Learning to Share:
Selective Memory for Efficient Parallel Agentic Systems

Joseph Fioresi, Parth Parag Kulkarni*, Ashmal Vayani*, Song Wang, Mubarak Shah
Institute of Artificial Intelligence (IAI) @ University of Central Florida

Institute of Artificial Intelligence
UNIVERSITY OF CENTRAL FLORIDA

ICML International Conference On Machine Learning

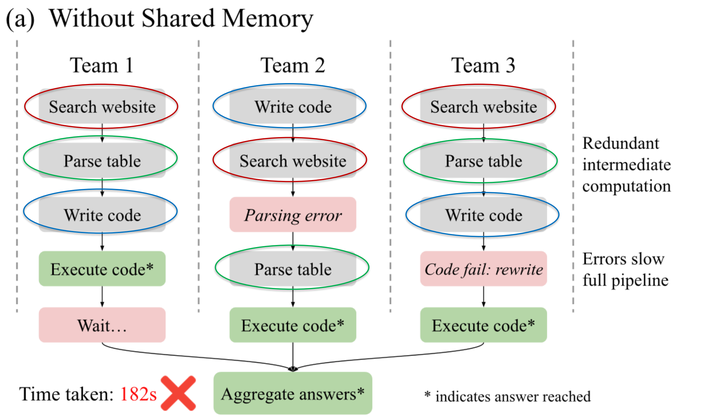
Motivation – Parallel Execution

- Self-consistency improves performance in agentic tasks
- However, teams are blind to each other; no sharing progress
- Overlapping computation, isolated errors slow the full system

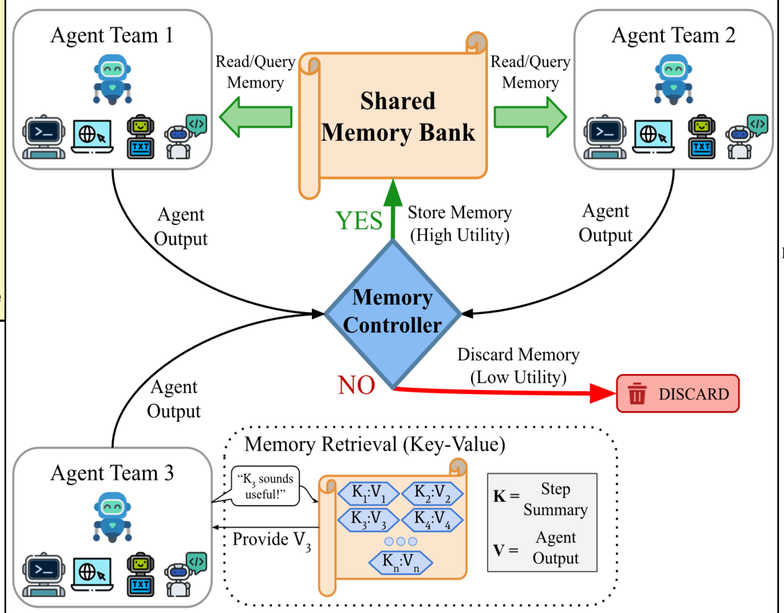
Contributions

- Propose memory to reduce redundant parallel computation
- Design RL-based strategy with usage-aware reward shaping
- Show selective sharing improves efficiency AND performance

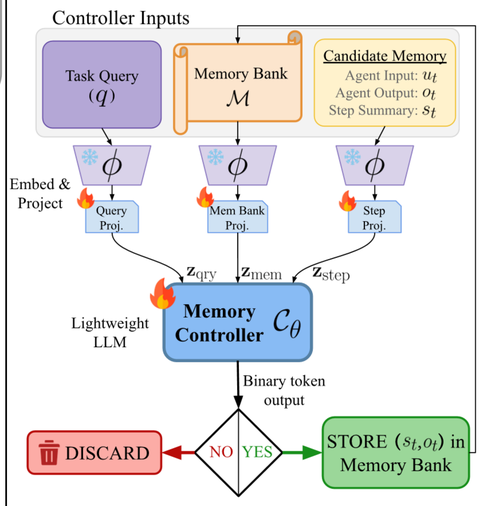
Parallel Agentic Framework



Workflow Diagram



Memory Controller Design/Inputs



Reward Modeling + Usage-Aware Shaping

Base reward: $R(\tau) \in [0,1]$ (benchmark score for trajectory τ)

Reward function: $R(\tau) = \underbrace{R_{\text{agg}}(\tau)}_{\text{Reward of final aggregated answer}} + \underbrace{\lambda_{\text{first}} R_{\text{first}}(\tau)}_{\text{Scaled reward of first completing team's answer}}$

Per-trajectory advantage score: $A^{(i)}_{\text{base}} = \dfrac{R(\tau^{(i)}) - \mu_R}{\sigma_R + \epsilon}$

Shaped step-wise advantage function:

$$\hat{A}^{(i)}_t = A^{(i)}_{\text{base}} + \beta \cdot \mathbb{I}\left(t \in \mathcal{U}^{(i)} \wedge R(\tau^{(i)}) > 0\right)$$

- Increases reward for saving retrieved memories

Advantage applied in **policy loss**:

$$L^{\text{policy}}_t(\theta) = -\log \pi_\theta(z_t \mid c_t) \cdot \hat{A}_t$$

Sparsity penalty to avoid degenerate "always YES" policy:

$$L^{\text{sparse}}_t(\theta) = \pi_\theta(z_t = \text{YES} \mid c_t)$$

- Admitting all steps increases usage reward (reward hacking)

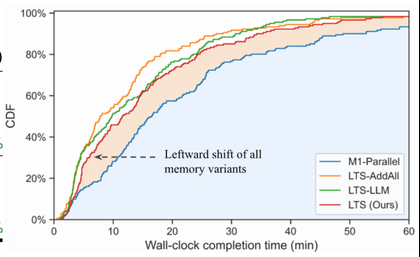
Complete optimization objective:

$$\mathcal{L}(\theta) = \mathbb{E}_{\tau \sim \pi_\theta}\left[\sum_{t=1}^{T}\left(L^{\text{policy}}_t(\theta) + \lambda_{\text{sparse}} L^{\text{sparse}}_t(\theta)\right)\right]$$

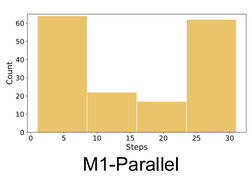
Results

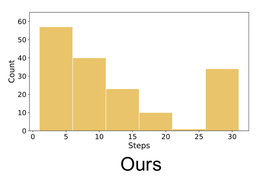
Method	Shared Memory	Model	AssistantBench Easy Acc.	Med. Acc.	Hard Acc.	All Acc.	Execution Runtime (↓)	GAIA Lvl 1 Acc.	Lvl 2 Acc.	Lvl 3 Acc.	All Acc.	Execution Runtime (↓)
MagenticOne [1]	✗	Qwen3 (32B)	53.1	17.4	8.5	13.4	1084s	41.5	16.3	3.8	22.5	758s
M1-Parallel [2]	✗		52.6	21.4	8.5	14.7	2239s	43.4	18.6	7.7	24.8	1569s
LTS (Ours)	✓		**65.8**	**25.2**	**14.4**	**20.3**	**1479s** ↓40.8%	**47.2**	**25.6**	**11.8**	**30.4**	**892s** ↓55.0%
MagenticOne [1]	✗	GPT-5.1	43.6	31.5	15.3	21.7	724s	48.1	37.3	16.7	37.5	815s
M1-Parallel [2]	✗		57.3	**35.2**	16.0	24.0	1389s	60.4	46.5	**26.9**	47.9	1005s
LTS (Ours)	✓		**61.0**	35.1	**20.0**	**26.7**	**882s** ↓44.6%	**62.3**	**47.7**	**26.9**	**49.1**	**781s** ↓25.1%

Runtime CDF

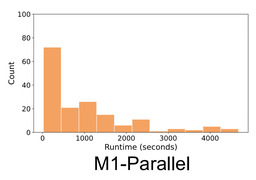


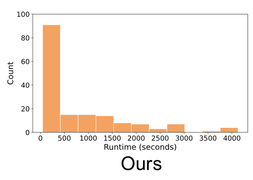
Steps Taken / Task Distribution





Runtime / Task Distribution





Shared Memory Improves Average Runtime!

Component Ablation:

Variant	GAIA Task Acc. (%)	Runtime (s)	Memories Saved (%)
LTS (Ours)	**49.1**	**792s**	**84.9**
w/o Usage-Aware Shaping	47.7	925s	89.5
w/o Sparsity Loss ($\lambda_{\text{sparse}} = 0$)	47.4	983s	98.9

References

[1] Fourney, Adam, et al. "Magentic-one: A generalist multi-agent system for solving complex tasks." arXiv preprint arXiv:2411.04468 (2024).

[2] Zhang, Enhao, et al. "Optimizing sequential multi-step tasks with parallel llm agents." arXiv preprint arXiv:2507.08944 (2025).

Webpage/Code: