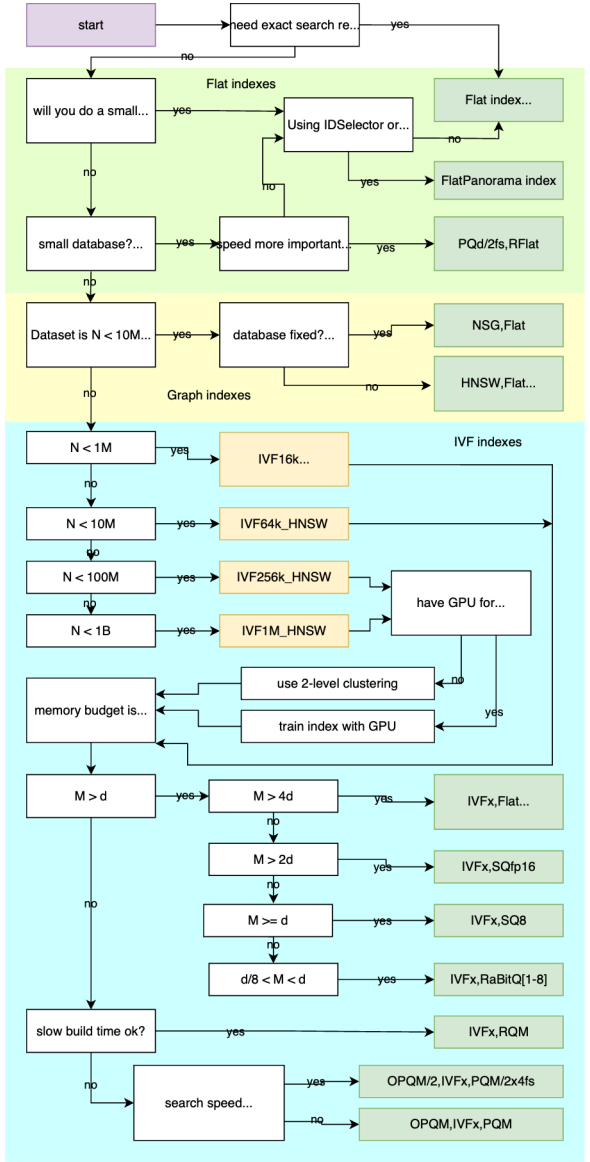
- Build index offline: split documents, embed chunks, build index
- Query online: embed query, perform approximate nearest neighbors search (ANN) on index, fetch top-K chunks, add retrieved documents into context



https://docs.nvidia.com/nemo-framework/user-guide/24.07/rag/ragoverview.html

“Approximate” Nearest Neighbor Search

- How do we find the closest top-k embeddings to our query?
  - Exact? Scan each item
  - … otherwise resort to other index choices…

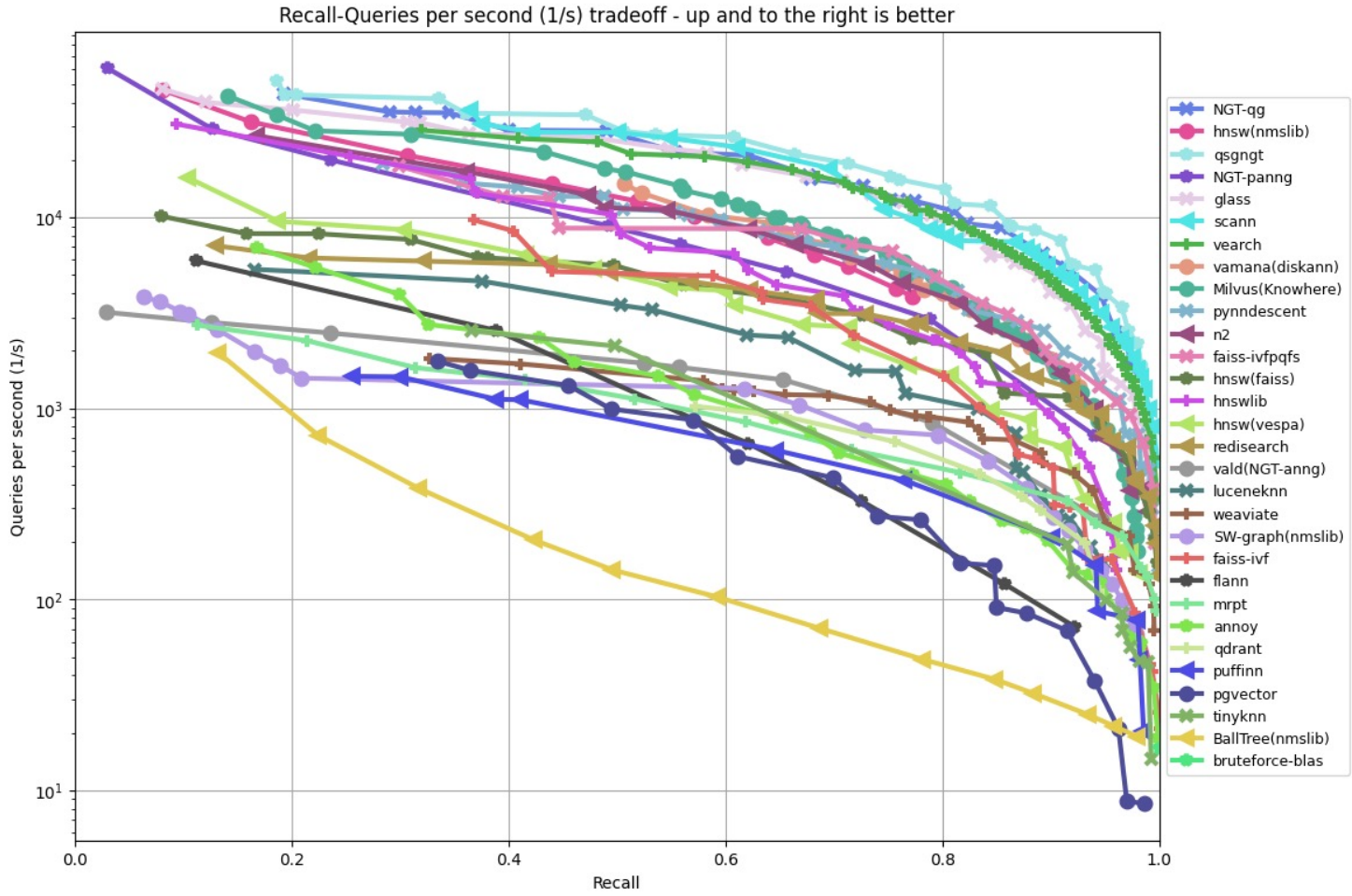


https://github.com/facebookresearch/faiss/wiki/Guidelines-to-choose-an-index

Vector indices

- Fundamentally, trade-off between accuracy (recall) and speed (queries per second)

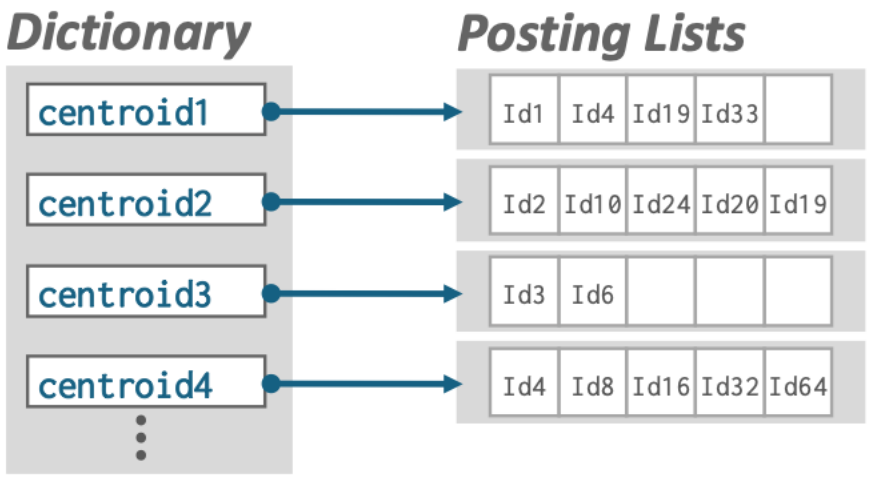
- Two common variants of indices
  - Partition-based
  - Graph-based

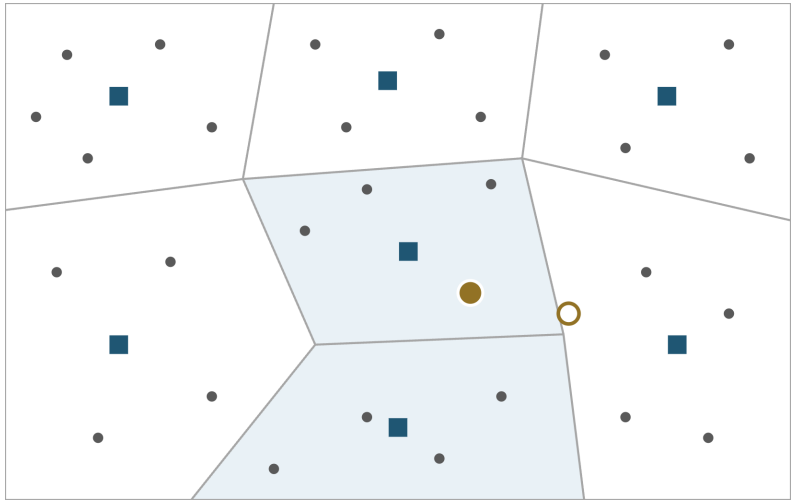


https://ann-benchmarks.com/index.html

Inverted File Index (IVF)

- Idea: Partition vectors into smaller groups using a clustering algorithm, then build an inverted index that maps cluster centroids to records
- To search: scan all centroids to find *p* closest, then scan their postings and return the best *k* candidates

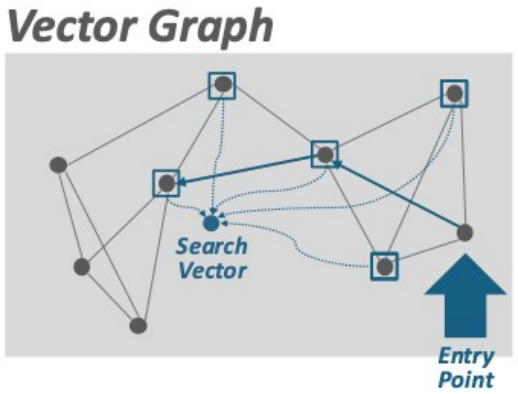


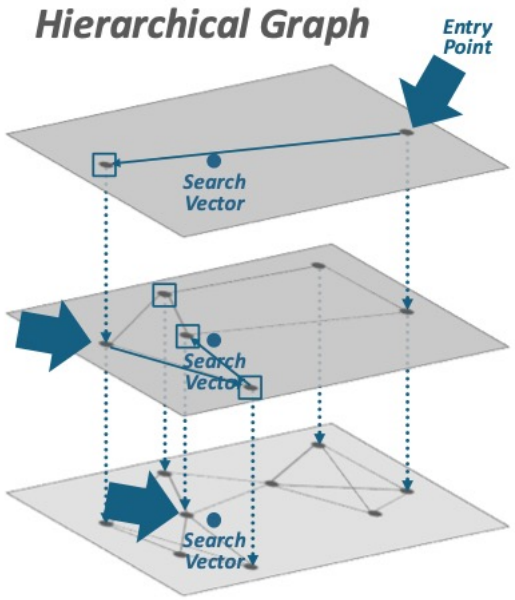


■ centroid · ● query · shaded: the nprobe = 2 cells scanned · ○ true nearest neighbor, missed

Graph indices

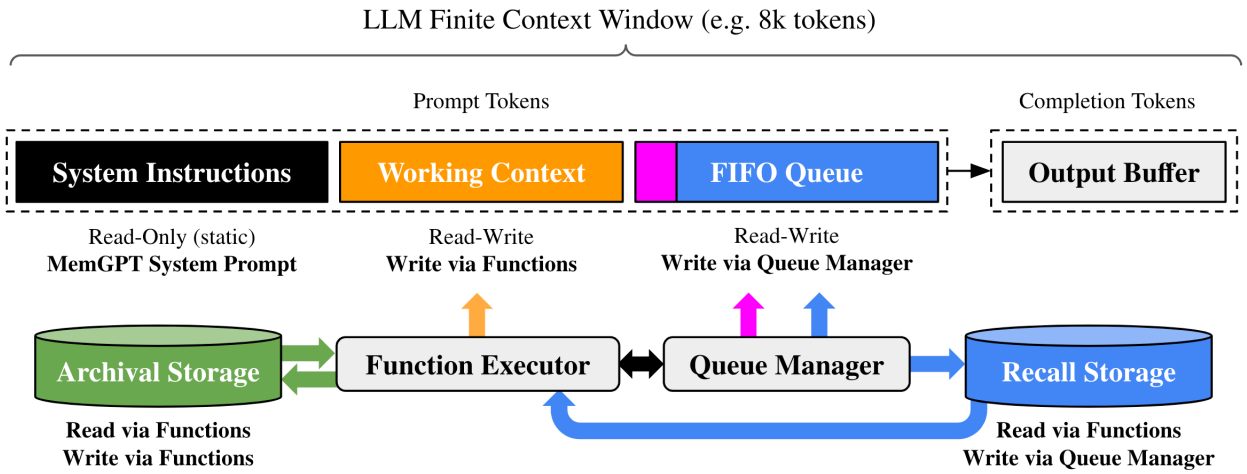
- Idea: build a graph where each node represents a vector and has edges to some of its nearest neighbors
- Search: Enter the graph, then greedily choose the next edge that moves closer to to that vector
- Optimization: Hierarchical Navigable Small Worlds (HNSW)
  - Idea: Higher layers are sparser (faster search), lower layers are denser (more detailed search)

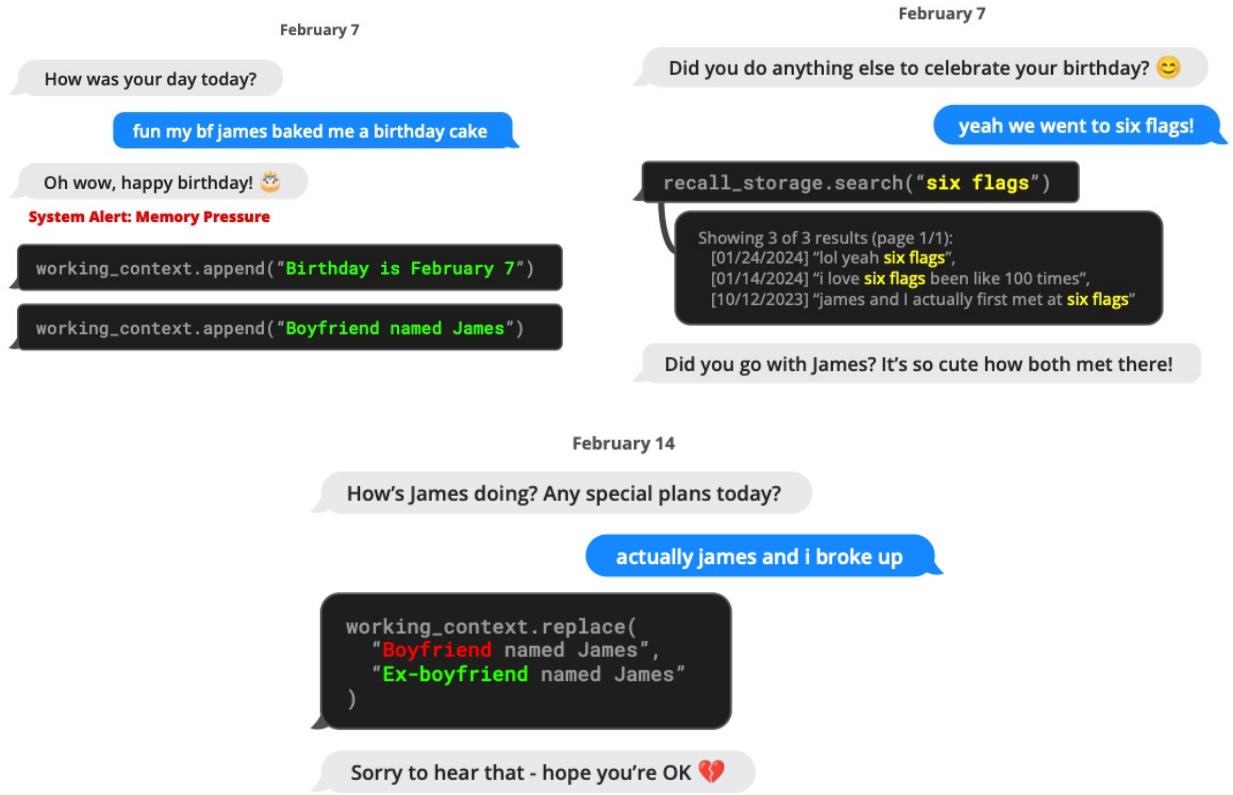




Agentic Memory

- Key idea: Extend LLM's working context (analogous to RAM) with external context (analogous to disk).
  - Give models tools to edit its own memory and "page" between the two

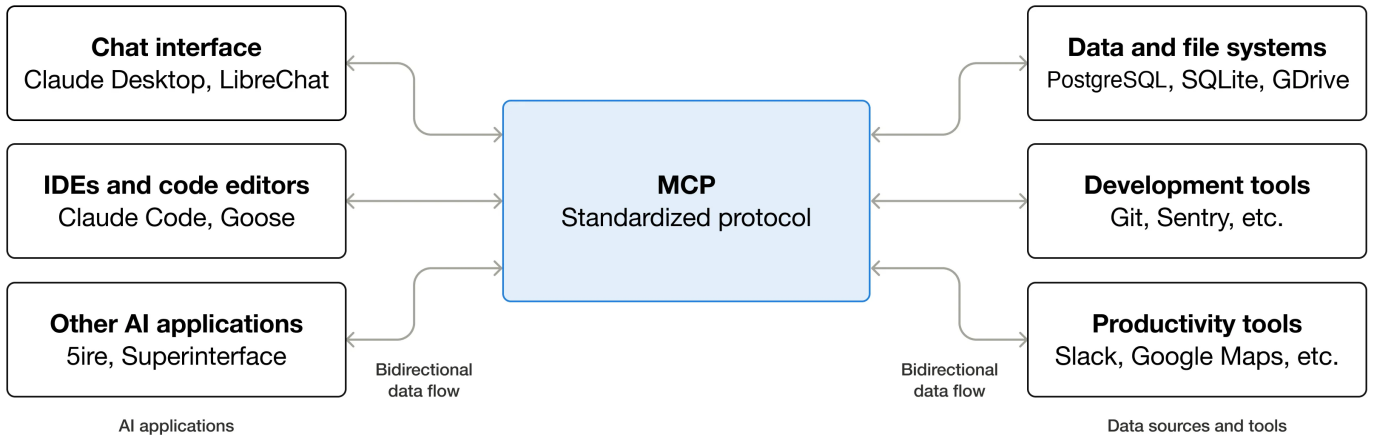




MemGPT: Towards LLMs as Operating Systems; https://arxiv.org/abs/2310.08560

Model Context Protocol

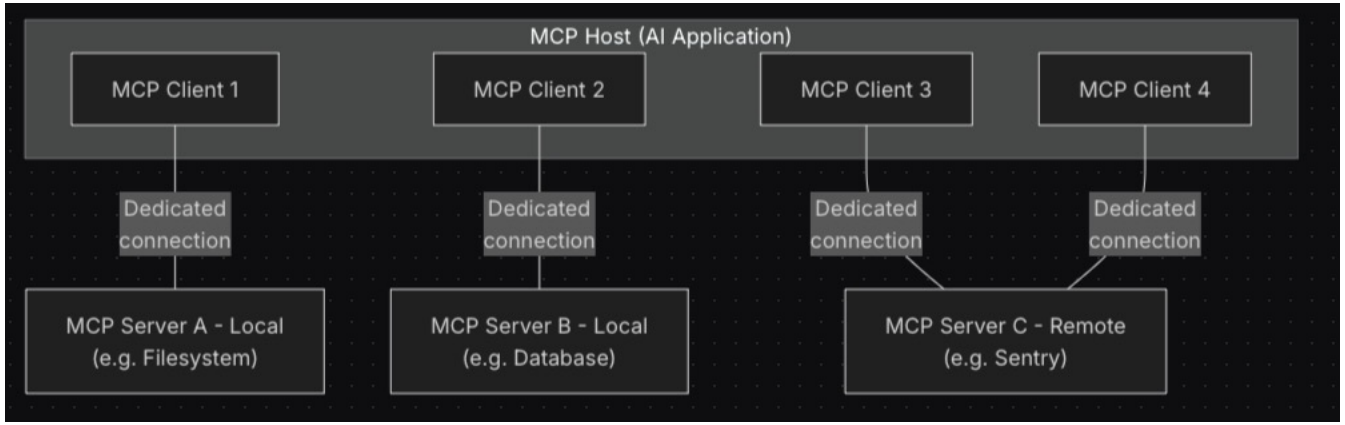
- Standardize how LLMs interface with external data sources and tools
- Analogy: “USB-C for AI applications”



https://modelcontextprotocol.io

MCP Architecture

- Client-server architecture
  - AI app (host) creates one MCP client for each MCP server
  - Each client maintains a dedicated connection with each server
  - Data layer: JSON-RPC 2.0 to define message structure/semantics
  - Transport layer: stdio for local servers, HTTP for remote



MCP Example

1 · List tools · client → server

```
{"jsonrpc": "2.0", "id": 1, "method": "tools/list",
 "params": {"_meta": {
   "io.modelcontextprotocol/protocolVersion":
       "2026-07-28",
   "io.modelcontextprotocol/clientCapabilities": {}}}}
```

→

Reply · server → client

```
{"jsonrpc": "2.0", "id": 1, "result": {
 "resultType": "complete",
 "tools": [{"name": "search_docs",
   "description": "Top-k search over course notes",
   "inputSchema": {"type": "object", "properties":
     {"query": {"type": "string"},
      "k": {"type": "integer"}},
     "required": ["query"]}}],
 "ttlMs": 300000, "cacheScope": "public"}}
```

2 · Call a tool · client → server

```
MCP-Protocol-Version: 2026-07-28
Mcp-Method: tools/call  Mcp-Name: search_docs
{"jsonrpc": "2.0", "id": 2, "method": "tools/call",
 "params": {"name": "search_docs",
  "arguments": {"query": "IVF nprobe", "k": 2},
  "_meta": {…same two keys as request 1…}}}
```

→

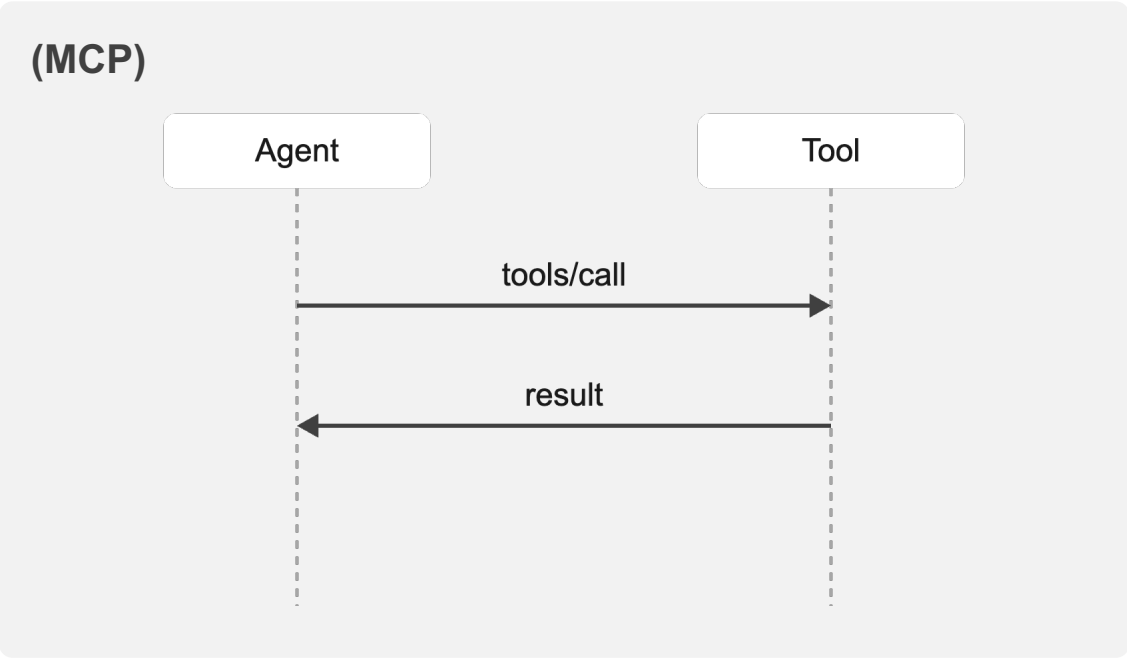
Reply · server → client

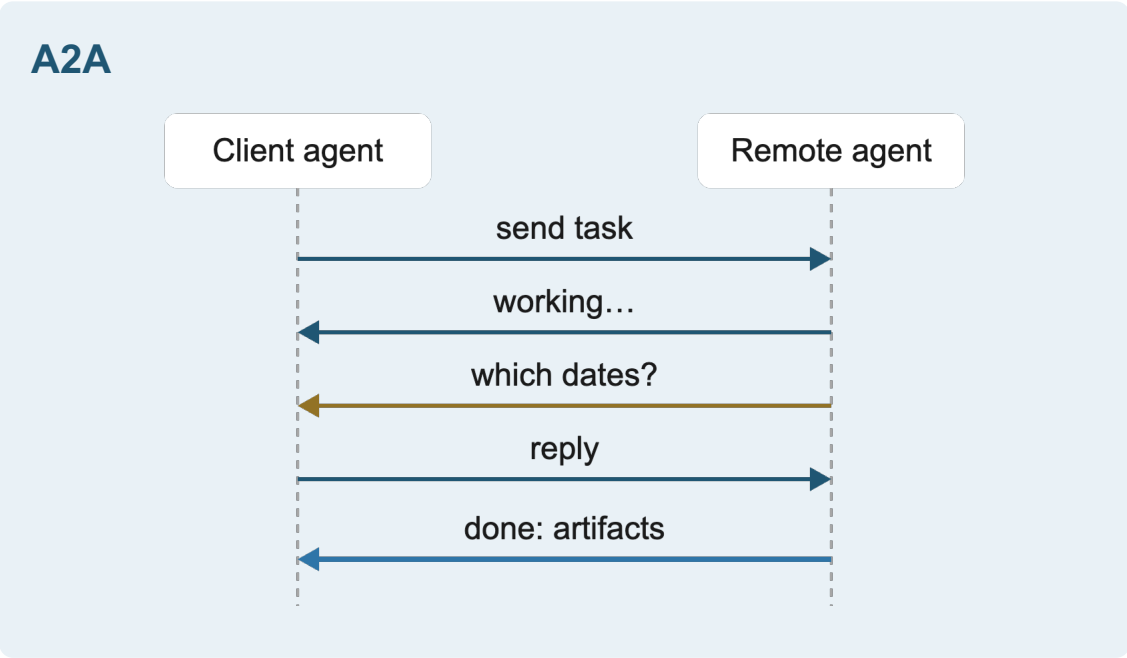
```
{"jsonrpc": "2.0", "id": 2, "result": {
 "resultType": "complete",
 "content": [{"type": "text",
   "text": "IVF scans only the nprobe closest…"},
  {"type": "text",
   "text": "HNSW walks a layered graph…"}],
 "isError": false}}
```

Agent2Agent Protocol (A2A)

- MCP is to tools as A2A is to agents

- Common protocol to standardize how agents communicate and collaborate with each other (potentially across different owners/nodes/services)





https://a2a-protocol.org/latest/

A2A Lifecycle

1 · Discover · client → remote agent

```
GET /.well-known/agent-card.json
Host: research.example.edu
```

Reply · the Agent Card

```
{"name": "Research Agent",
 "supportedInterfaces": [{
   "url": "https://research.example.edu/a2a",
   "protocolBinding": "JSONRPC",
   "protocolVersion": "1.0"}],
 "capabilities": {"streaming": true},
 "skills": [{"id": "lit-survey",
   "name": "Literature survey", "tags": […]}],
 "securitySchemes": {…}}
```

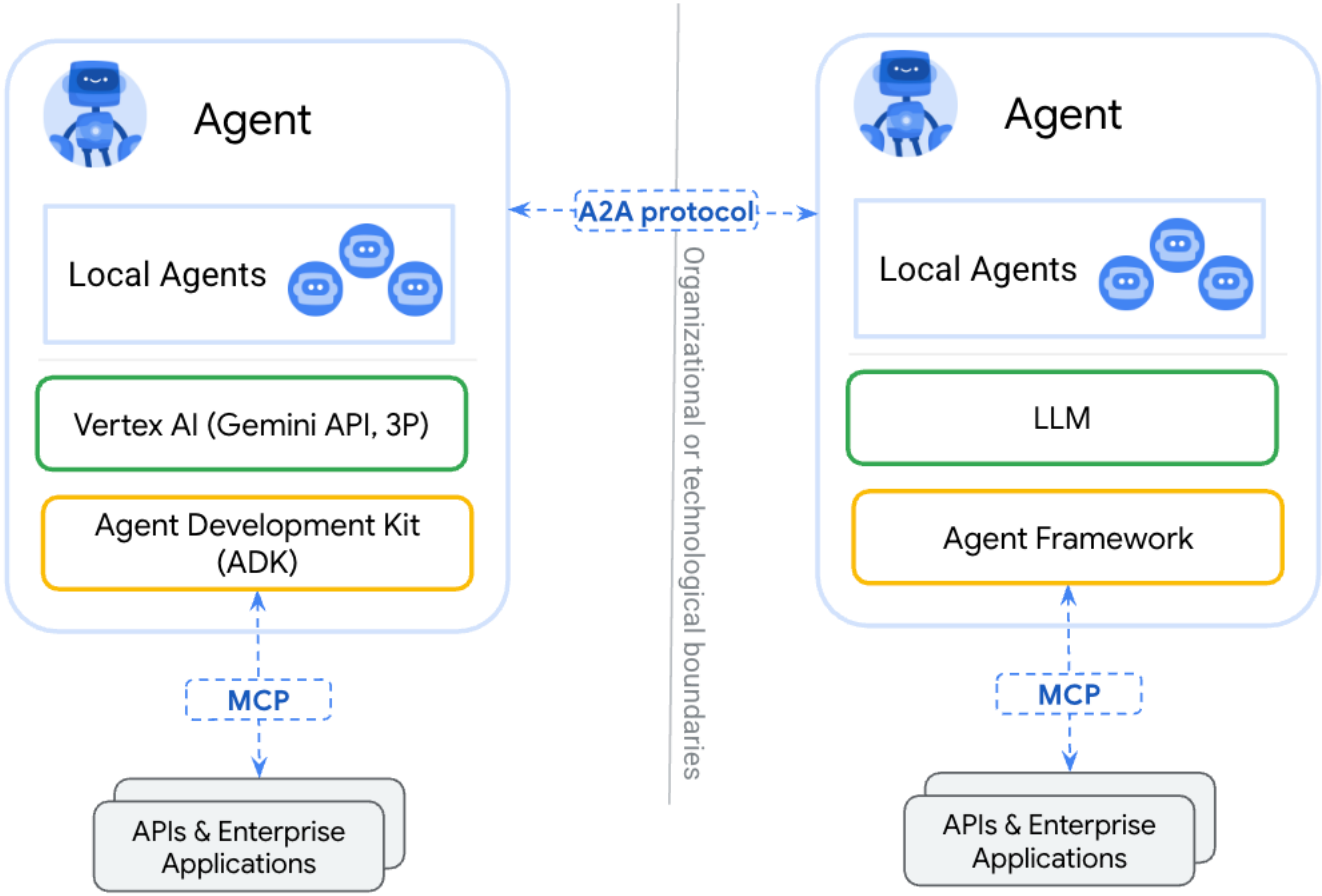
2 · Delegate a task · client → remote agent

```
POST /a2a   A2A-Version: 1.0
{"jsonrpc": "2.0", "id": 1, "method": "SendMessage",
 "params": {"message": {
   "role": "ROLE_USER", "messageId": "m-1",
   "parts": [{"text":
     "Survey KV-cache eviction for agents"}]}}}
```

Reply · a Task with artifacts

```
{"jsonrpc": "2.0", "id": 1, "result": {"task": {
   "id": "task-42", "contextId": "ctx-7",
   "status": {"state": "TASK_STATE_COMPLETED"},
   "artifacts": [{"artifactId": "a-1",
     "name": "Survey", "parts": [{"text":
       "InferCept, Continuum, Agentix, …"}]}]}}}
```

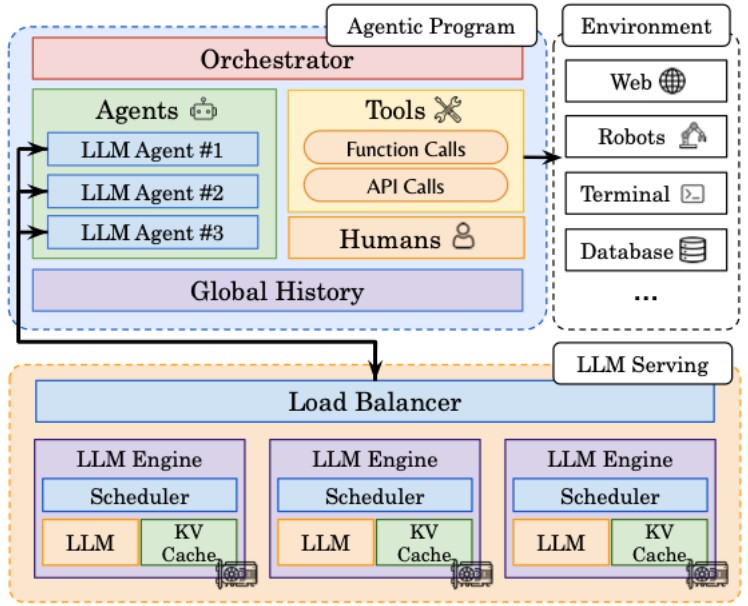
A2A and MCP

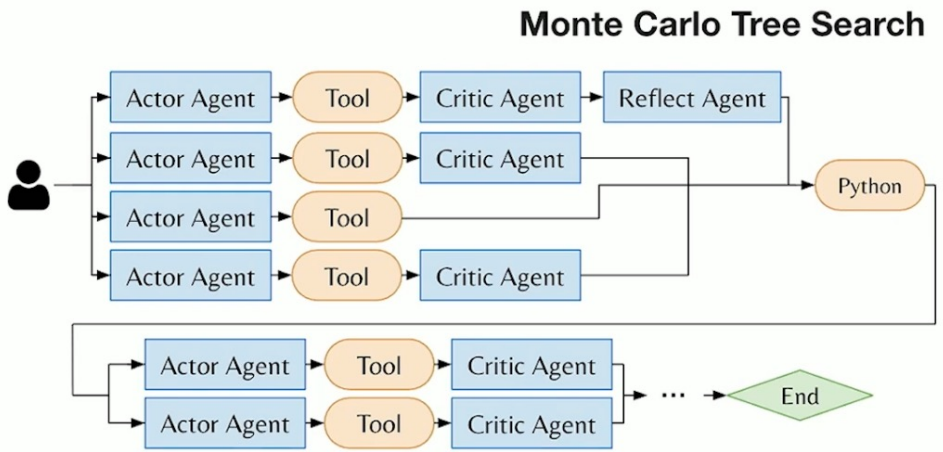

Agent
Local Agents
Vertex AI (Gemini API, 3P)
Agent Development Kit (ADK)
MCP
APIs & Enterprise Applications
A2A protocol
Organizational or technological boundaries
Agent
Local Agents
LLM
Agent Framework
MCP
APIs & Enterprise Applications

Serving Agentic Workloads

Challenges:

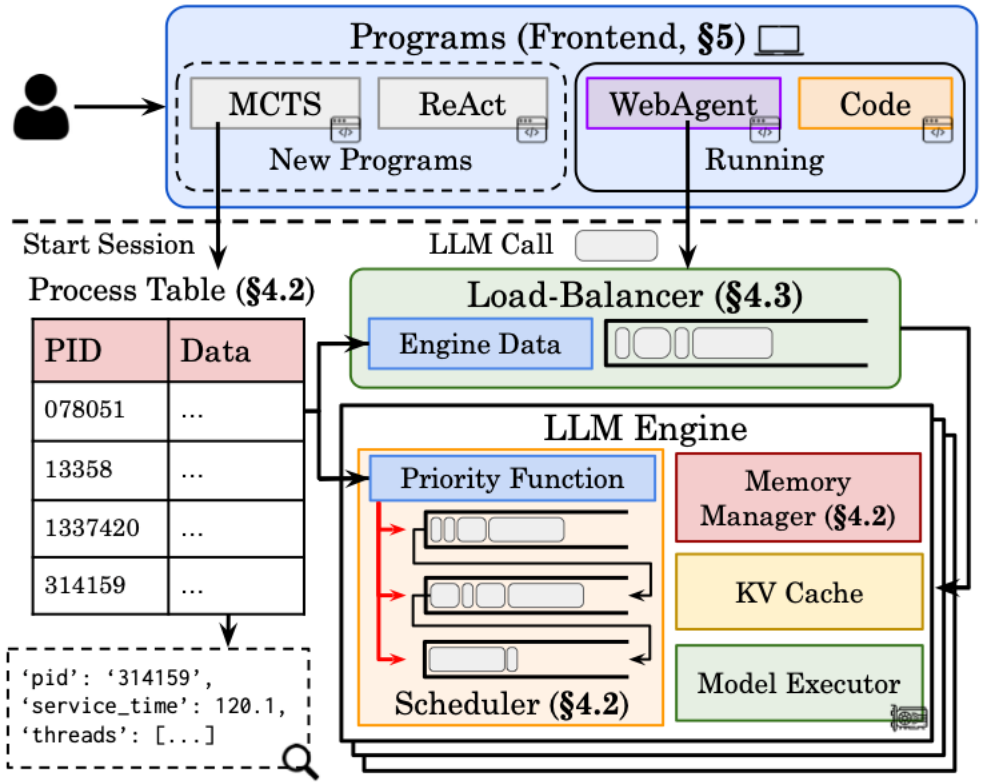
- Agent infrastructure layer decoupled from LLM serving layer
  - *LLM serving system can't exploit properties inherent in agent workflows*
- Agent workflows getting more complex
  - Highly dynamic (unknown length)
  - Distributed (multiple parallel calls)
  - Non-deterministic

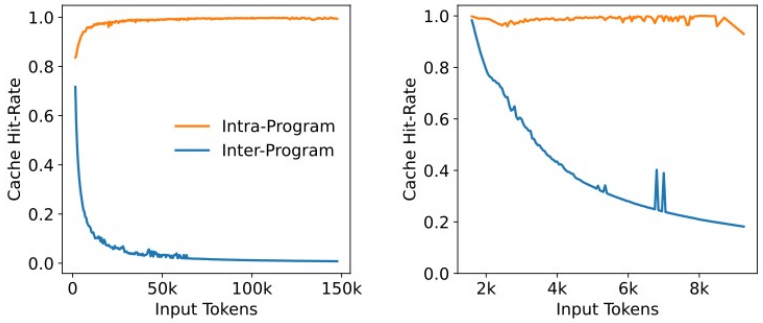




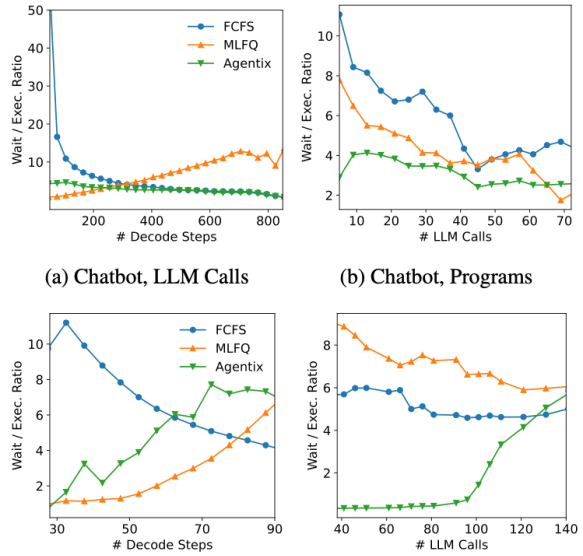
https://www.usenix.org/conference/nsdi26/presentation/luo

Agentix: Serving engine for Agentic Programs





(a) Single thread: Chatbot (b) Multiple threads: MCTS



(a) Chatbot, LLM Calls (b) Chatbot, Programs

(c) MCTS, LLM Calls (d) MCTS, Programs