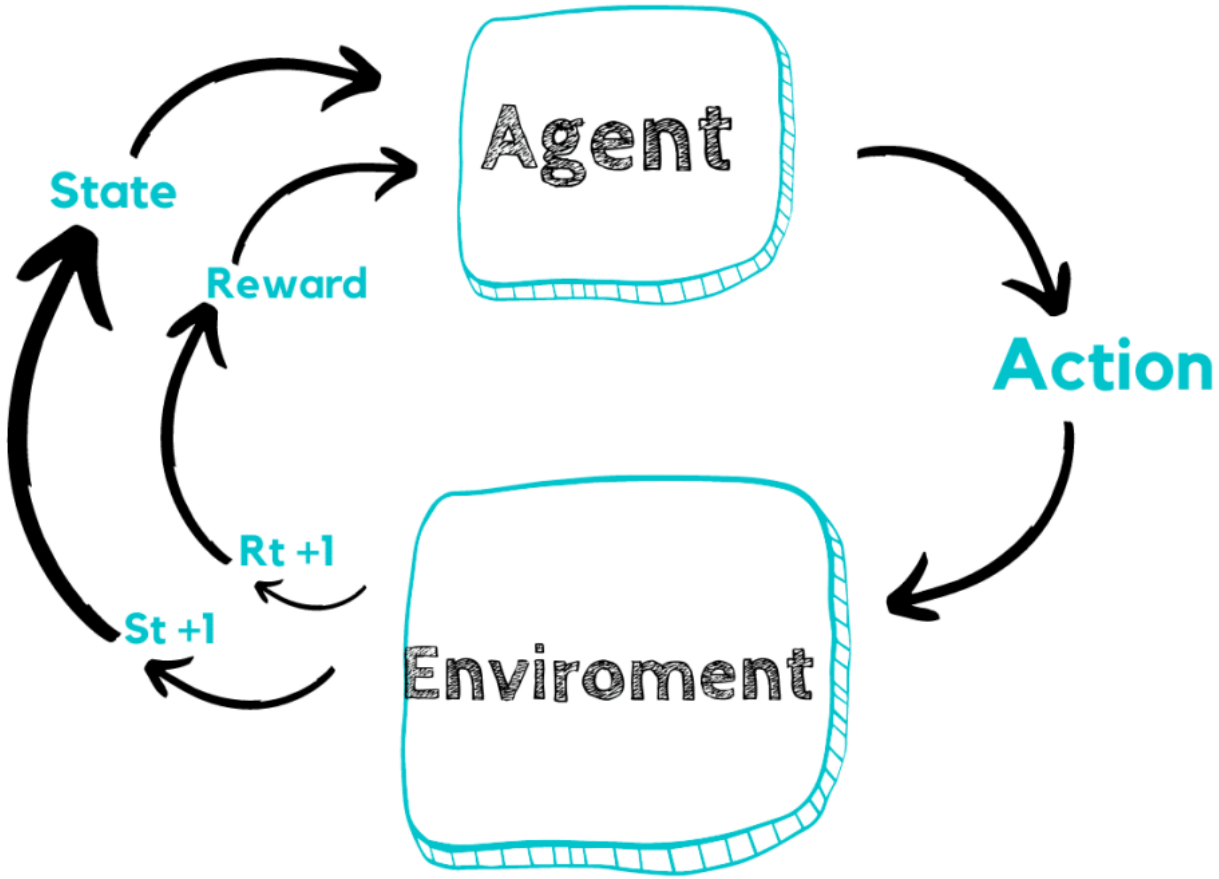
ArtMem: RL for memory tiering

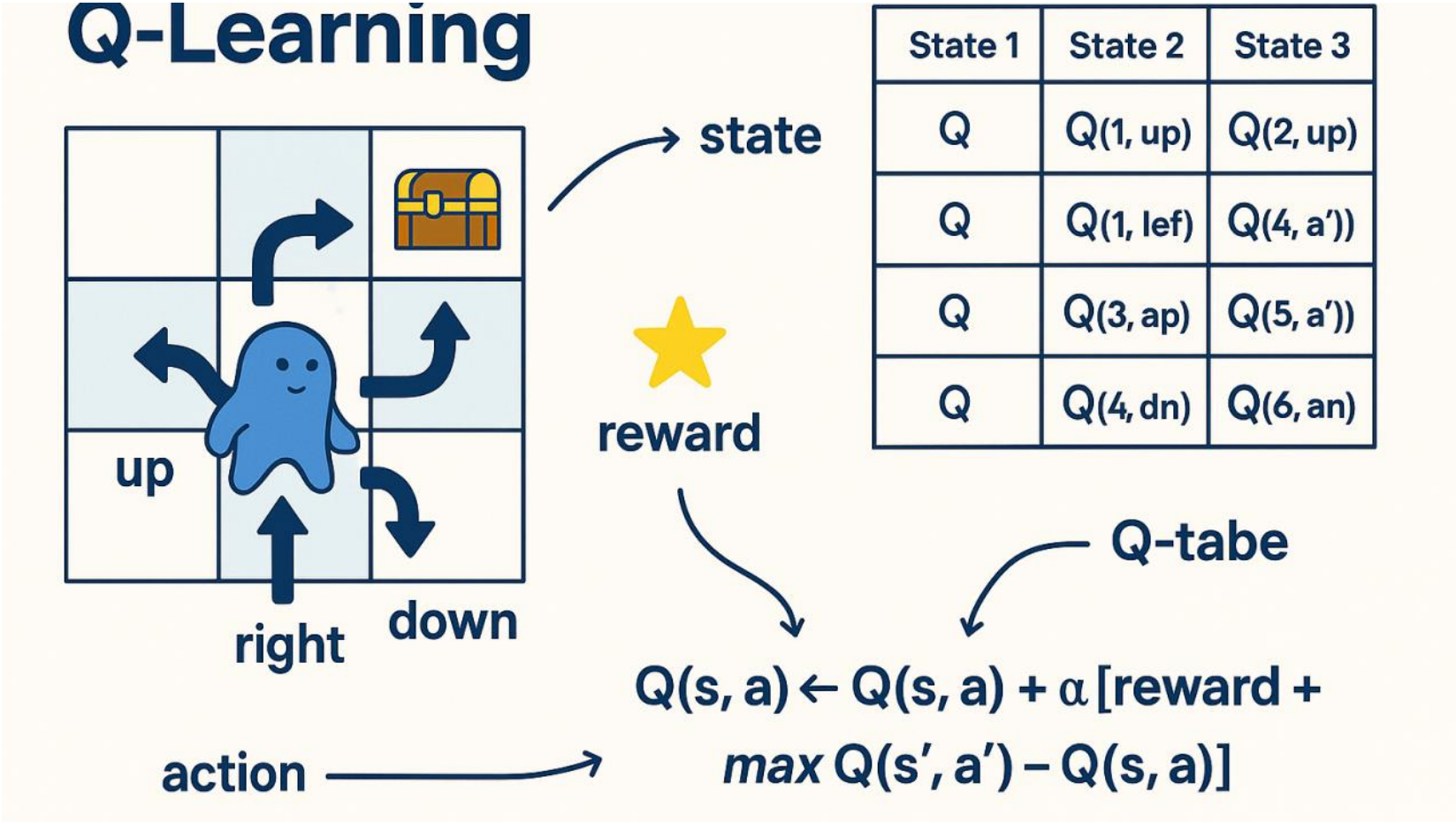
Jan 29

Reinforcement Learning



Q-learning

- Value-based
- Model-free
- Off-policy



Memory tiering

- Demand for memory increasing
- Memory ~35% of total rack cost

- Problem: DRAM scaling has stalled; limited pins on chip

- Solution: use memory alternatives
  - E.g. Non-volatile memory (NVM) or CXL
  - Challenge: New tiers are slower

Tiering systems

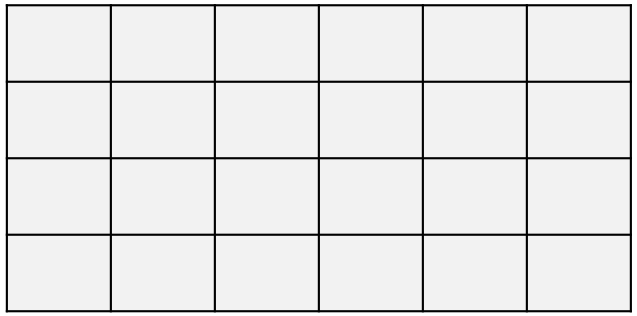
- Goal: Minimize accesses to slow tiers; place "hot" data in fast tier and "cold" data in slow tier

- Which pages to place in fast tier?
  - Access frequency or access recency

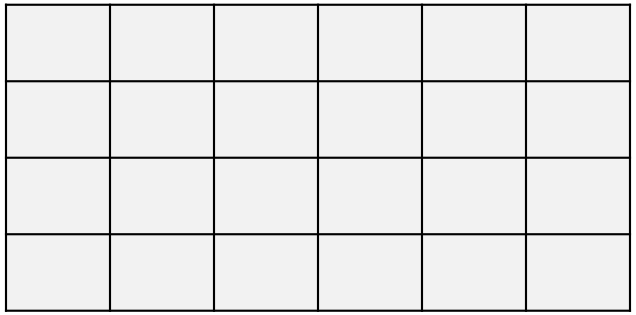
  E.g.: Page accessed 100 times? Then hot!

- How to adapt to changes in hotness?
  - Migrate pages; either periodically or certain events

Memory tiering: clever data placement & migration

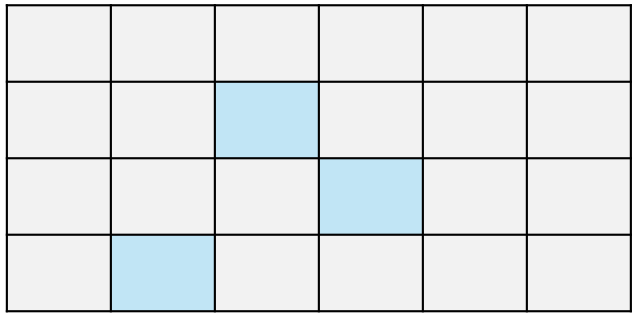


Local DRAM

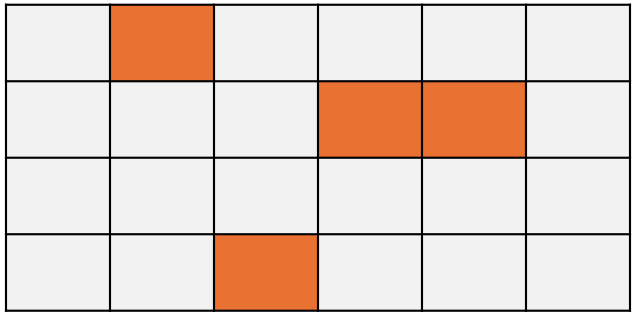


CXL memory

Memory tiering: clever data placement & migration



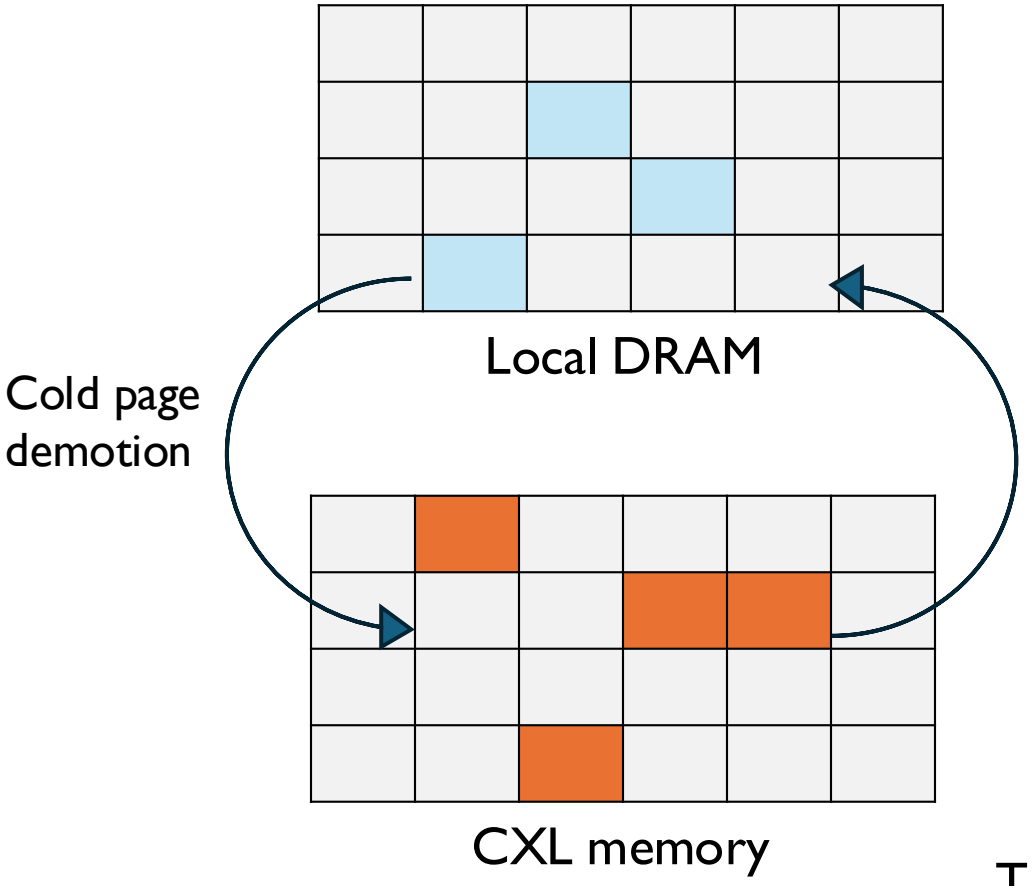
Local DRAM



CXL memory

Step 1: Identify “hot” and “cold” data

Memory tiering: clever data placement & migration



Step 1: Identify "hot" and "cold" data

Step 2: Promote hot data to local DRAM, demote cold data to CXL memory

Different approaches:

1) Hardware based: E.g. Intel Flat MM
2) Software based
   - Application-transparent (e.g. TPP, HMSDK, Hemem, Memtis)
   - Application-aware (e.g. MaPHeA)

This talk!

Challenge

- Uses static values
  - How to configure them?

Parameter	Defaults
Hot threshold	8 accesses
Cooling trigger	18 accesses
Sampling rate	1 in 10K
Migration interval	10ms
Migration rate	2 GB/s
….	

Discussion

1. What are the challenges of RL for systems? When does it work? When might it fail?

2. Another way to build an adaptive tiering system is to use a learned policy directly. The learned policy can predict which pages to migrate and when. Compare this with ArtMem.