A Reinforcement Learning-Based Caching in File System

Peihao Wang, Yuehao Wang, Rui Wang

Introduciton

- General Cache
  - *Data Buffering*
  - *Fast Access*
  - *Replacement*

- Examples
  - *CPU ⇔ Memory*
  - *Virtual Memory*
  - *Translation Lookaside Buffer (TLB)*
  - *File System*
  - *Networking Content*

Introduciton

- Big Idea: Optimal Replacement Policy
  - *Keep the farthest content OUT*
  - *Keep the most recent content IN*
  - *Clairvoyant*
  - *No actual implementation*
- Our Idea: Reinforcement Learning

States and Actions

State Space

- Suppose C slots for caching.
- State transitions happen on cache misses.
- Denote state as $\boldsymbol{s} = \{\mathcal{F}_0, \mathcal{F}_1, \dots, \mathcal{F}_c\}$
  - Features descriptors to represent large state space.
  - $\mathcal{F}_0$ is the feature vector of currently requested resource.
  - $\mathcal{F}_1, \dots, \mathcal{F}_c$ is the feature vector of each cache slot.
  - In our design, $\mathcal{F}_i = \{f_i^s, f_i^m, f_i^l\}$, where f_i^s, f_i^m, f_i^l represent the recent access times within short, middle, long terms, respectively.

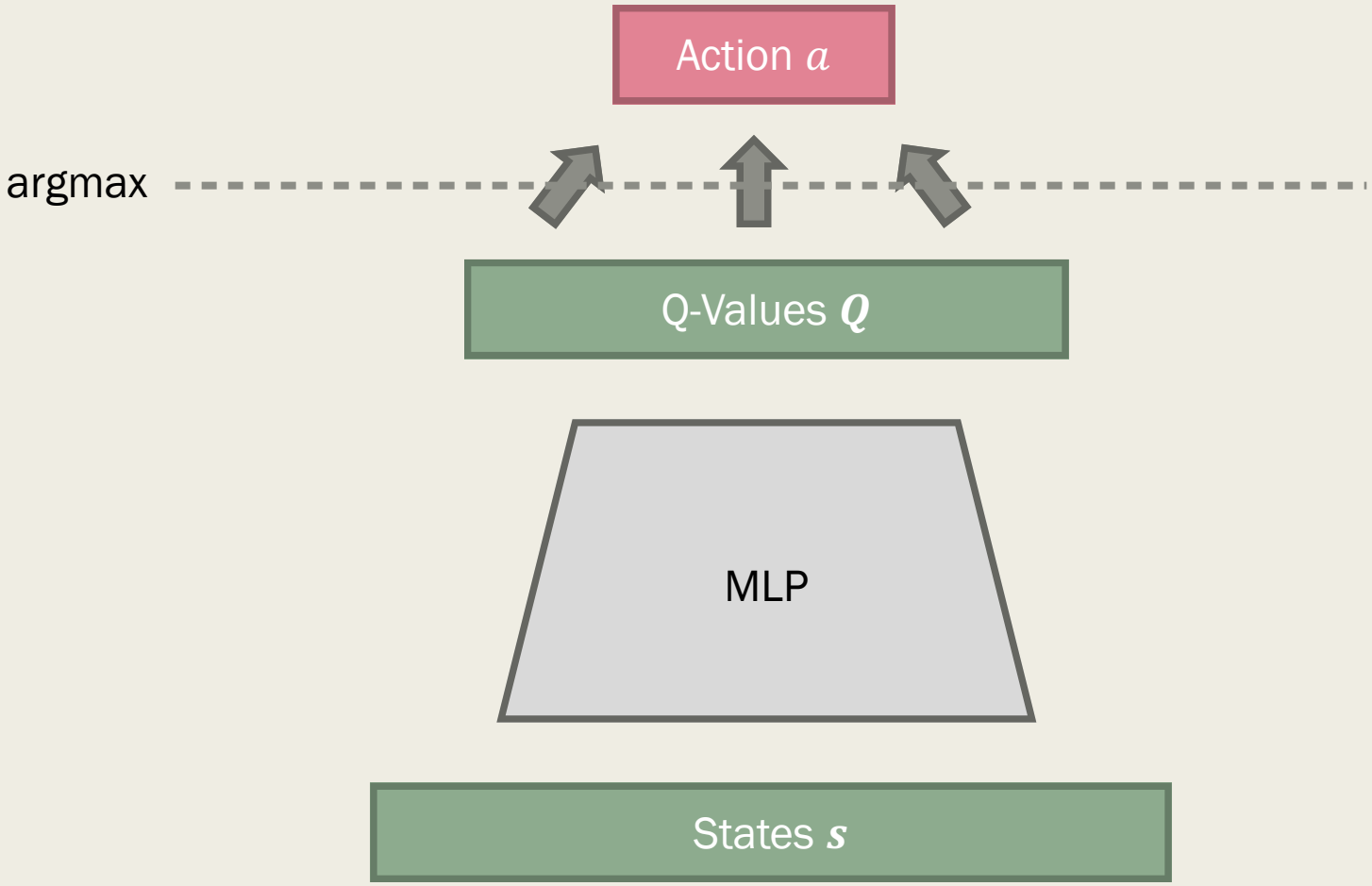
Action Space

- Action is taken when caches conflict.
- Denote action as $a \in \{1, \dots, C\}$.
  - a indicates which cache slot should be replaced.

Rewards

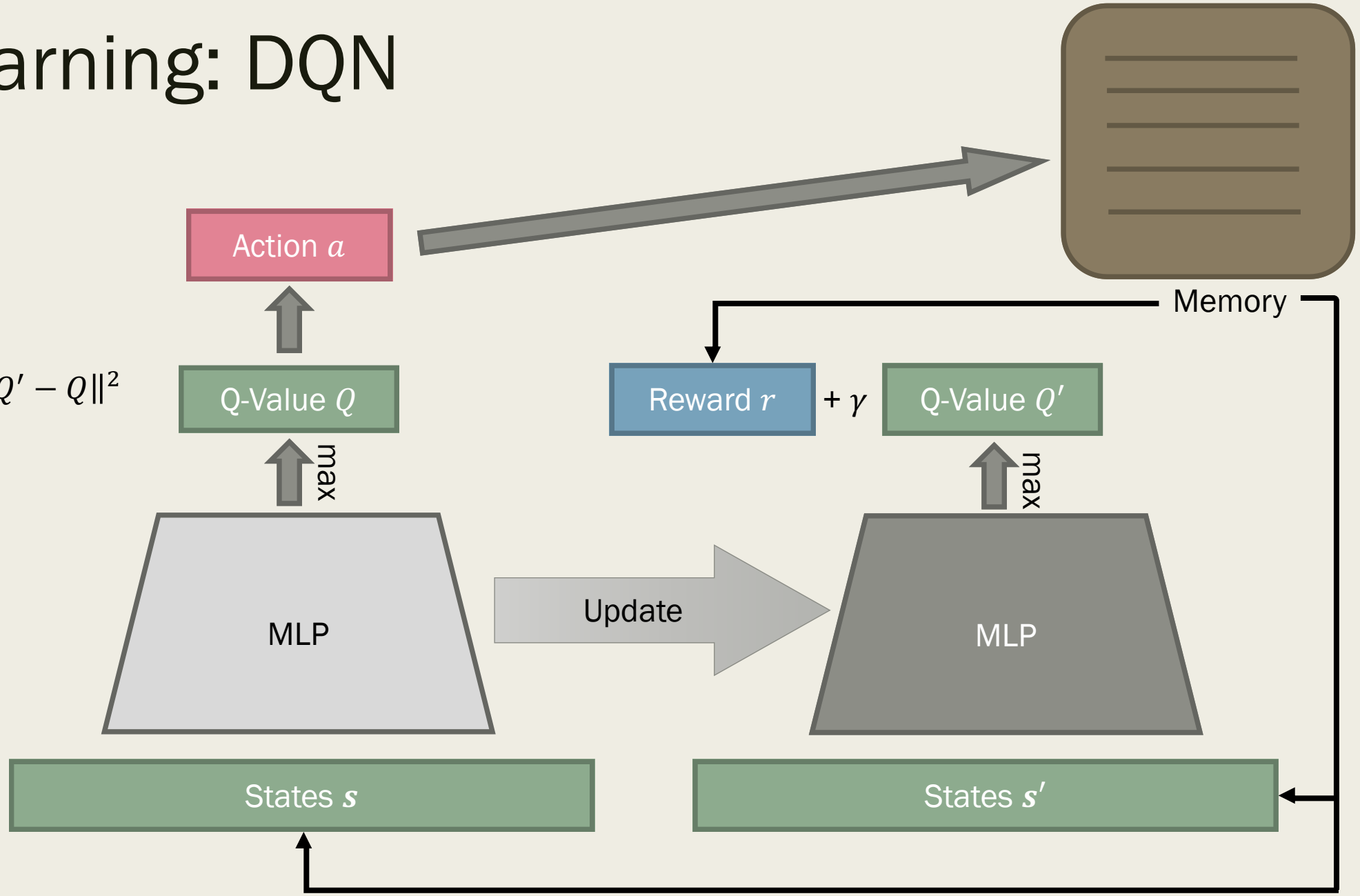
- Rewards evaluate the replacement choice.
- Denote rewards of a transition as $R(s, a, s') = H(s, s') + \lambda \cdot P(s, a, s')$.
  - $H(s, s') = \sum_i (n'_i - n_i)$ evaluates the total number of hits between two following cache misses.
    - n_i represent the number of hits at slot i for state s.
    - n'_i represent the number of hits at slot i for state s'.
  - $P(s, a, s')$ is the penalty of action a, resulting in the next cache miss.
    - i.e. $P(s, a, s') \neq 0$ if replacement at s by a causes the cache miss at s'.
    - It should decrease as cache hits $H(s, s')$ accumulate.
    - e.g. $P(s, a, s') = \mu + \frac{\psi}{H(s,s')}$ if a causes the next cache miss, where $\psi < 0$.

Q-Learning: DQN



Q-Learning: DQN

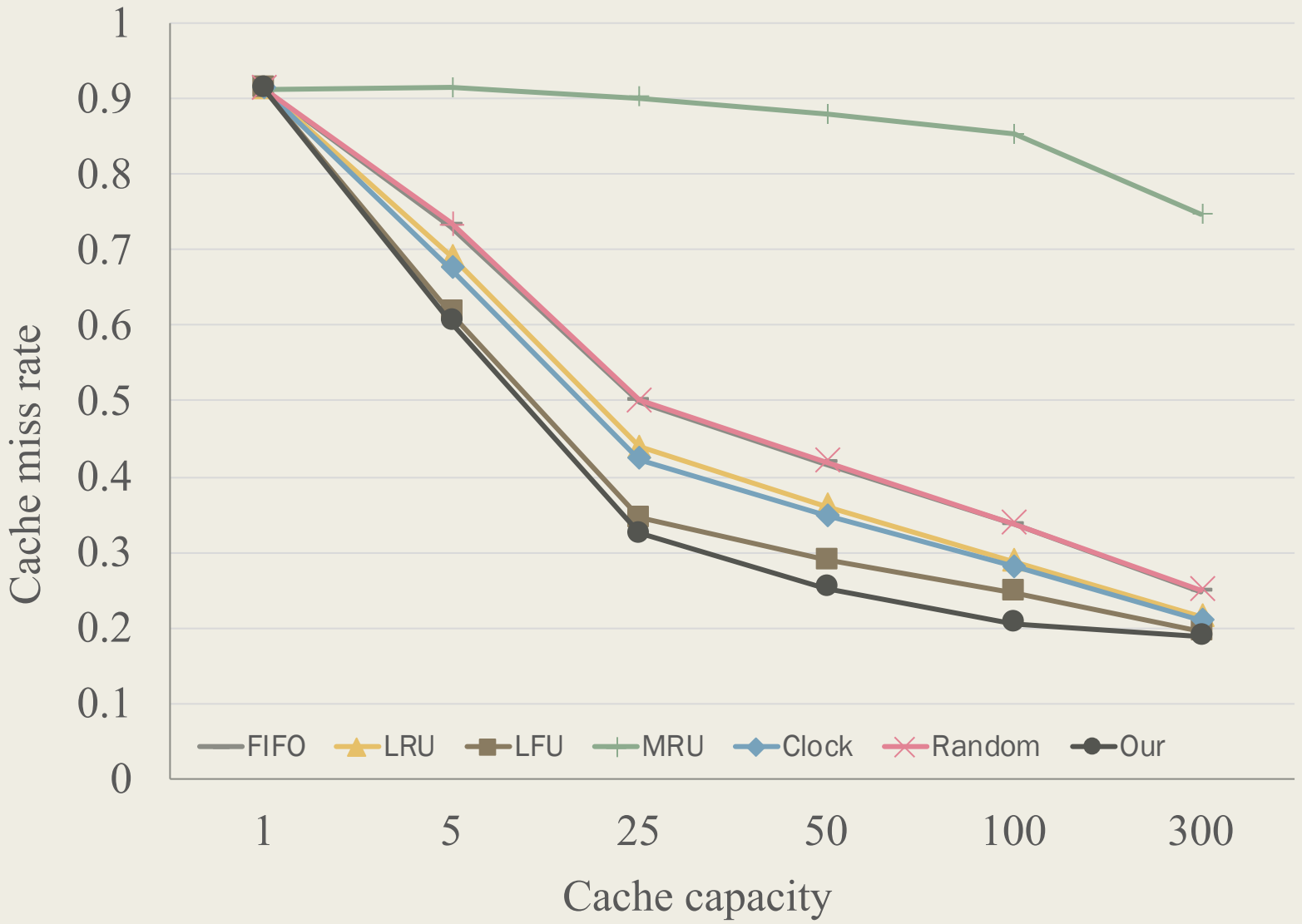
$$\mathcal{L} = \|r + \gamma Q' - Q\|^2$$



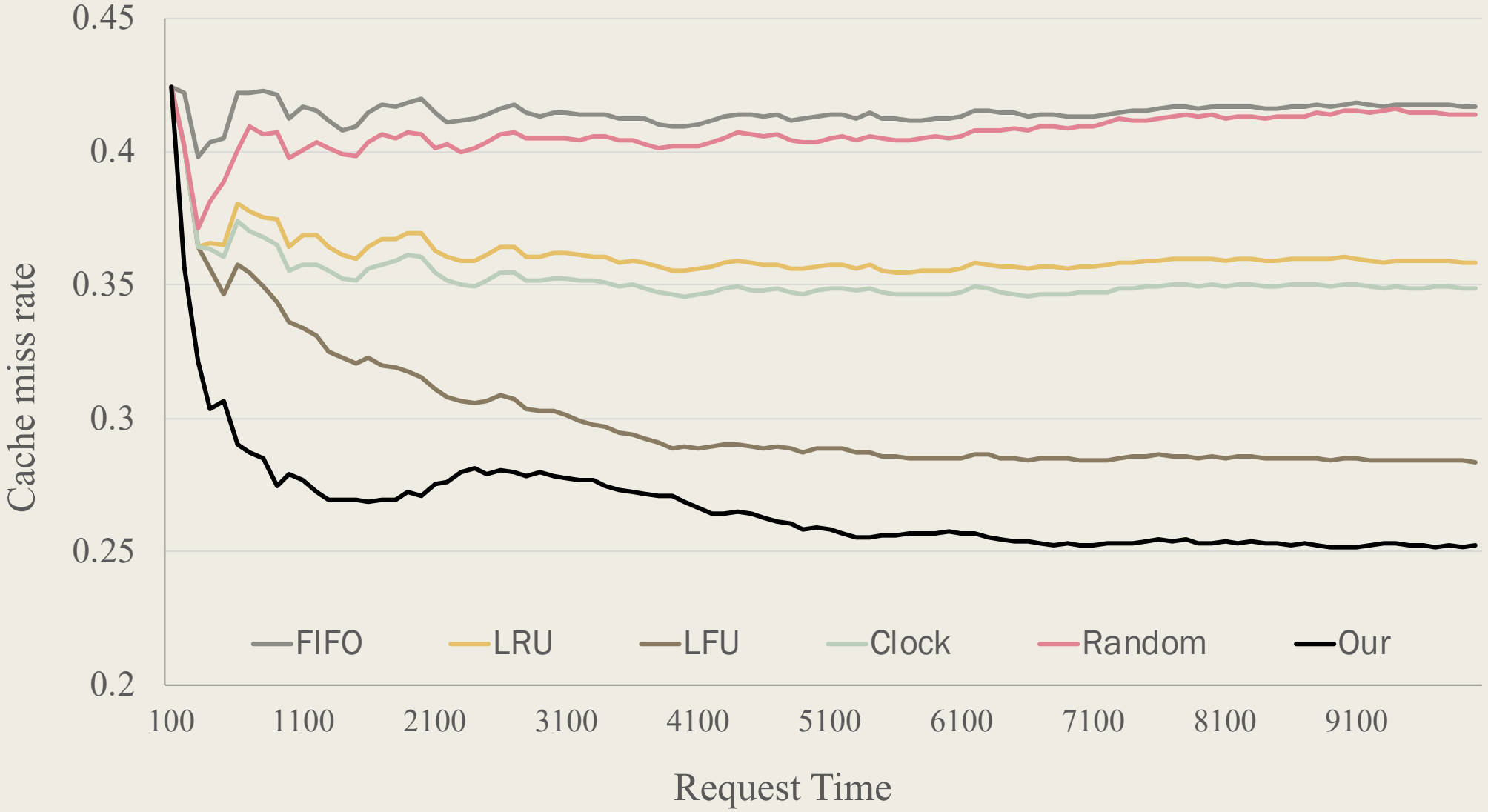
Experiment Ⅰ

- *Zipf* distribution:
  - *Zipf* is a general **content popularity distribution.**
  - In our simulation, our distribution parameter is 1.3.
- Compared with FIFO, Random, LRU, LFU, Clock, MRU.

Experiment Ⅰ: Miss rate vs. Capacity



Experiment Ⅰ: Miss rate vs. Time c = 50



Experiment Ⅱ

- 5 typical testcases from Pintos:
  - The 5 typical testcases have enough cache accesses. => Produce enough cache misses.
  - Compared with FIFO, LRU, LFU, MRU, Clock, Random
  - Cache size = 50

Experiment Ⅱ: Comparison in Pintos

	FIFO	LRU	LFU	MRU	Clock	Random	Our
dir-open	20.38%	19.91%	19.56%	42.83%	19.91%	19.41%	18.03%
dir-vine	1.49%	1.44%	57.67%	62.16%	1.44%	1.55%	1.80%
grow-create	21.46%	20.96%	20.59%	42.29%	20.96%	20.88%	19.17%
grow-file-size	17.80%	17.40%	17.09%	39.97%	17.40%	16.66%	16.12%
grow-seq-sm	20.45%	19.99%	19.65%	43.14%	19.99%	19.61%	18.25%
syn-rw-persistence	15.74%	14.61%	14.13%	68.08%	14.71%	15.72%	14.94%

Experiment Ⅲ

- Extreme case: sequential flooding
- A common issue for LRU
- Cache capacity = 5
- Sequence: 1 2 3 4 5 6 repeatedly

	FIFO	LRU	LFU	MRU	Clock	Random	Our + 1 ep	Our + 40 eps
Miss Rate (%)	100%	100%	100%	20%	100%	33.41%	31.32%	**27.22%**

P.S. ep(s) = episode(s)

Experiment IV

- In Table 1, combine different features and test their miss rates.
- In Table 2, use different rewards and test their miss rates.
- Cache capacity = 50

Table 1: Features

Freq Only	0.2522
Freq + VT	0.246
Freq + VT + IT	0.2639
Freq + VT+ IT + Cache state	0.3414

Table 2: Rewards

# hit per epoch	0.2676
# hit per epoch + penaly	0.2522
short+long term hit rate	0.2754

Freq=frequency, VT = visit time, IT = swap-in time

References

- *Giuseppe Vietri, Liana V. Rodriguez, Wendy A. Martinez, Steven Lyons, Jason Liu, Raju Rangaswami, Ming Zhao, and Giri Narasimhan. 2018.* ***Driving cache replacement with ML-based LeCaR****. In Proceedings of the 10th USENIX Conference on Hot Topics in Storage and File Systems (HotStorage'18). USENIX Association, Berkeley, CA, USA, 3-3.*
- *C. Zhong, M. C. Gursoy and S. Velipasalar, "****A deep reinforcement learning-based framework for content caching****", 2018 52nd Annual Conference on Information Sciences and Systems (CISS), Princeton, NJ, 2018, pp. 1-6.*
- *Dulac-Arnold, Gabriel, et al. "****Deep reinforcement learning in large discrete action spaces****." arXiv preprint arXiv:1512.07679 (2015).*
- *Susanne Albers, Sanjeev Arora, and Sanjeev Khanna. 1999.* ***Page replacement for general caching problems****. In Proceedings of the tenth annual ACM-SIAM symposium on Discrete algorithms (SODA '99). Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 31-40.*
- *Mnih, Volodymyr, et al. "****Playing atari with deep reinforcement learning.****" arXiv preprint arXiv:1312.5602 (2013).*
- *Suksomboon, Kalika, et al. "****PopCache: Cache more or less based on content popularity for information-centric networking.****" 38th Annual IEEE Conference on Local Computer Networks. IEEE, 2013.*
- *Wang, Haonan, et al. "****Learning Caching Policies with Subsampling.****" NeurIPS Machine Learning for Systems Workshop, 2019.*
- *Reinforcement-learning-with-tensorflow https://github.com/MorvanZhou/Reinforcement-learning-with-tensorflow*

Thanks for Listening