Project Pages

Paper

Why HeMeR?

Embodied agents act on a **memory context** composed from declarative, procedural, and working memory. Each step changes only a few memory contents, yet conventional serving treats the updated context as a new input and **recomputes the full KV cache**, even when most of it is still valid.

Problem

- Memory insertions, evictions, and reorderings shift token positions and break naive KV reuse
- Procedural adapter (LoRA) switches make previously cached KV states incompatible
- Existing KV reuse targets prefixes, segments, or adapters in isolation, not heterogeneous memory updates

Core Insight

Declarative, procedural, and working memory serve different roles, yet they all land in one place: **the KV cache**. Anchor each memory content to its KV range, and every memory update becomes a **local cache edit**.

HeMeR

HeMeR indexes the memory context as **schema-typed chunks**, diffs consecutive contexts into a **reconciliation plan**, reuses compatible KV ranges with RoPE remapping for shifted chunks, and **recomputes only** the ranges whose content, dependency, or adapter changed.

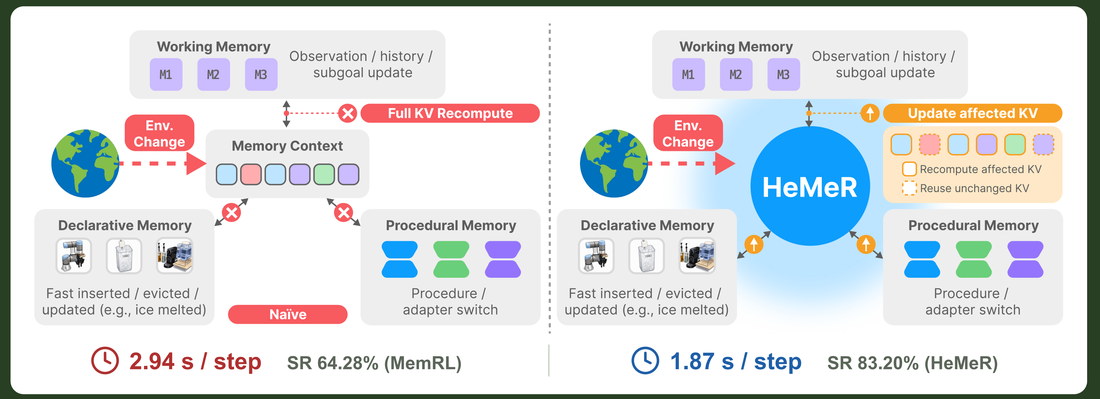
Contributions

- Memory-aware KV reuse for embodied agents
- Memory schema: typed chunks ↔ KV ranges
- Chunk-level KV reconciliation + RoPE remap
- Adapter switches under the same schema

HeMeR: Heterogeneous Memory Reconciliation

for Embodied Agents via Structured KV Reuse

Saehun Chun, Wonje Choi, Jinwoo Jang, Taeyoon Kwack, Honguk Woo*

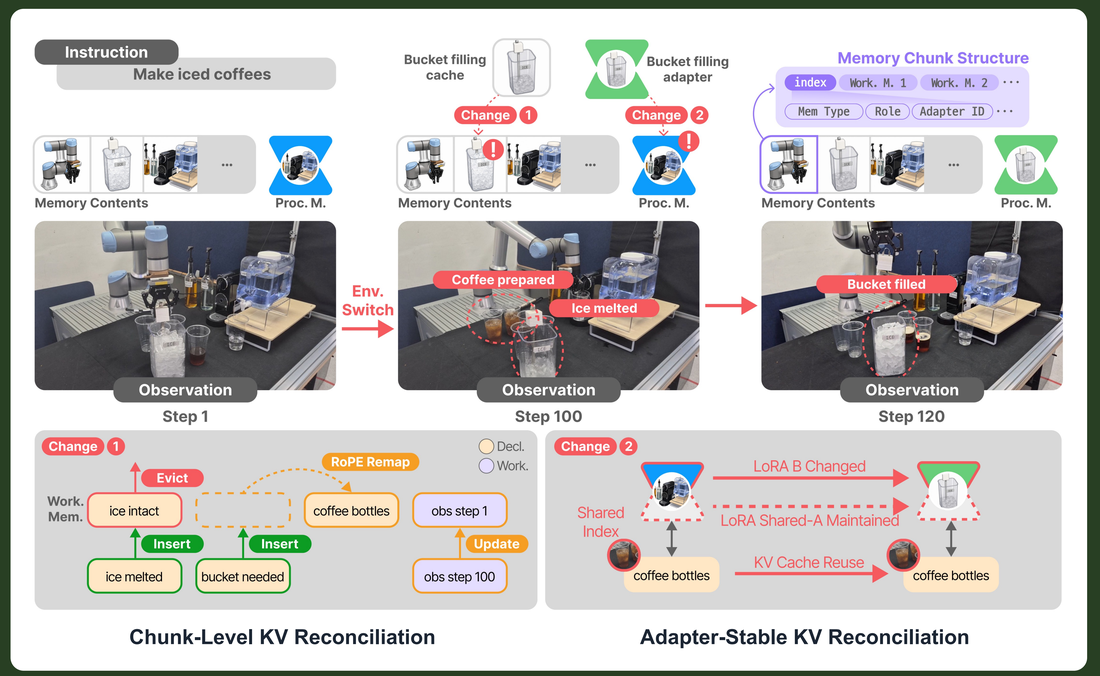


Anchor — Memory contents as typed chunks bound to KV ranges
→ schema records type, role, dependency, and adapter

Diff — Consecutive memory contexts into a reconciliation plan
→ each chunk is inserted, evicted, reordered, or reused

Remap — Shifted but unchanged chunks via RoPE rotation
→ cached keys stay valid without recomputation

Adapt — KV compatibility across procedural adapter switches
→ recompute only the procedural chunk and its dependents



Chunk-Level KV Reconciliation

Adapter-Stable KV Reconciliation

SUNG KYUN KWAN UNIVERSITY(SKKU)

NEURAL INFORMATION PROCESSING SYSTEMS

Experiment Setting & Result

Experimental Benchmarks.
ALFWorld, EmbodiedBench Habitat, and real-world UR7e tabletop manipulation.

Scenarios.
Unseen compositional tasks where no single memory covers the required procedure - room combination; Kitchen-Café and Beverage Station in the real world.

Backbones.
Llama-3.2-3B-Instruct (ALFWorld)
Qwen3-VL-4B-Instruct (Habitat, real world)
Rank-8 LoRA procedural adapters

Metrics.
Accuracy: SR (Success Rate)
Efficiency: Latency (s/step), TTFT
Cache: KV reuse ratio

	ALFWorld		Habitat		Real-world	
Method	SR ↑	Lat. ↓	SR ↑	Lat. ↓	SR ↑	Lat. ↓
ReAct	47.14	0.88	52.00	2.41	–	–
Reflexion	44.29	0.98	50.50	2.38	–	–
MemRL	52.86	0.90	61.81	2.68	64.28	2.94
CacheBlend	48.39	0.48	56.38	1.97	51.21	2.14
KEEP	51.81	0.46	59.40	2.09	63.40	2.43
LRAgent	46.43	0.52	57.50	2.03	53.43	2.21
HeMeR (ours)	71.43	0.18	76.28	1.82	83.20	1.87

SR (%), latency (s/step). 5 seeds in simulation, 10 physical trials in the real world.

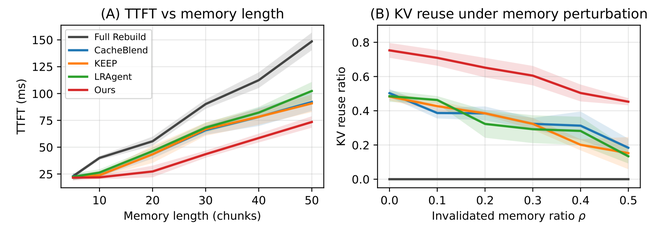
+17.32%p — Higher average SR than MemRL

−0.88s — Faster average latency per step than MemRL

1st — SR and latency in all 3 environments

Higher SR comes from reusing only the KV states that are still schema-compatible.

Cost Efficiency under Memory Updates.



Poster message

- Do not rebuild the full KV cache when only a few memory contents change.
- Anchor memories as typed chunks, reuse what is still valid, and reconcile only what changed.