

MemGPT: Towards LLMs as Operating Systems

Managing memory, not just enlarging context

Presenter: Chi-Han Chiu

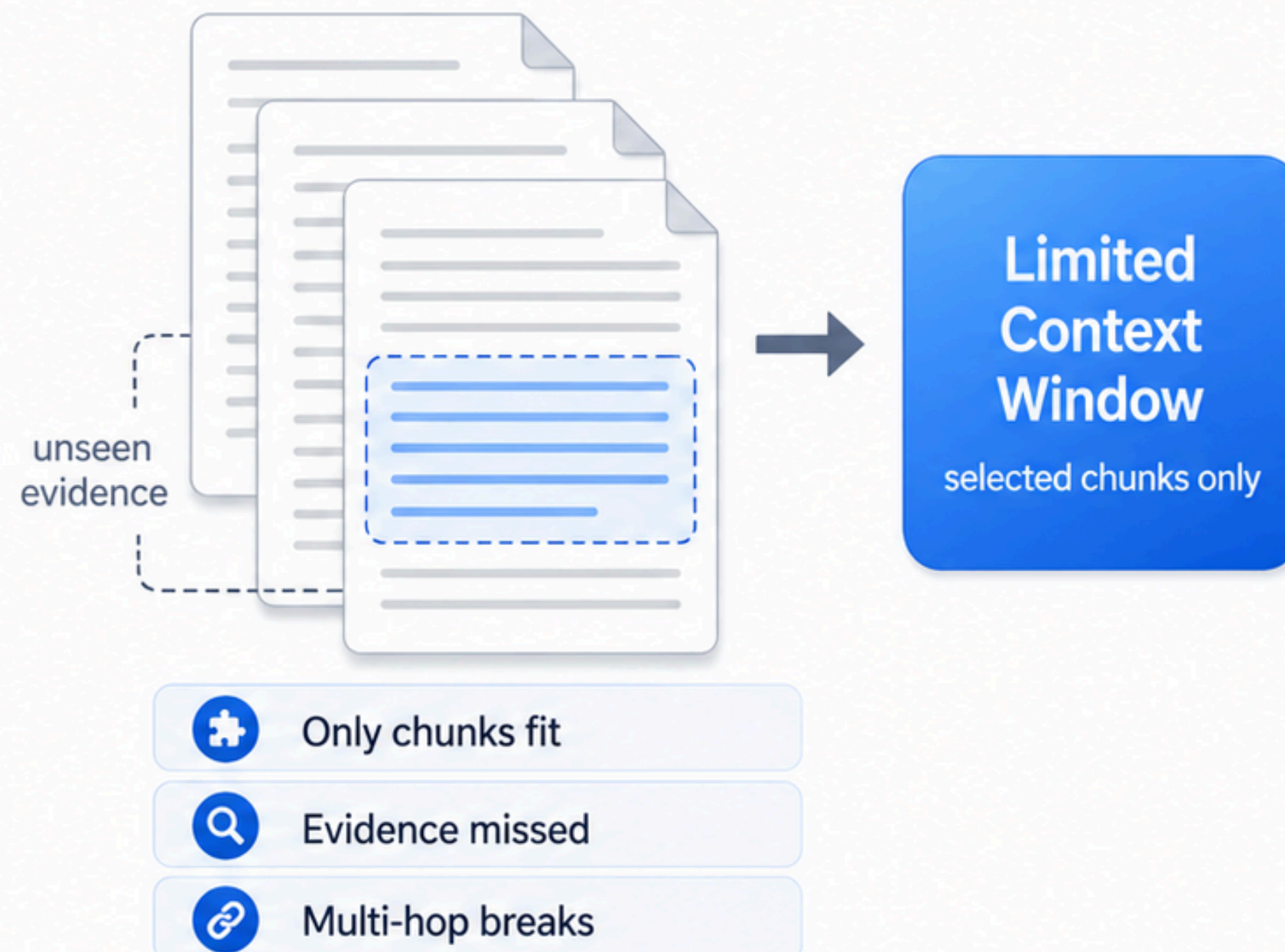
Fixed context is not memory

Two failure modes that motivate MemGPT

Long chat → forgetting



Long documents → partial reading



Discussion 1

“Is a larger context window enough?”

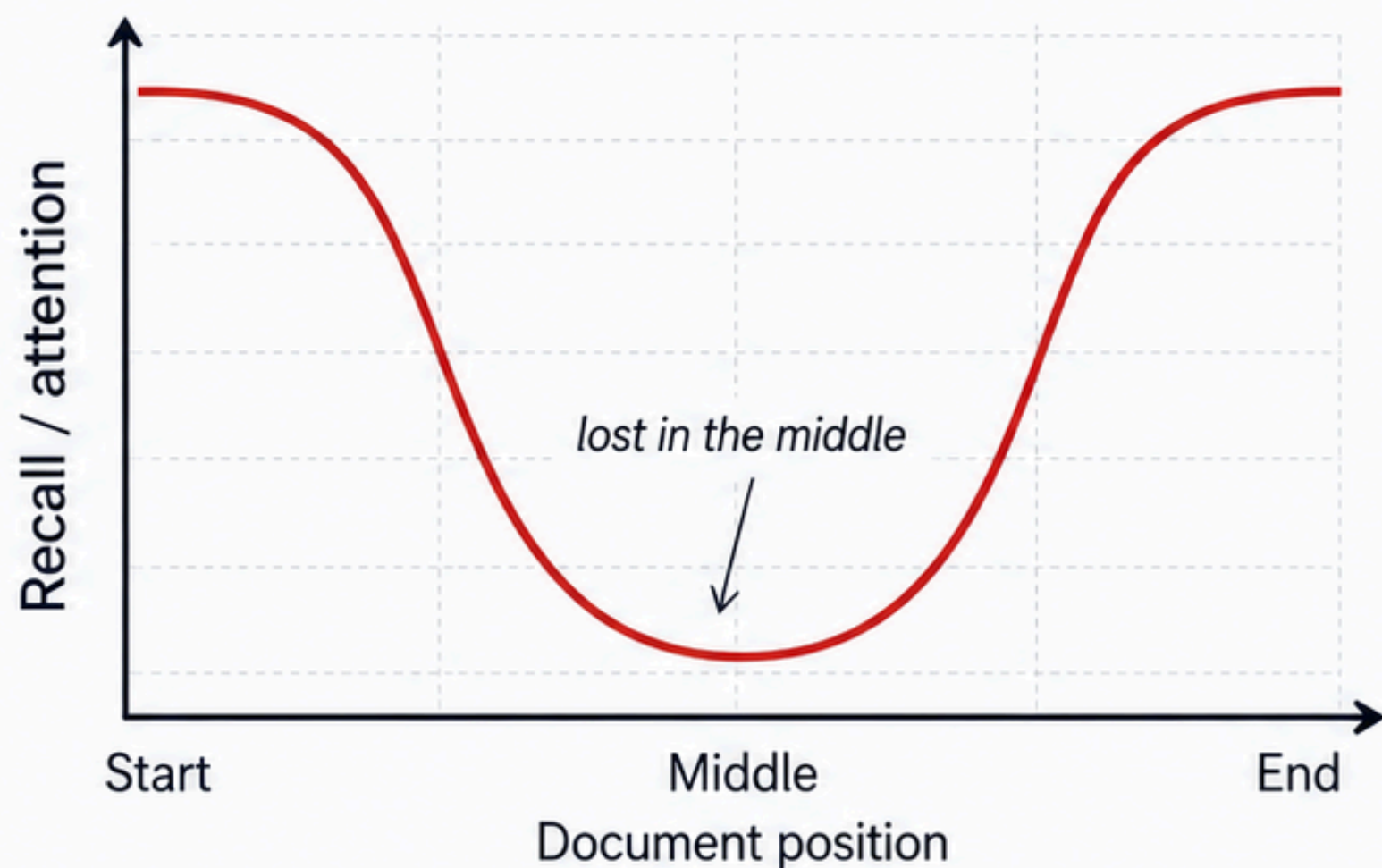
What problem does longer context actually solve?



Why longer context alone is not enough

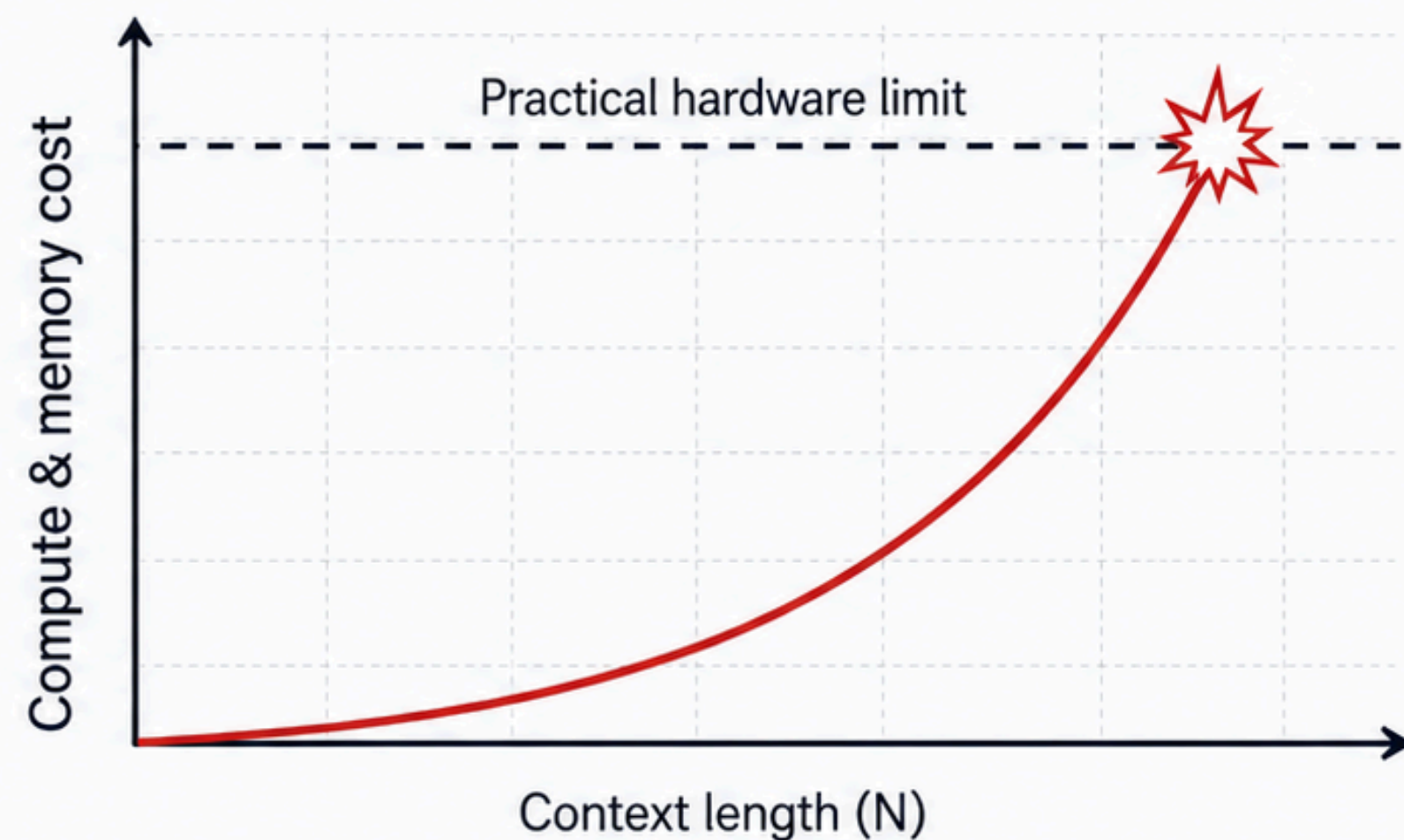
Two limits of finite working memory

1 Attention limit



Important details in the middle are harder to retrieve or reason over.

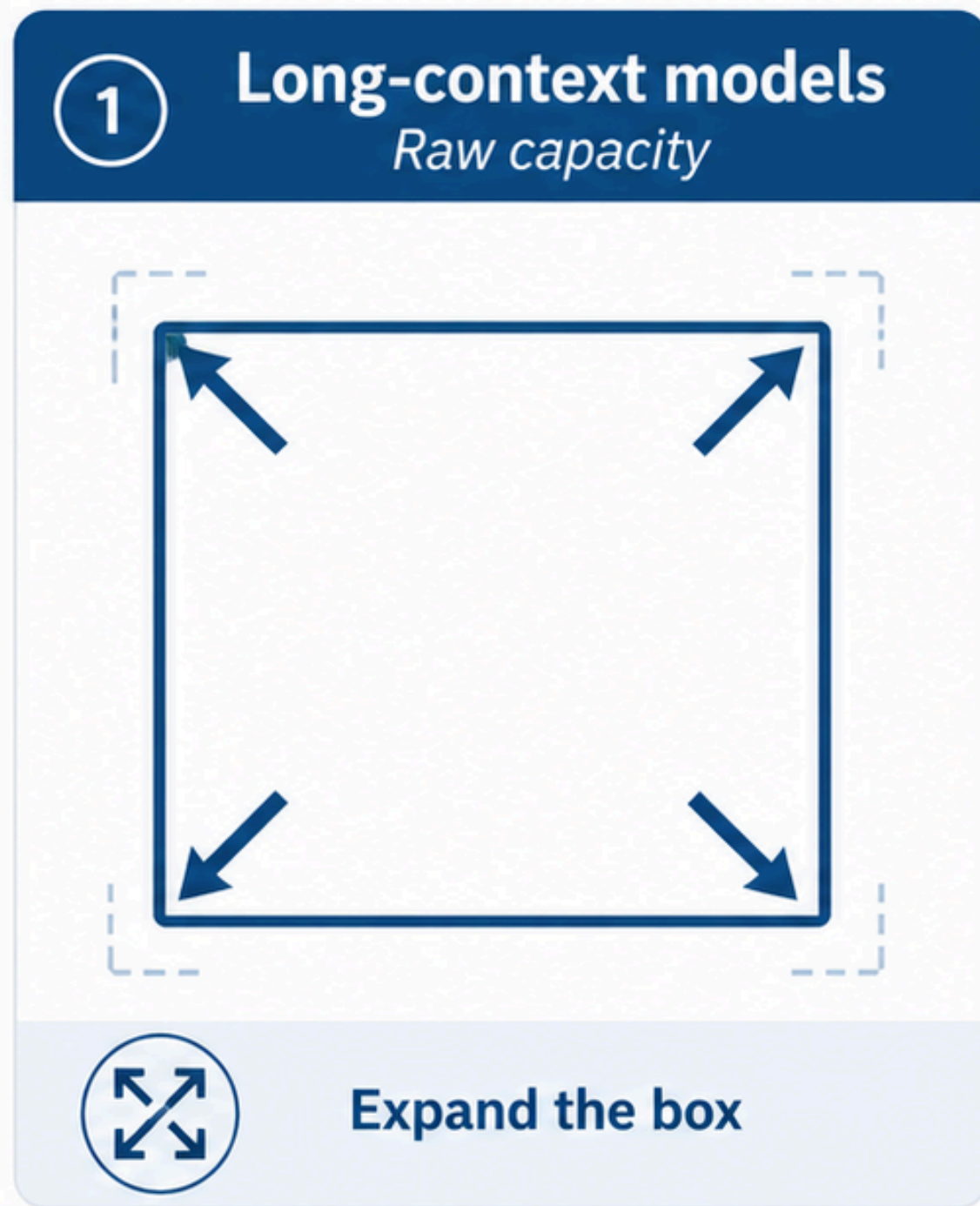
2 Hardware limit



Longer context raises compute and memory cost rapidly.

Not just retrieval

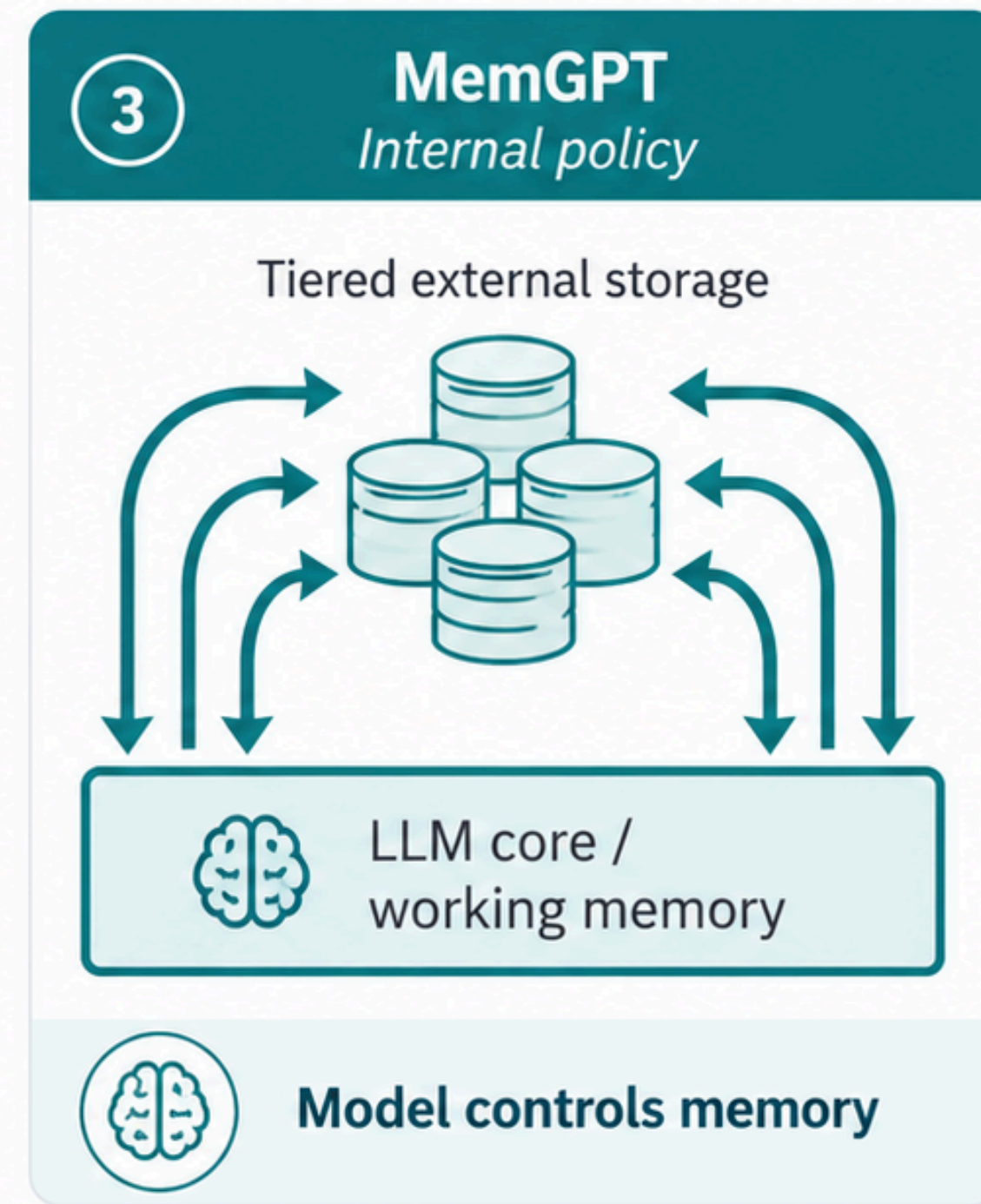
Model-managed memory over time



More capacity



Retrieve what to include



Actively manage over time

Discussion 2

“Who should control memory?”

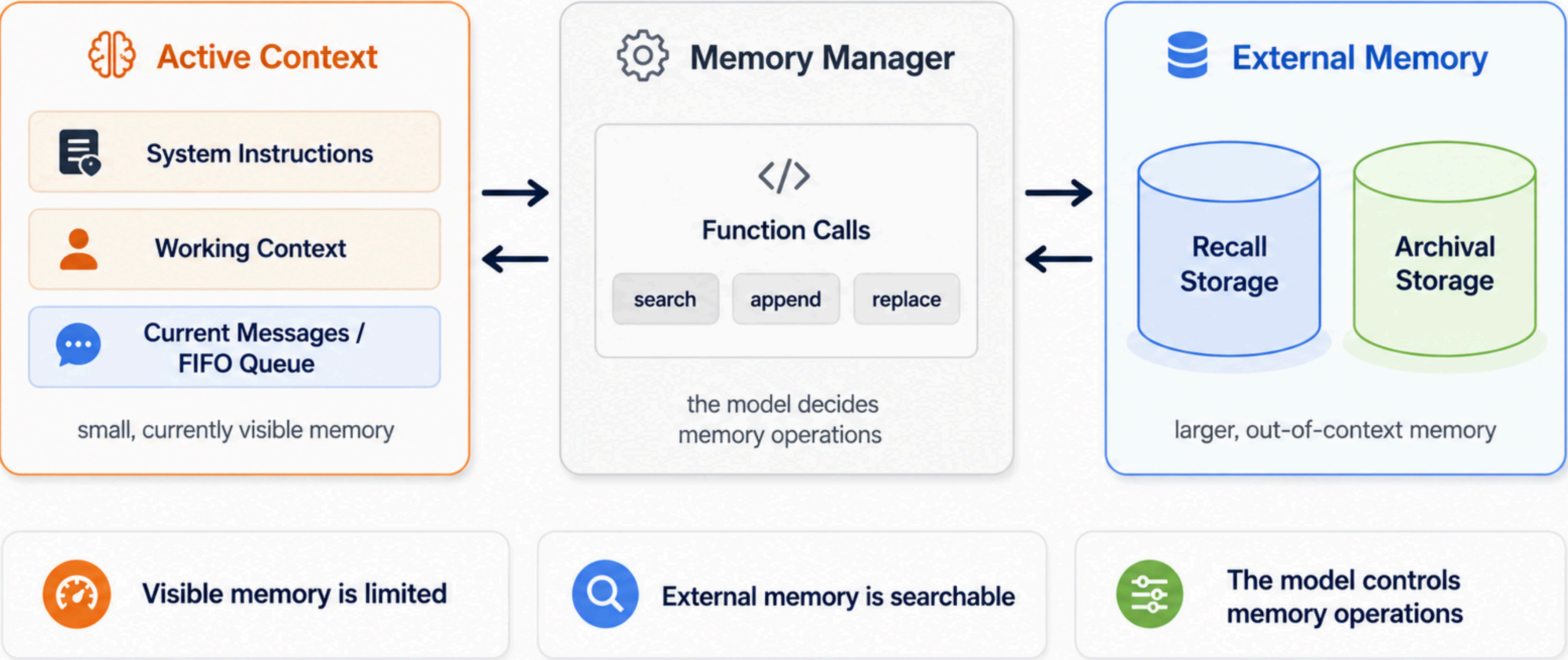
Pipeline policy, or the model itself?



MemGPT = LLM + Memory Manager + External Storage

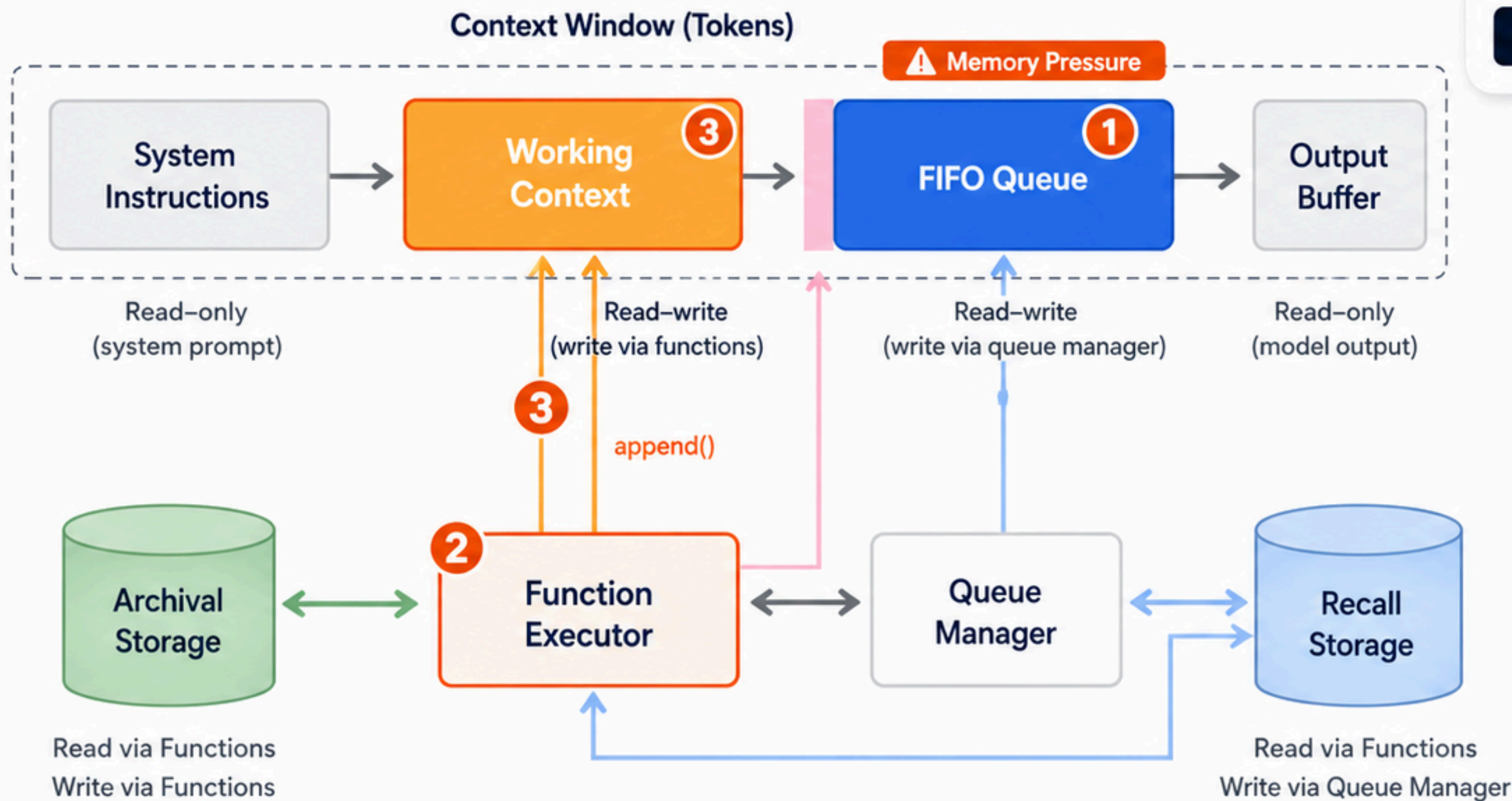
The model actively decides what to keep, retrieve, and update.

Not just retrieval — model-managed memory over time



Writing memory under pressure

How memory writes durable facts when the context is full.



February 7

How was your day today?

fun my bf james baked me a birthday cake

Oh wow, happy birthday! 🎂

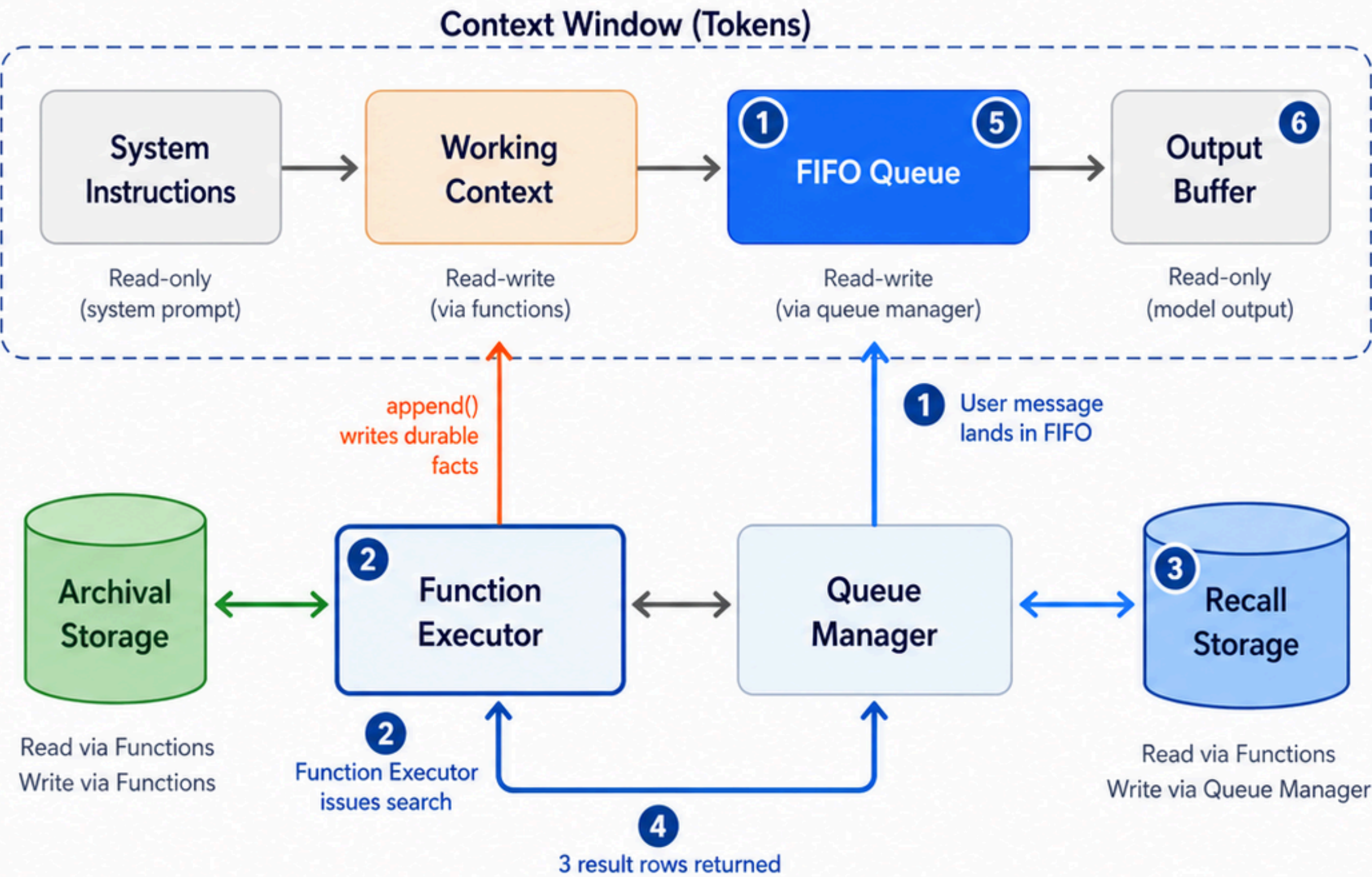
⚠️ System Alert: Memory Pressure

```
working_context.append("Birthday is February 7")
working_context.append("Boyfriend named James")
```

- 1 User message reaches the FIFO queue**
The message arrives as the queue nears its token limit.
- 2 Function executor is triggered**
Memory pressure activates the function executor.
- 3 append() writes durable facts**
Structured facts are written upward into the working context.

Searching memory when needed

How retrieval brings relevant past facts back into the active context.



February 7

Did you do anything else to celebrate your birthday? 😊

yeah we went to six flags!

`recall_storage.search("six flags")`

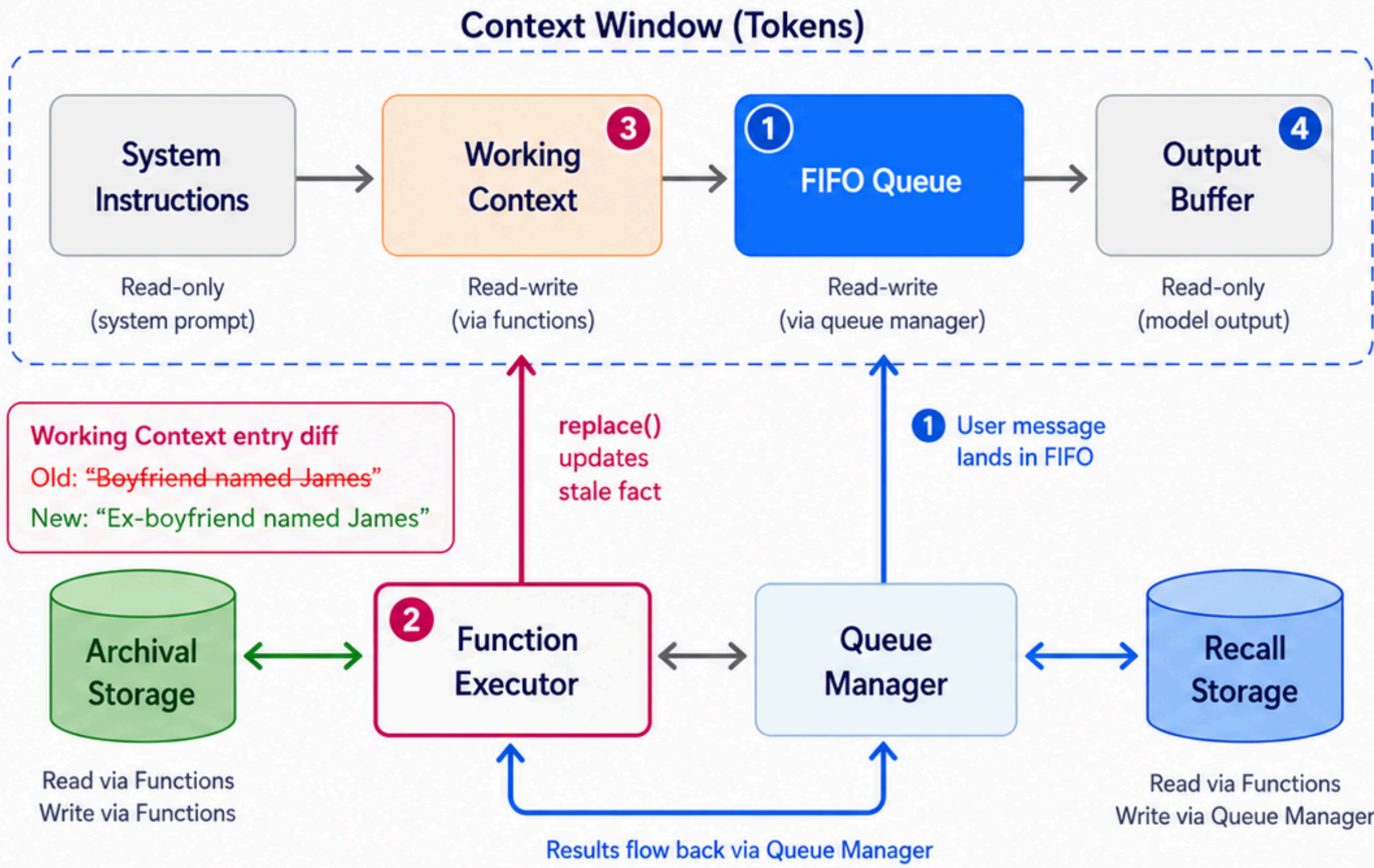
Showing 3 of 3 results [page 1/1]:
[01/24/2024] "lol yeah **six flags**"
[01/14/2024] "i love **six flags** i've been like 100 times"
[10/12/2023] "james and i actually first met at **six flags**"

Did you go with James? It's so cute how both met there!

- 1 User message lands in FIFO
- 2 Function Executor issues search
- 3 Recall Storage matches
- 4 Results travel back via Queue Manager
- 5 Pushed into FIFO as tool output
- 6 Model emits informed reply

Updating stale memory

How MemGPT revises outdated facts instead of keeping stale state.



February 14

How's James doing? Any special plans today?

actually james and i broke up

```
working_context.replace(  
  "Boyfriend named James",  
  "Ex-boyfriend named James"  
)
```

Sorry to hear that - hope you're OK 💔

- 1 Breakup message lands in FIFO Queue
- 2 Function Executor detects stale fact
- 3 replace() updates entry
- 4 Reply emitted

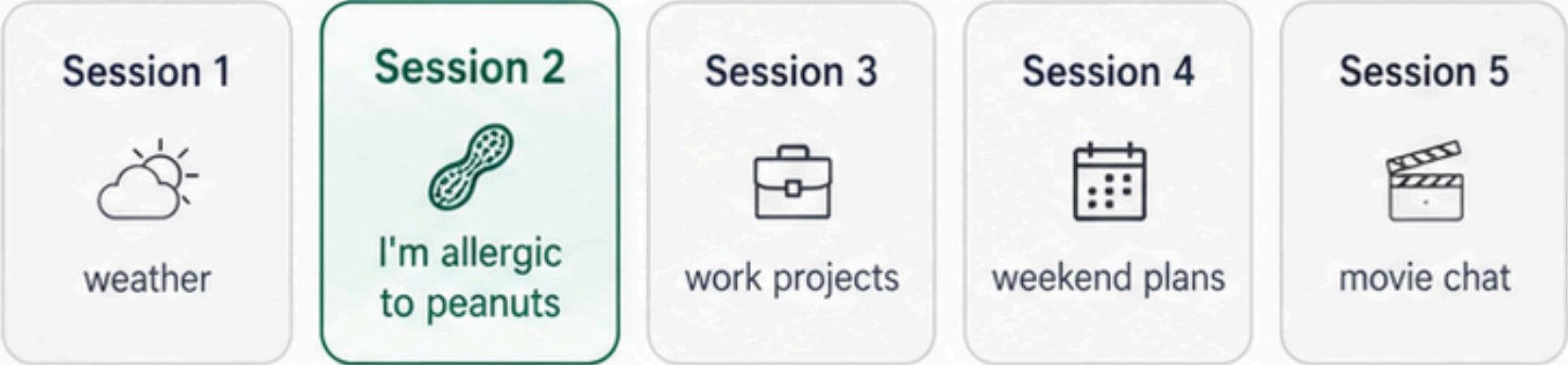
MemGPT improves long-term conversational memory

Deep Memory Retrieval (DMR) tests whether an agent can recover facts from earlier sessions.



Deep Memory Retrieval task

Agent is asked a question about a topic discussed in prior sessions (1–5).



EVALUATION QUESTION

"I'm planning a trip — any food I should avoid?"

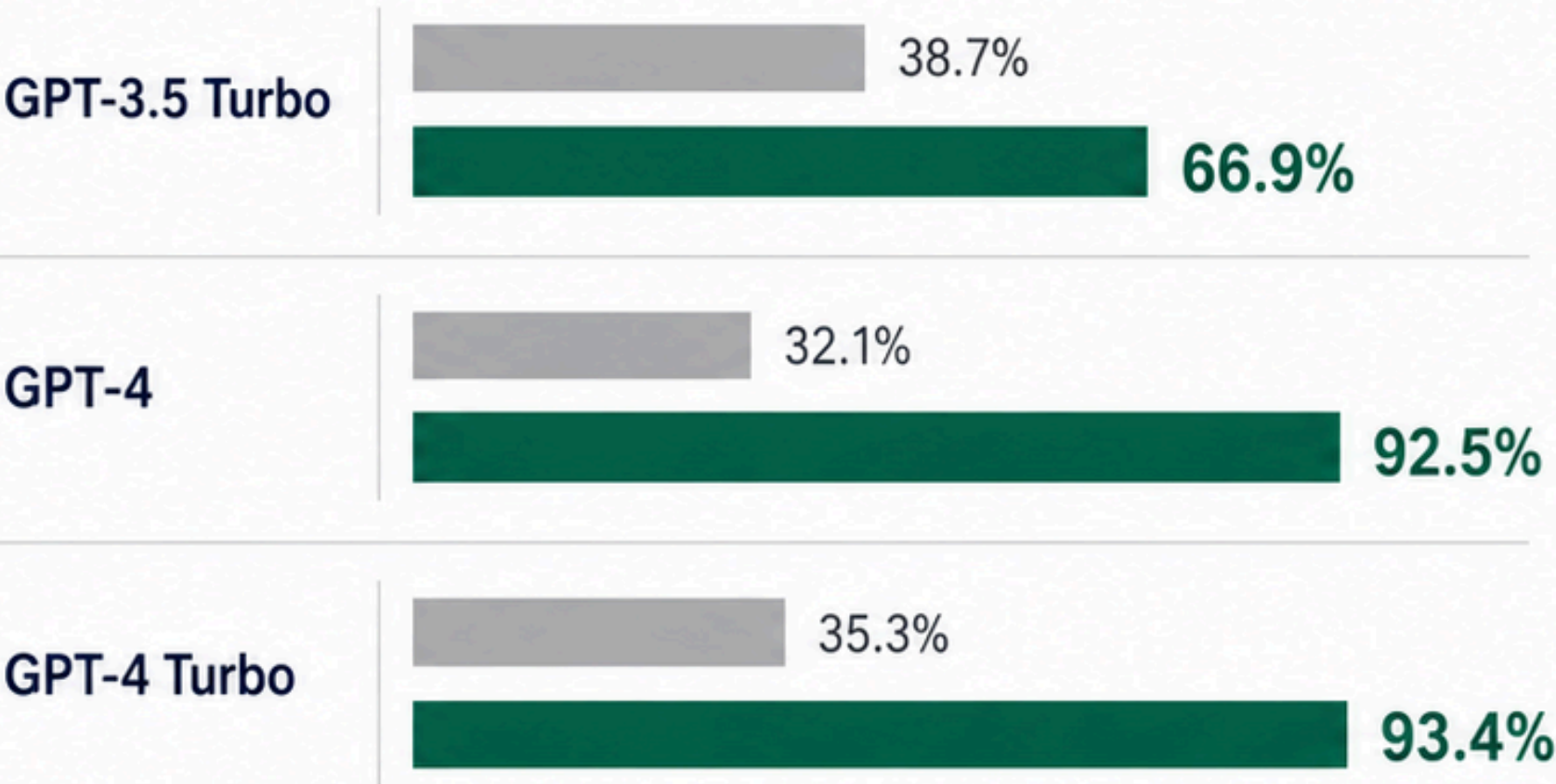


The answer is buried in an earlier session.
Fixed context forgets; **MemGPT retrieves.**



Accuracy on DMR

Fixed context (baseline) + MemGPT



Takeaway: MemGPT consistently improves deep memory retrieval, but the gain still depends on the base model's *memory-tool* use.

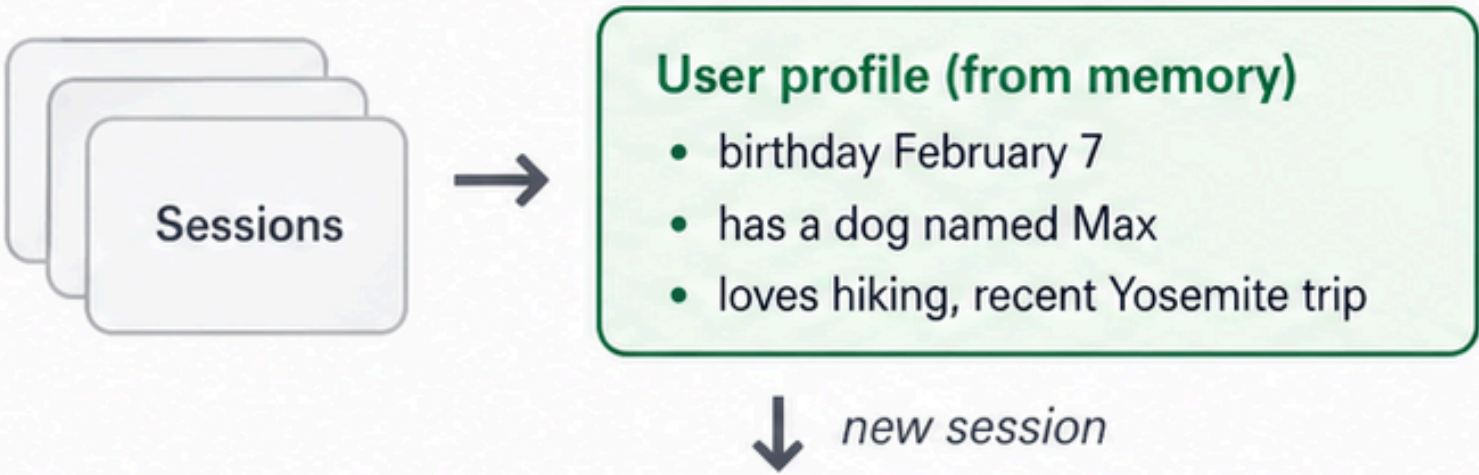
MemGPT improves persona-consistent conversation openers

Conversation opener evaluates whether an agent can start a new session with an opener that reflects user memory accumulated across sessions.



Conversation Opener task

Can the agent start a new session with an opener that reflects what it knows about the user?



Agent's opener

"Hey! How's Max doing? Did the hike last weekend live up to expectations?"



Memory enables the opener to reflect the user's persona.
More memory-aware, more personalized.



Conversation opener performance

Method	SIM-1	SIM-3	SIM-H
Human	0.800	0.800	1.000
GPT-3.5 Turbo	0.830	0.812	0.817
GPT-4	0.868	0.843	0.773
GPT-4 Turbo	0.857	0.828	0.767

Similarity to gold persona (SIM-1)



MemGPT can produce openers that are more persona-aligned than human-written openers.



Takeaway: Memory helps the agent produce more persona-consistent openers, but quality still depends on the underlying base model.

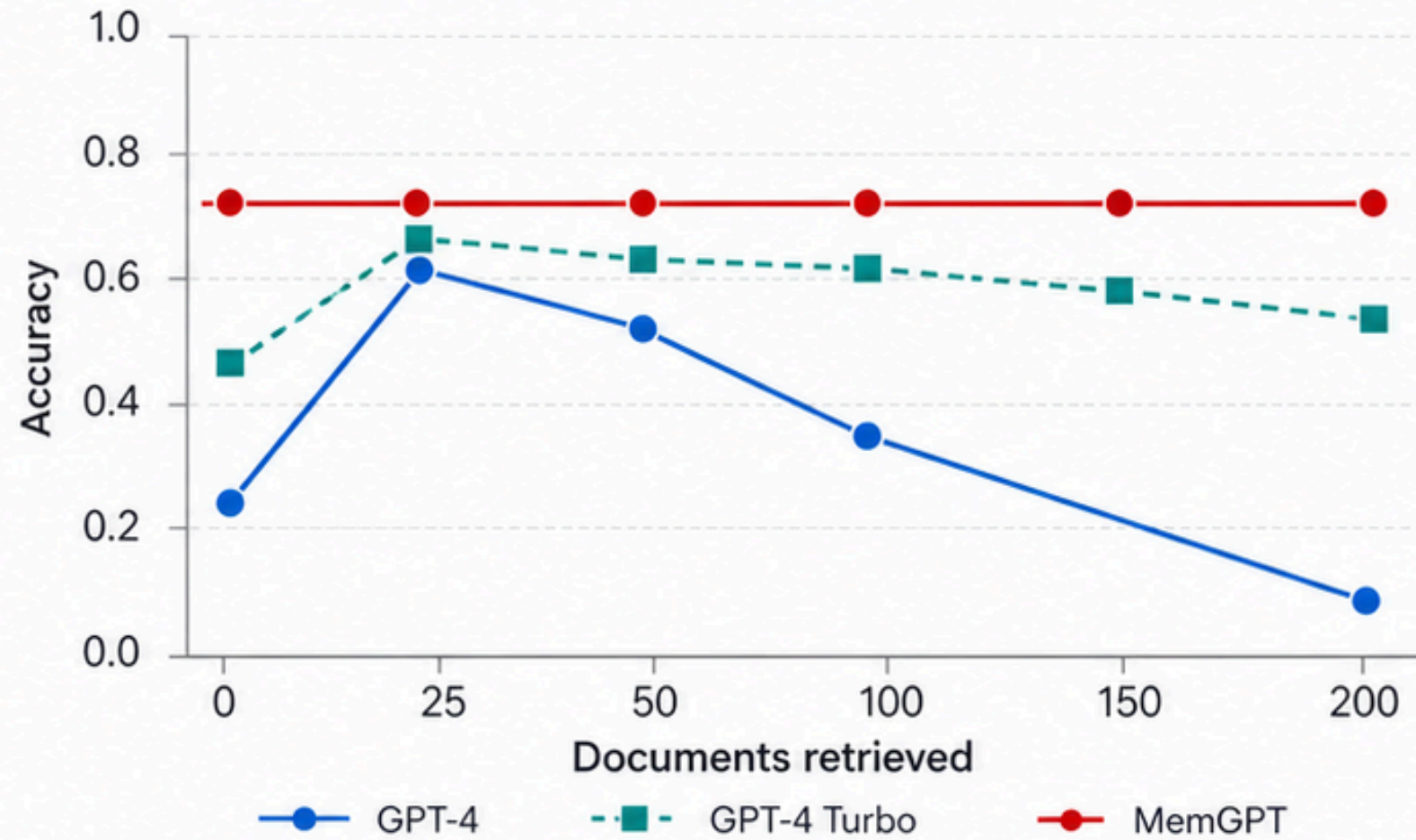
Iterative search helps document QA survive long contexts

MemGPT can page through archival storage instead of relying on a single fixed context window.



Document QA performance

Accuracy vs. documents retrieved



As retrieval depth grows, fixed-context accuracy drops, while **MemGPT remains stable** through iterative search.



Example: search → paginate → answer

Archive storage upload complete

Who won the first Nobel Prize in physics?

1

```
archival_storage.search("nobel physics")
```

Page 1 results (showing 10 of 124):

"The **Nobel Prizes**, beginning in 1901, and the ...
"This award is administered by the **Nobel** Foundation ..."

2

```
archival_storage.search("nobel physics", page=2)
```

Page 2 results (showing 10 of 124):

"The **Nobel Prize in Physics** is a yearly award given ..."
"The 1901 **Nobel in physics** was awarded to
Wilhelm Conrad Röntgen ..."



Wilhelm Conrad Röntgen

Pagination lets the model keep searching beyond the first page of results.



Takeaway: MemGPT improves document QA by allowing repeated search over external memory, but success still depends on good queries and reliable tool use.

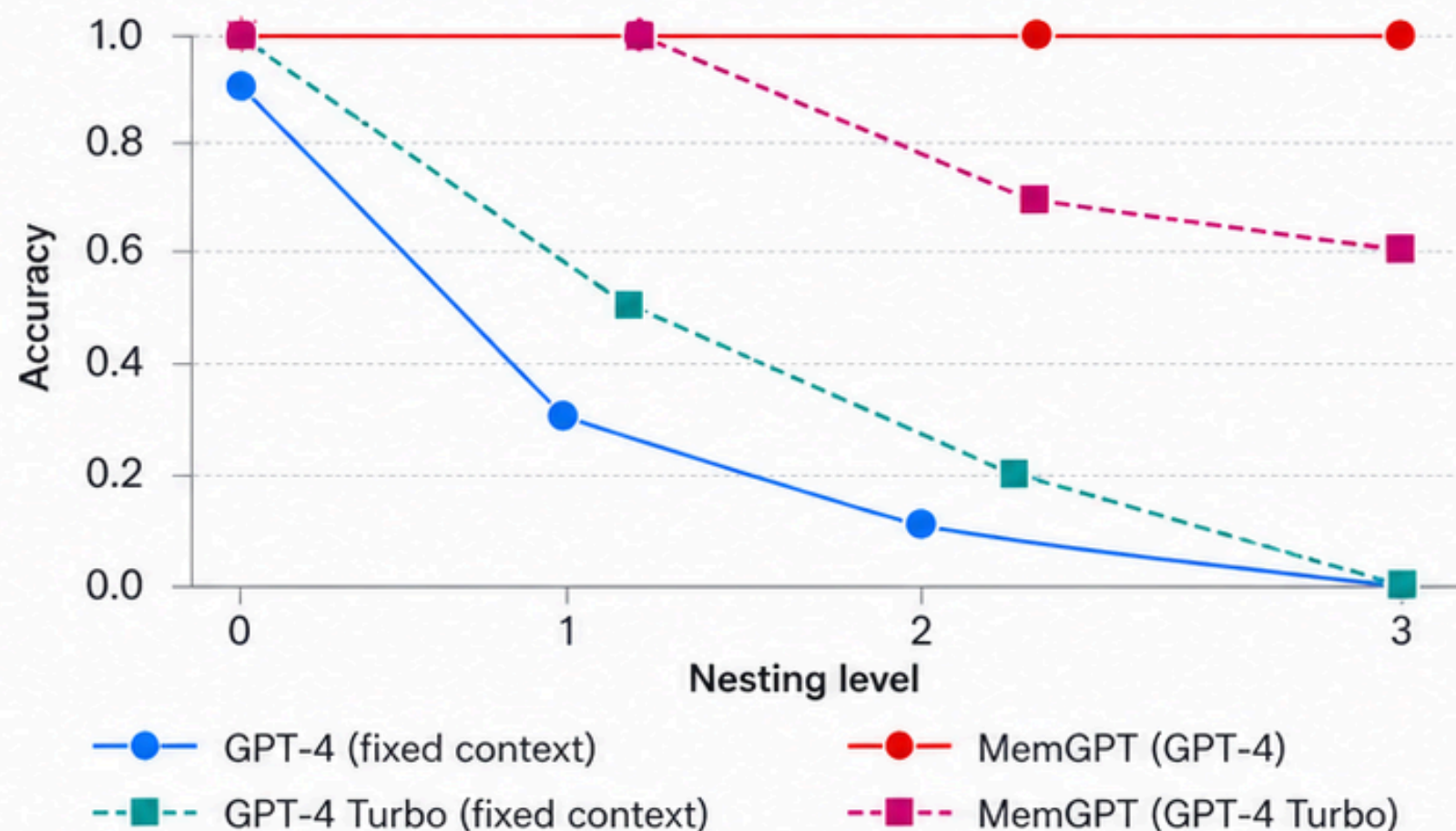
Nested KV exposes fragile multi-hop retrieval

Each retrieved value becomes the next query, so one wrong hop can break the chain.



Nested KV performance

Accuracy vs. nesting level



As chaining depth grows, performance becomes limited by reliable tool use and state tracking—not memory capacity alone.



Example: follow the key chain

Find the value for key 831...ea5

1

```
archival_storage.search("831...ea5")
```

Result: Key: 831...ea5, Value: 5b8...4c3

2

```
archival_storage.search("5b8...4c3")
```

Result: Key: 5b8...4c3, Value: f37...617

3

```
archival_storage.search("f37...617")
```

Result: Key: f37...617



Final answer: f37...617

Each retrieved value becomes the next query.



Takeaway: Nested KV tests more than memory access—it requires iterative tool use, exact copying, and chain-state tracking.

Discussion 3

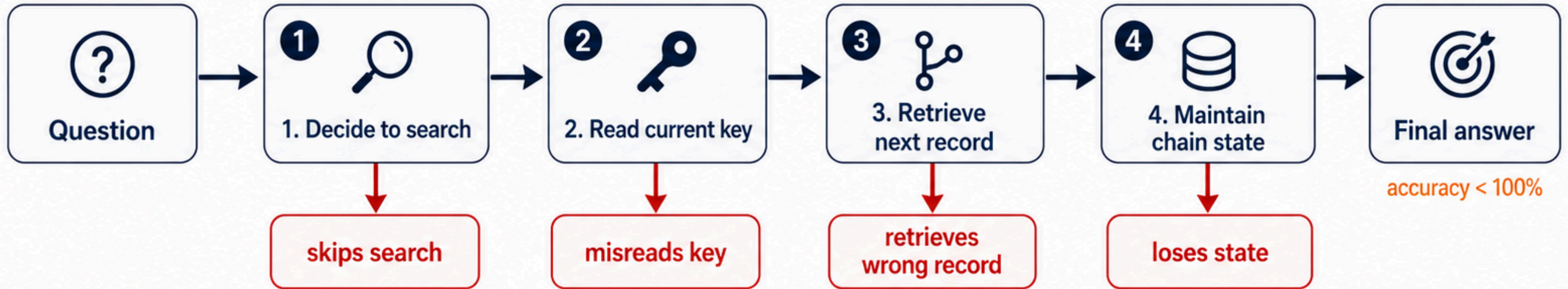
“Why doesn’t MemGPT reach 100% on Nested KV?”

Where do errors accumulate across hops?



Why MemGPT still falls short on Nested KV

Nested KV requires repeated retrieval and reliable state tracking across hops.



Compounding error

$$\text{Success} \approx p^k$$

p = per-hop reliability, k = number of hops



Memory $\neq$ execution

Memory access alone does not solve multi-hop chaining.



Nested KV stresses composition

It requires iterative retrieval, exact copying, and state tracking.



The bottleneck shifted

MemGPT helps with context limits, but execution remains brittle.

Discussion 4

“What are the core limitations of MemGPT?”

Where does the architecture still fall short?



Limits of MemGPT

What MemGPT solves — and what it leaves open

1 Retrieval is brittle

Multi-hop search errors compound.
Missing one step can break the whole chain.



multi-hop synthesis degrades fast

2 Memory is flat

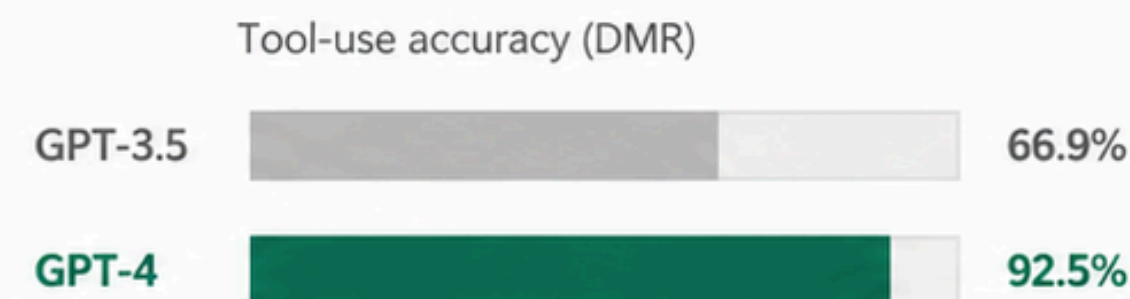
Storage records messages, but does not automatically build higher-level abstractions.

[10/12]	"i love six flags"
[10/15]	"we hiked yosemite"
[10/22]	"love national parks"
[11/03]	"tahoe trip planned"

raw logs, no built-in abstraction

3 Control depends on the base LLM

MemGPT works only if the base model can decide when and how to call tools.



performance scales with base model tool use



Core takeaway

MemGPT improves memory access, but its memory remains flat and its success depends on reliable multi-step retrieval and tool use.

Beyond MemGPT: Toward Structured Dynamic Memory

MemTree is one possible direction — not the only answer.

MemGPT Tabular dynamic	RAPTOR Structured static	MemTree Dynamic hierarchical
		
 Structure: Flat rows	 Structure: Hierarchical tree/graph	 Structure: Hierarchical tree
 Update: Fast, on-the-fly	 Update: Offline rebuild	 Update: Real-time, adaptive
 Limitation: Weak at synthesizing scattered evidence	 Limitation: Slow to refresh	 Promise: Better balance of abstraction and update speed



Core takeaway

Future agent memory should support **hierarchical abstraction** and **dynamic updates** during inference.

A possible next step, not a settled standard.

Final Takeaway

What MemGPT changes — and what remains open

1 Remember



Move durable facts out of limited context.

write important memory

2 Retrieve



Bring back relevant memory when needed.

search and reinsert

3 Revise



Update stale memory instead of keeping outdated facts.

replace, not just append



Core takeaway

MemGPT shows that memory should be actively managed, not treated as a larger prompt.

But robust agent memory still needs structured abstraction and reliable multi-step tool use.

A useful step toward LLMs that manage memory like an operating system.