ArtMem: Adaptive Migration in Reinforcement Learning-Enabled Tiered Memory

Xinyue Yi, Hongchao Du, Yu Wang, Jie Zhang, Qiao Li, Chun Jason Xue

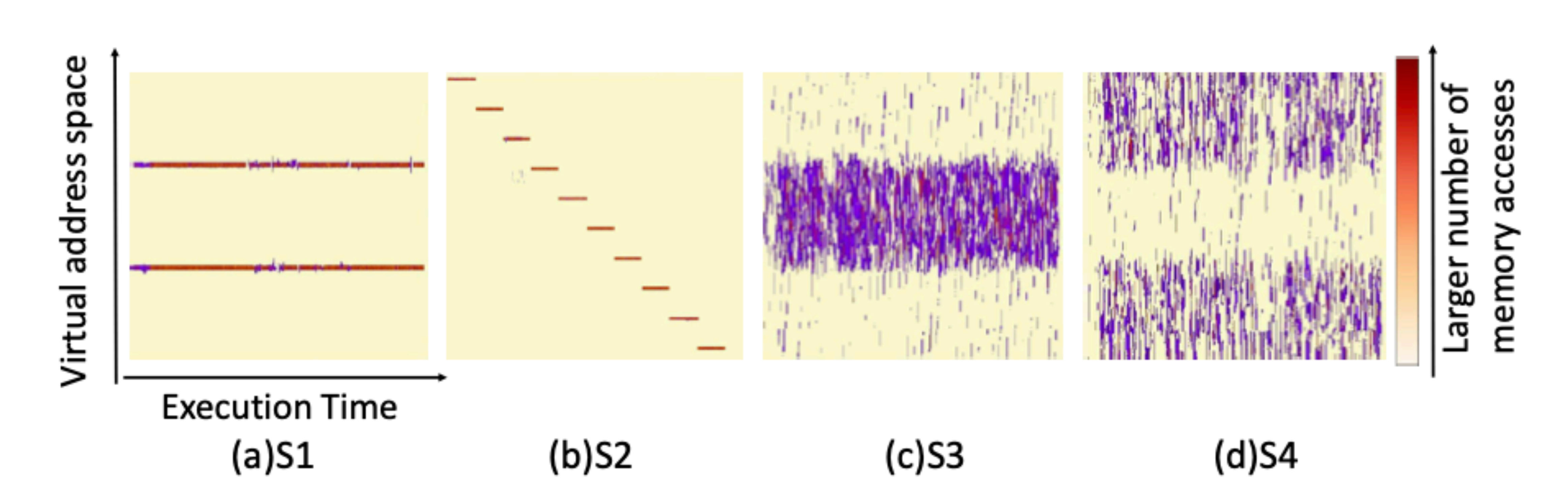
Kshitij Rana / January 29th 2025

Why do we need ArtMem?

- Existing systems cannot adapt to diverse workloads
- Migration strategies ignore fast tier utilization feedback
- Static migration scope limits tiered memory potential

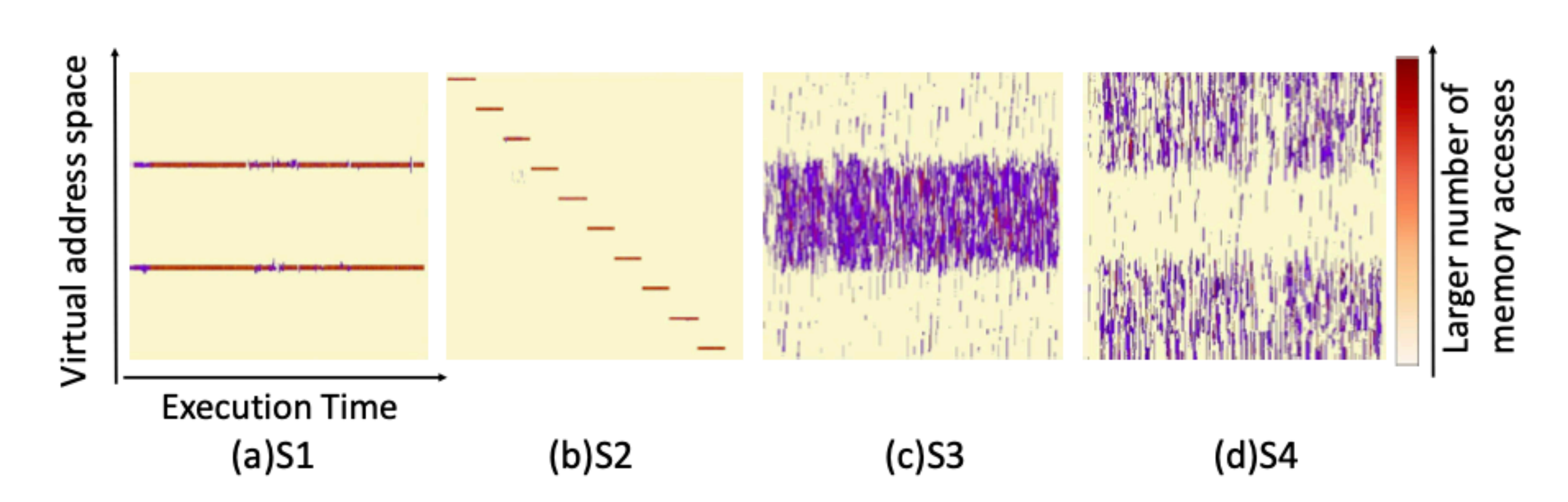
Why do we need ArtMem?

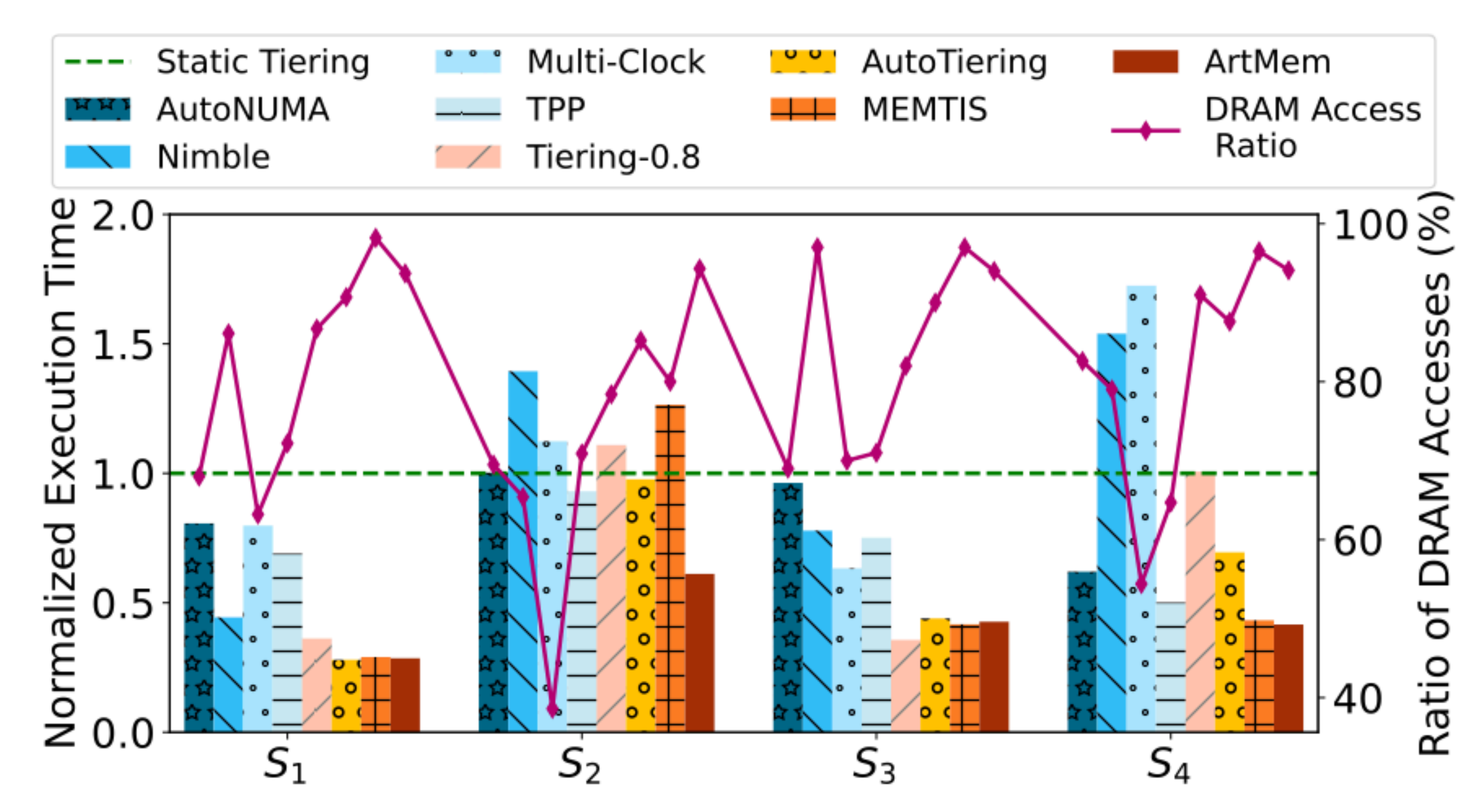
Observation 1: Existing systems cannot adapt to diverse workloads



Why do we need ArtMem?

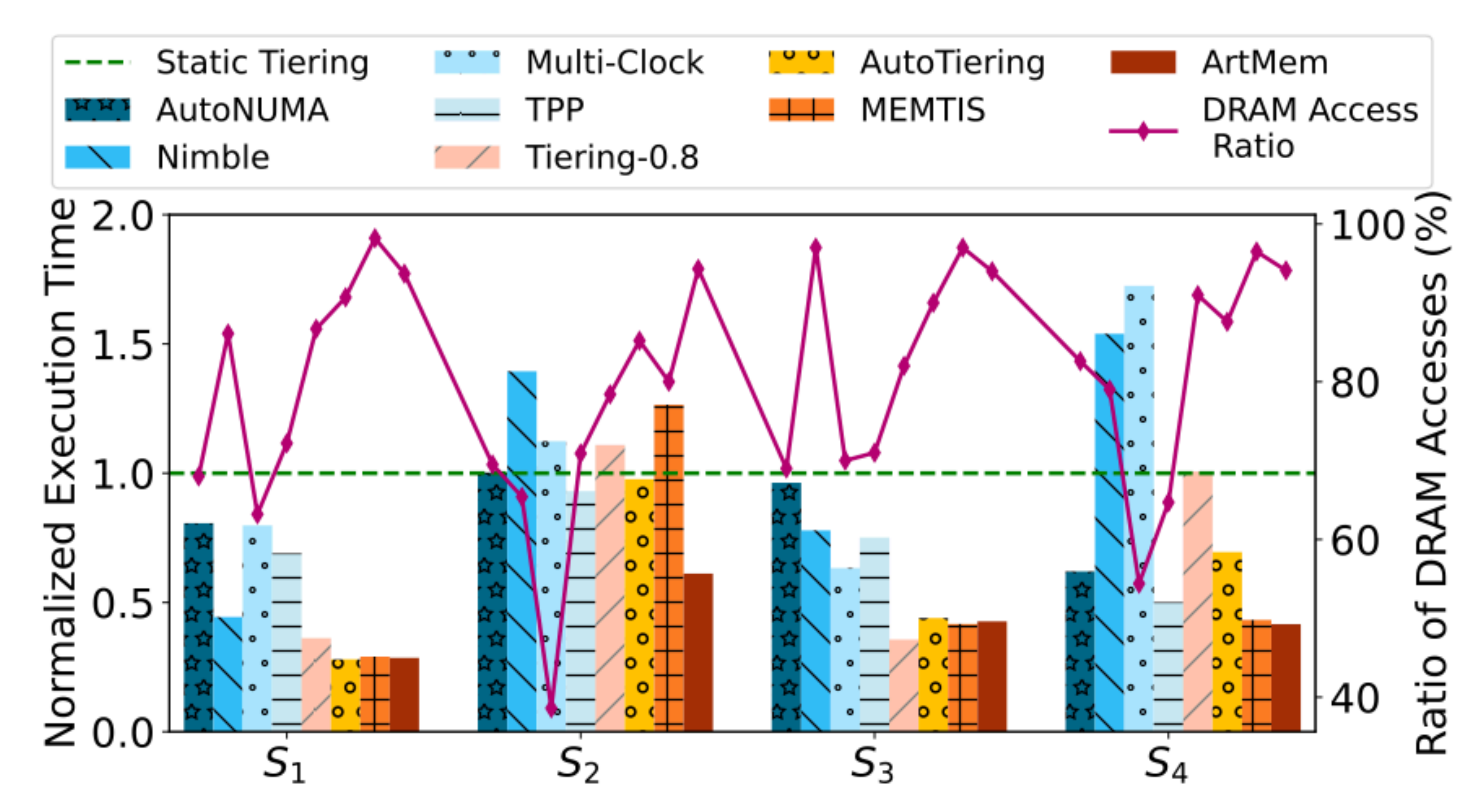
Observation 1: Existing systems cannot adapt to diverse workloads





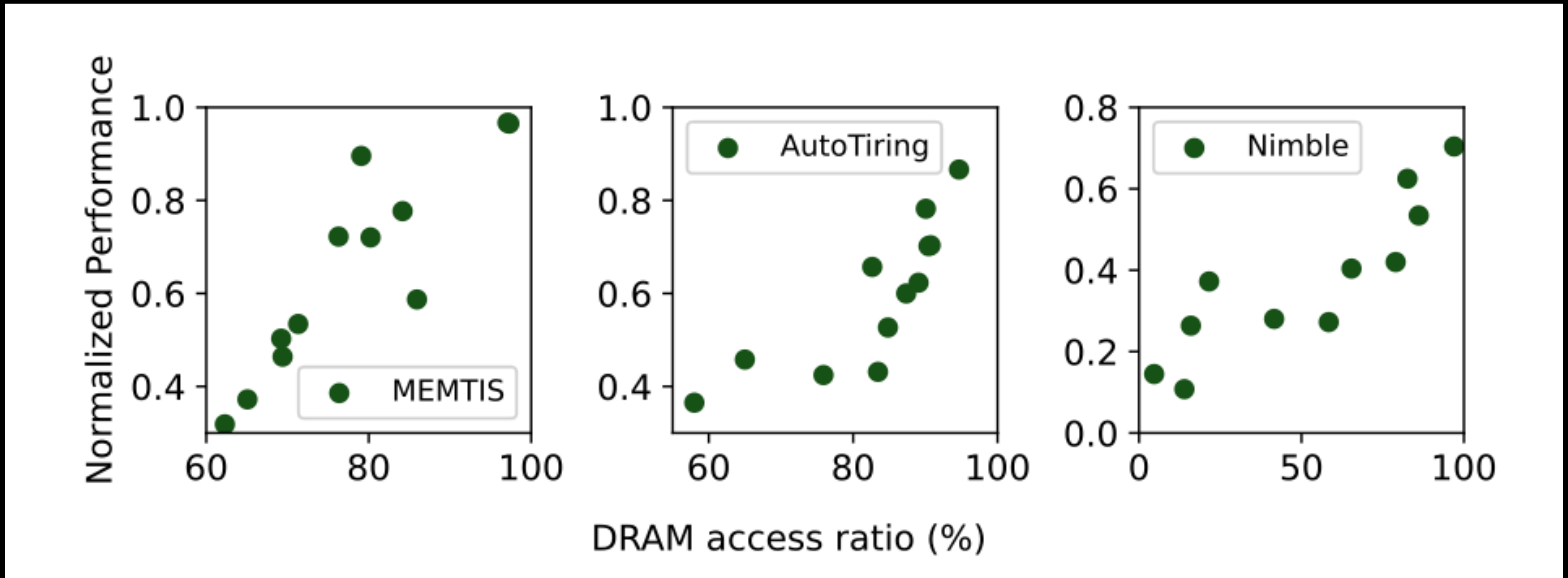
Why do we need ArtMem?

Observation 2: Migration strategies ignore fast tier utilization feedback



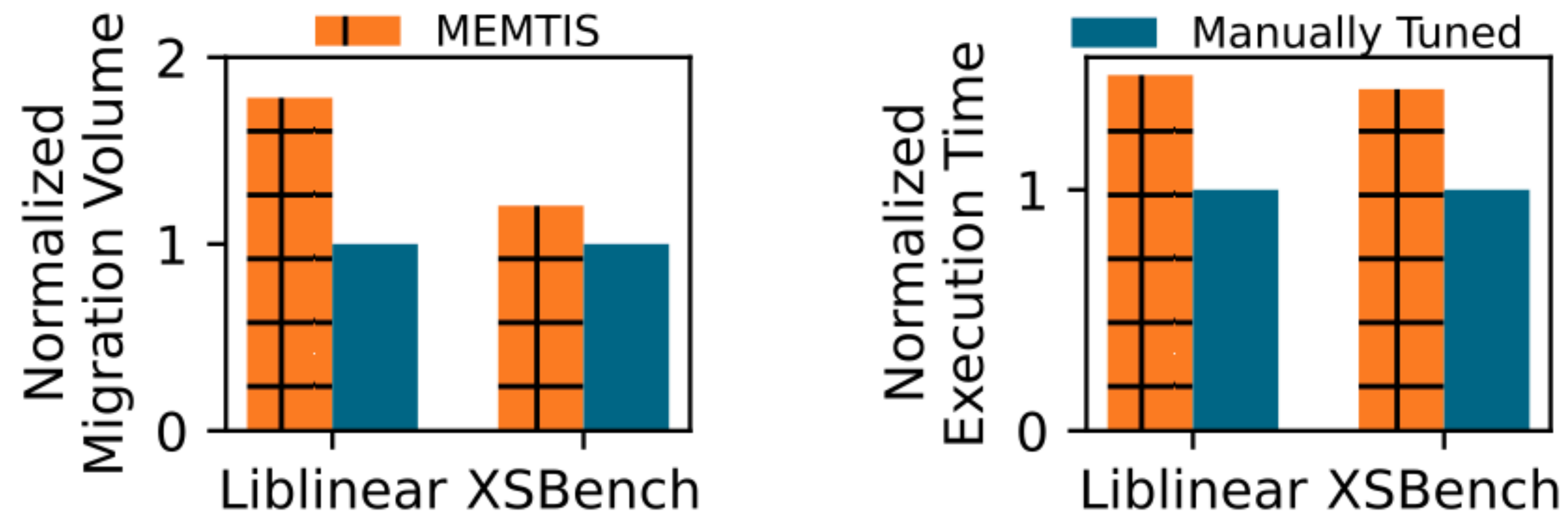
Why do we need ArtMem?

Observation 2: Migration strategies ignore fast tier utilization feedback



Why do we need ArtMem?

Observation 3: Static migration scope limits tiered memory potential

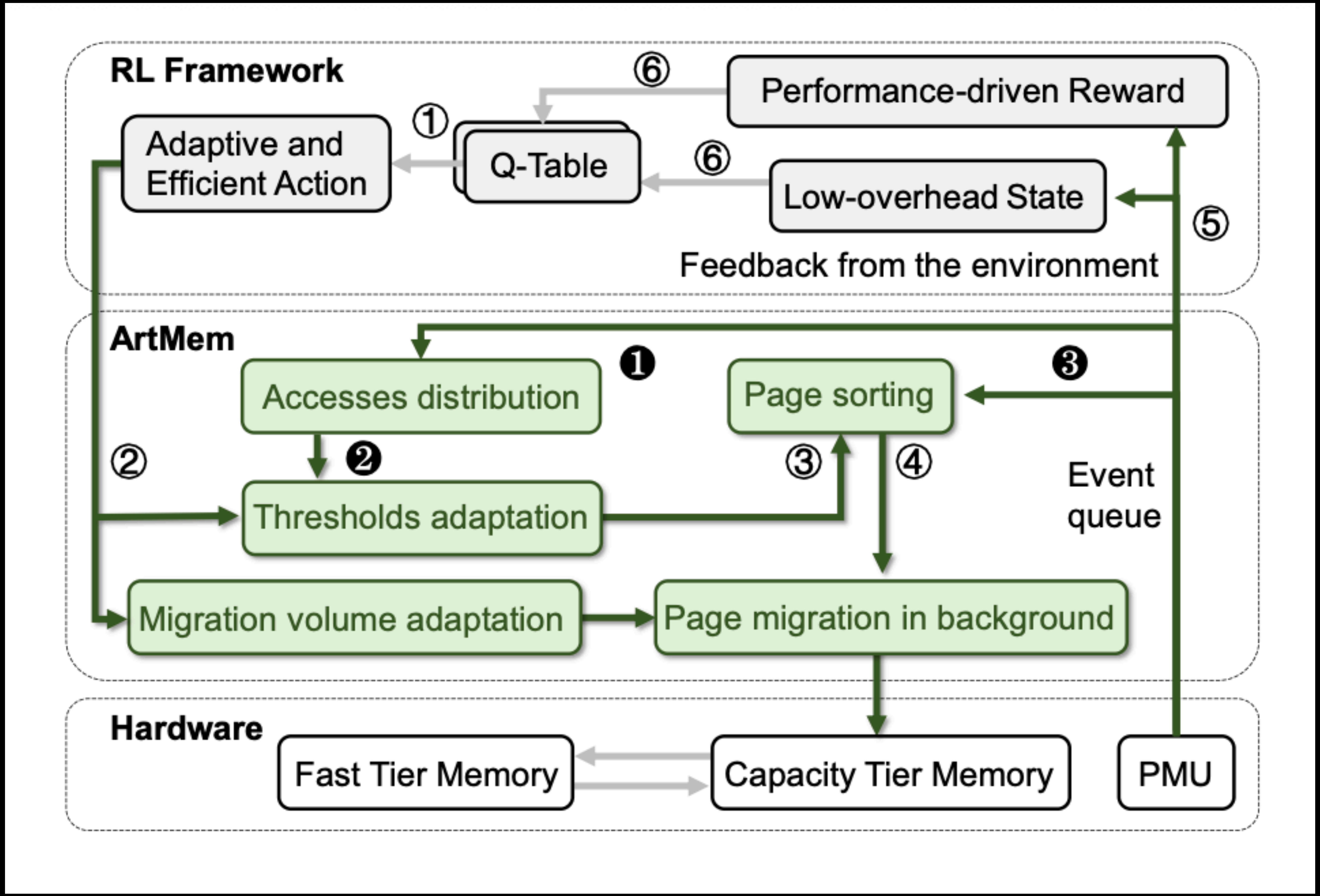


(a) Migration volume. (b) Performance.

Why do we need ArtMem?

- Memory access patterns are diverse and complex
- It is hard to manually adjust heuristics for real applications
- Adopting a learning based strategy can help
- RL is better suited for this problem than other ML techniques?

ArtMem Design



ArtMem Design
RL Framework

- Q-learning based learning approach
- Q-table maintains a list of states and actions that estimates how good it is to take an action in a particular state
- Minimize computational overhead — states and actions are defined at the system level and not page level

ArtMem Design

RL Framework: State Design

- Based on observation 2, ArtMem uses the ratio of DRAM accesses as the state
- It is descretized into k+1 states [0,..,k]

$$\tau = \left\lfloor \frac{DRAM_{access} \times k}{DRAM_{access} + PM_{access}} \right\rfloor$$

ArtMem Design

RL Framework: Action Design

- Based on observation 3, ArtMem adopts migration scope as the action target
- Depends on the **migration number** and the **hotness threshold**
- Two separate tables for each kind of action (<10 KB each)

ArtMem Design

RL Framework: Reward Design

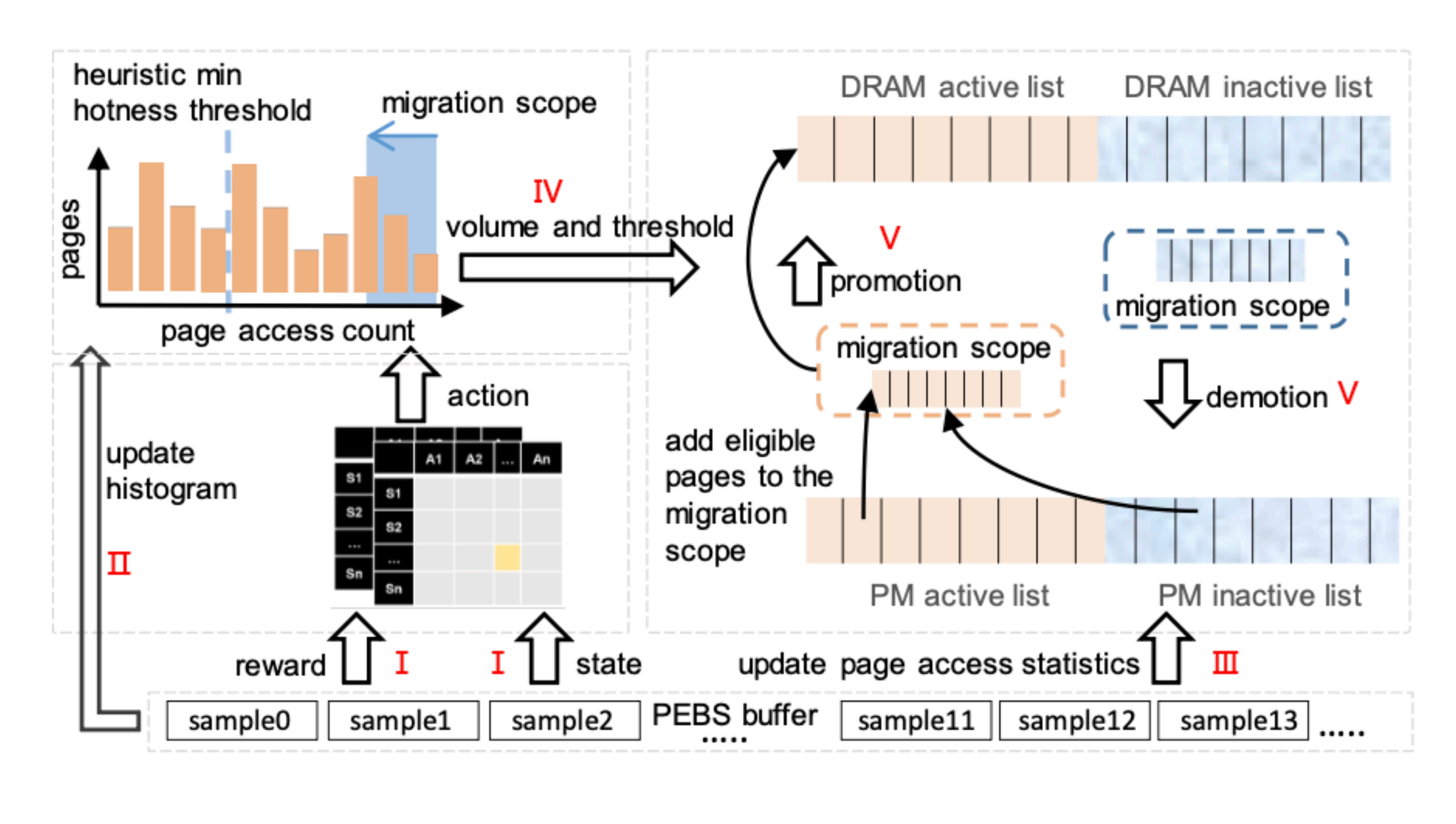
- Rewards are a feedback mechanism through which learning happens
- Optimal behavior is to migrate an appropriate number of pages, maintaining a high DRAM access ratio
- τi is the state from the previous equation
- β is the desired fast memory access tier ratio
- λ indicates whether a migration occurred in the last period to reward only the first item

$$\tau_i - \beta + \lambda(\tau_i - \tau_{i-1})$$

ArtMem Design

RL Framework: Workflow

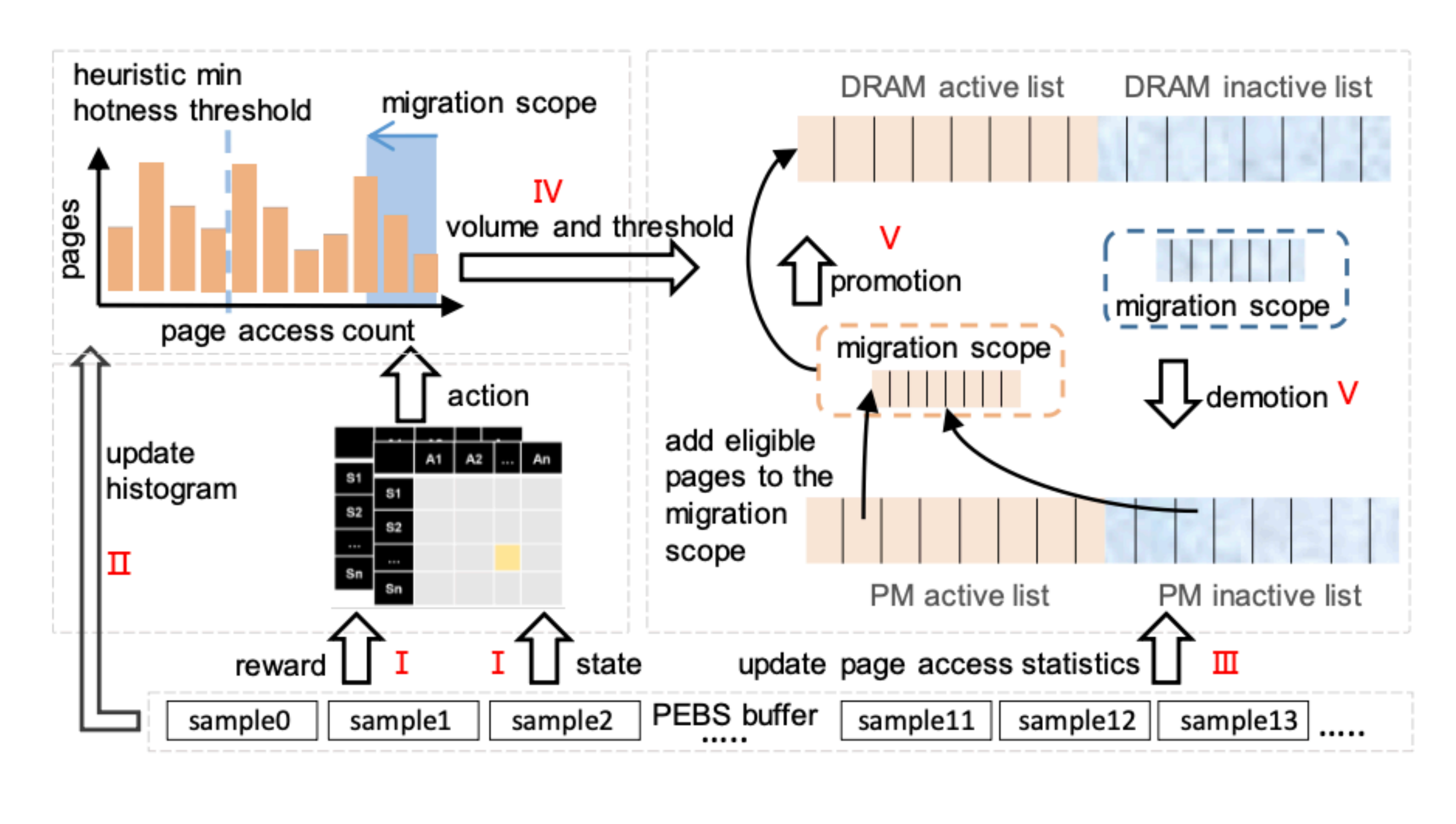
- I: Sampled data is used to learn the actions based on state and reward
- II & III: Sampled data is also used to maintain the distribution of accesses and the statistic of each sampled page



ArtMem Design

RL Framework: Workflow

- IV: A migration scope is determined based on the action
- V: Migration is performed



ArtMem Design

Implementation

- Use 2 MB hugepages to avoid address translation overhead
- In kernel sampling and migration threads for sampling events and migrating pages
- RL implementation in user space — uses pseudo filesystem to interact with the sampling threads

Evaluation

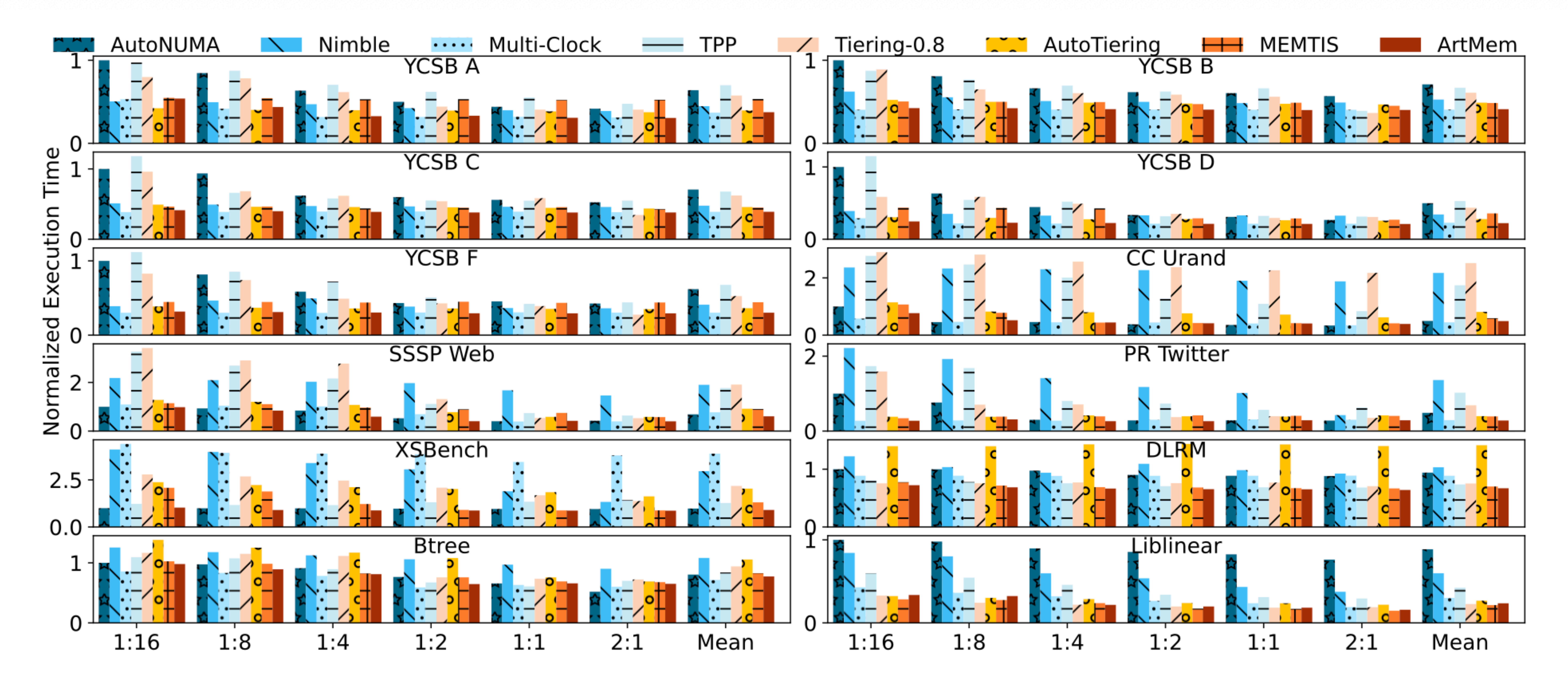
Main Results

- 12 different workloads
- 7 baselines — AutoNUMA, Nimble, Multi-Clock, TPP, Tiering-0.8, AutoTiering, MEMTIS
- Execution time of AutoNuma 1:16 is the baseline (others normalized to this value)

Workloads	Memory footprint	Descriptions
YCSB [16]	32G	In-Memory Database
CC [11, 55]	69G	Connected Components
SSSP [34, 37]	64G	Single Source Shortest Path
PR [13, 28]	25G	PageRank
XSBench [57]	69G	HPC Workloads
DLRM [24, 38]	72G	Deep Learning Recommendation Model
Btree [6]	24G	In-Memory Index Lookup
Liblinear [31]	68G	Machine Learning

Evaluation

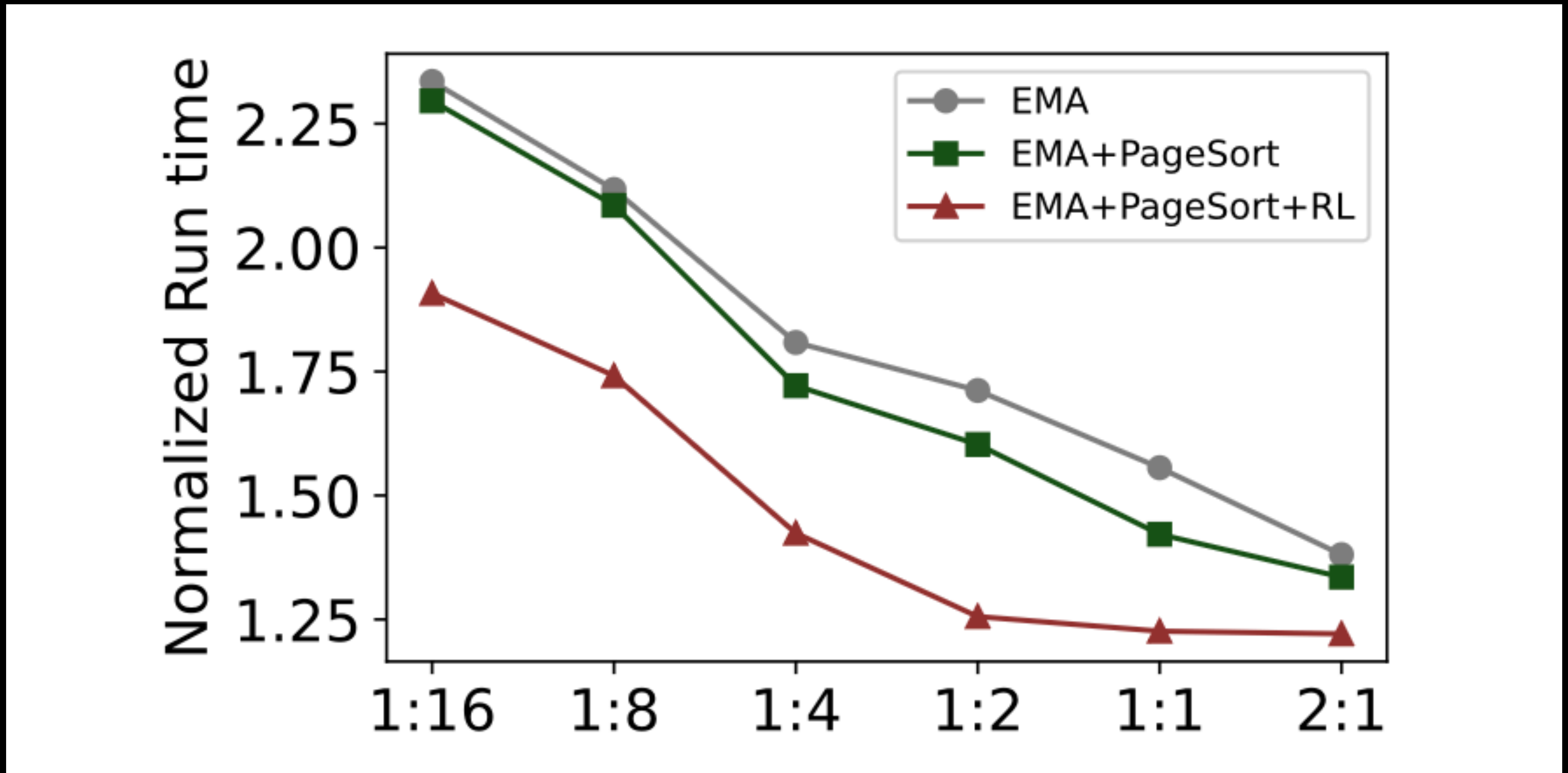
Main Results



Evaluation

Understanding ArtMem Performance

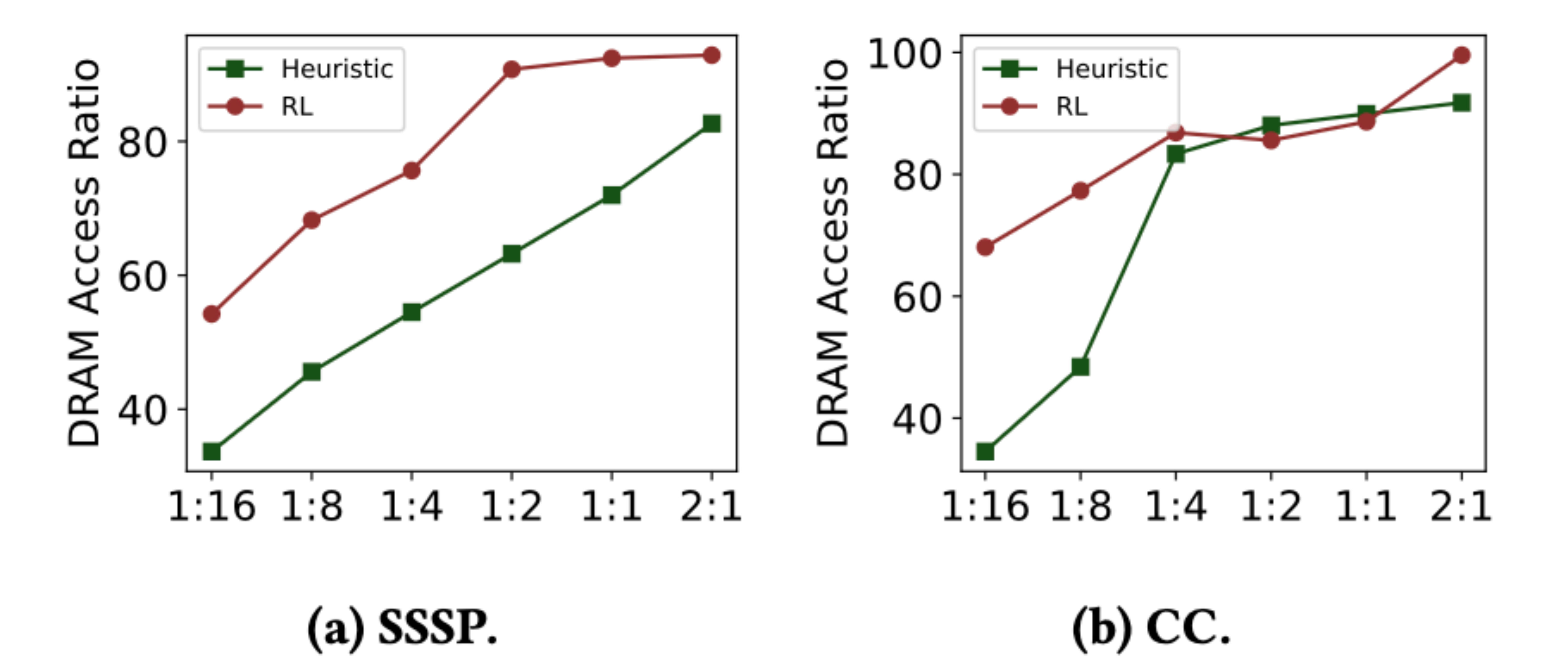
Ablation study



Evaluation

Understanding ArtMem Performance

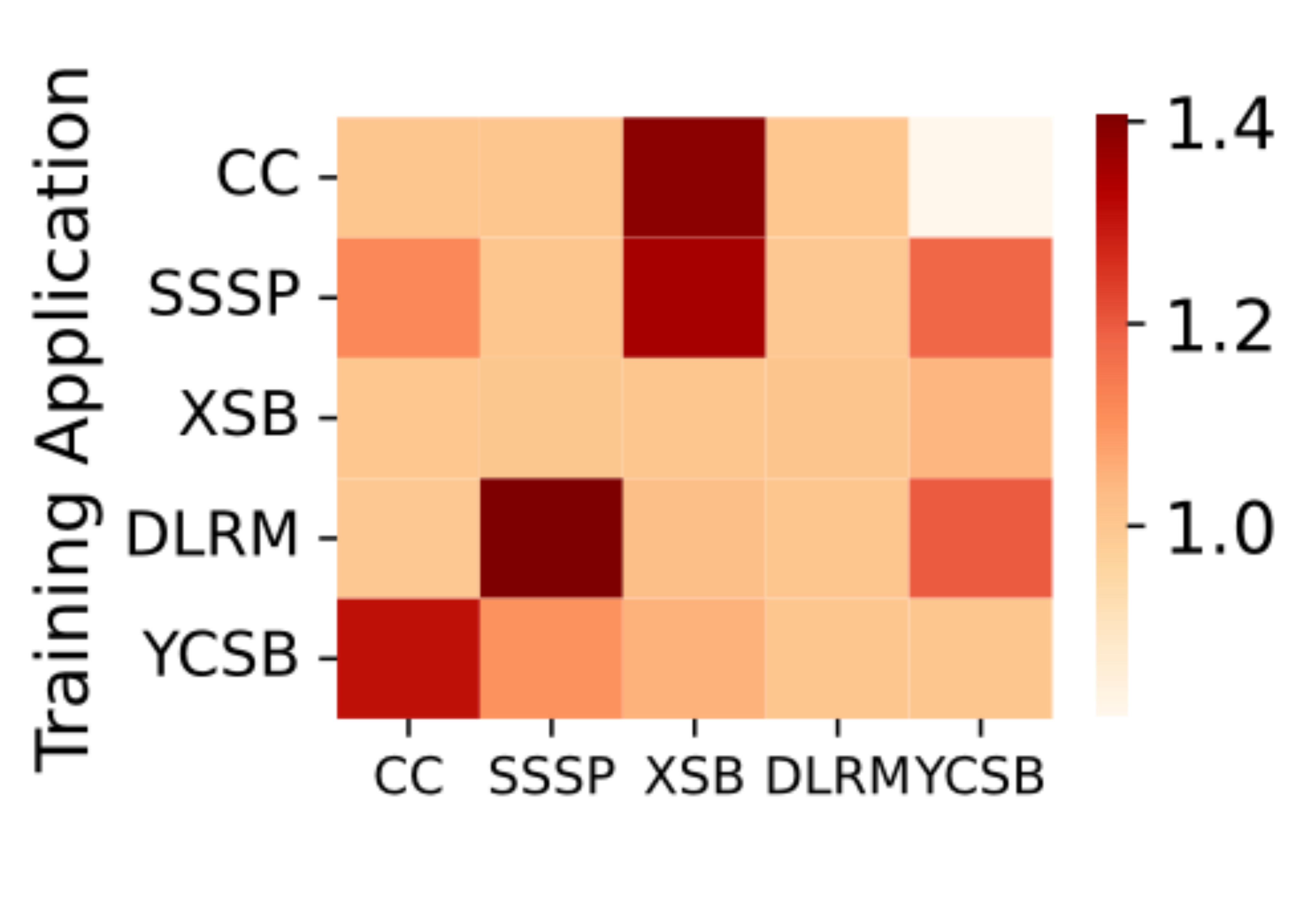
Fast memory access ratio



Evaluation

Understanding ArtMem Performance

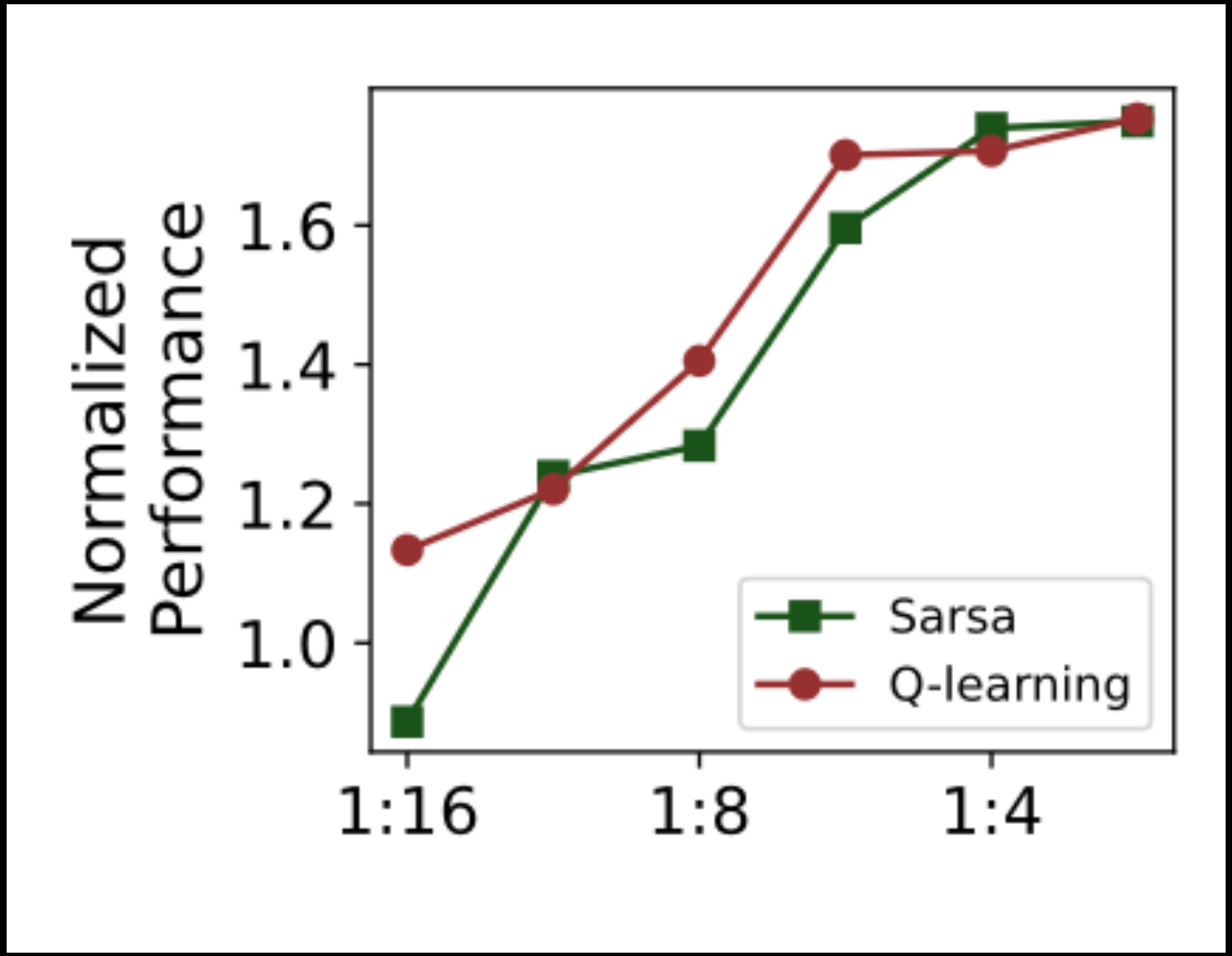
Sensitivity to initial training data



Evaluation

Understanding ArtMem Performance

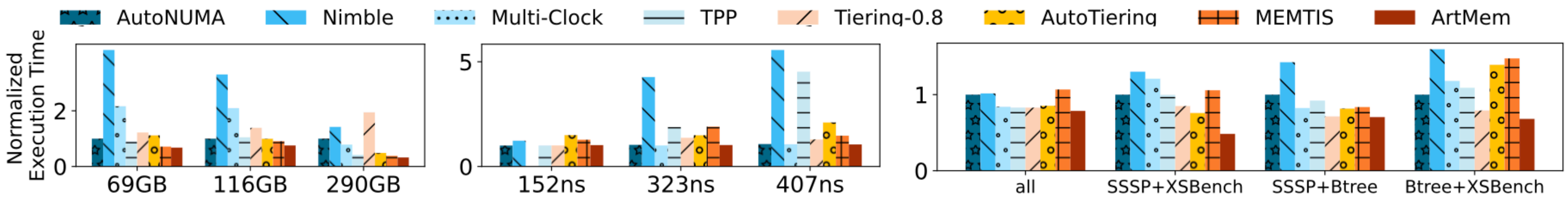
RL algorithm comparison



Evaluation

Understanding ArtMem Performance

Performance comparison



(a) Scalability with memory sizes. (b) Sensitivity to latency. (c) Adaptability for mixed workloads

Conclusion

- Existing tiered memory systems fail to support diverse workloads
- ArtMem combines RL with memory tiering to achieve adaptive memory management supporting a more diverse set of workloads
- It shows significant performance improvement over other systems

After Thoughts

- What I liked:
  - Problem motivation — they had three very clear observations and supported them with appropriate data
- What I didn't like:
  - The design section of the paper — they don't fully explain why this was a challenging system to build
- Open question in my mind: could they have used another ML technique here other than RL?