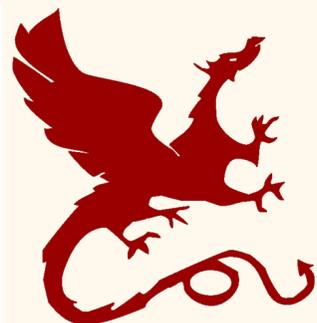


FIFO can be better than LRU: the power of LAZY PROMOTION and QUICK DEMOTION



Juncheng Yang, Ziyue Qiu, Yazhuo Zhang, Yao Yue, K. V. Rashmi

Carnegie
Mellon
University
School of
Computer
Science



EMORY
UNIVERSITY

Pelikan Foundation

Carnegie Mellon
Parallel Data Laboratory

Software cache and eviction

- Ubiquitous deployments of software caches
 - page cache, block cache, database cache
 - key-value cache, object cache...
- Cache metrics:
 - efficiency / effectiveness: miss ratio
 - throughput and scalability: requests /sec
- A core component of cache design: **eviction**



Cachelib



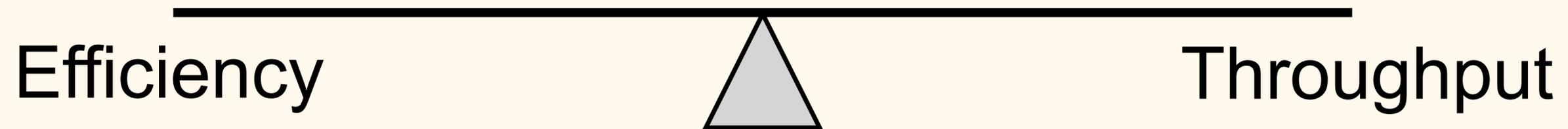
Pelikan



A long history of research centered around LRU

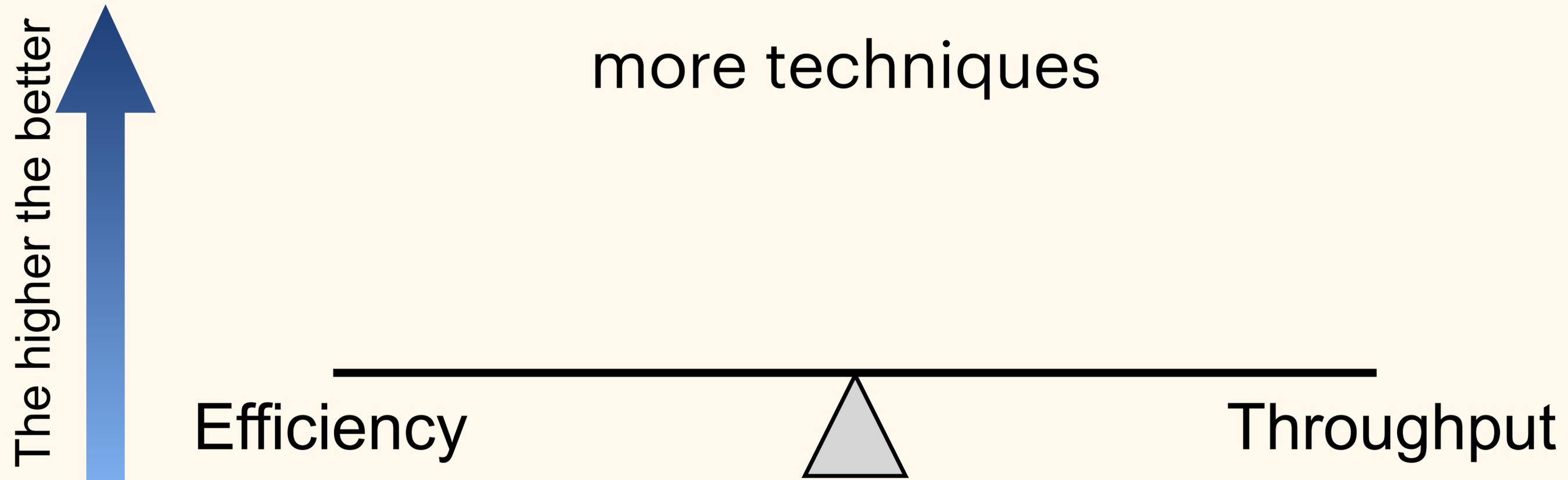
Least-recently-used (LRU)

eager promotion



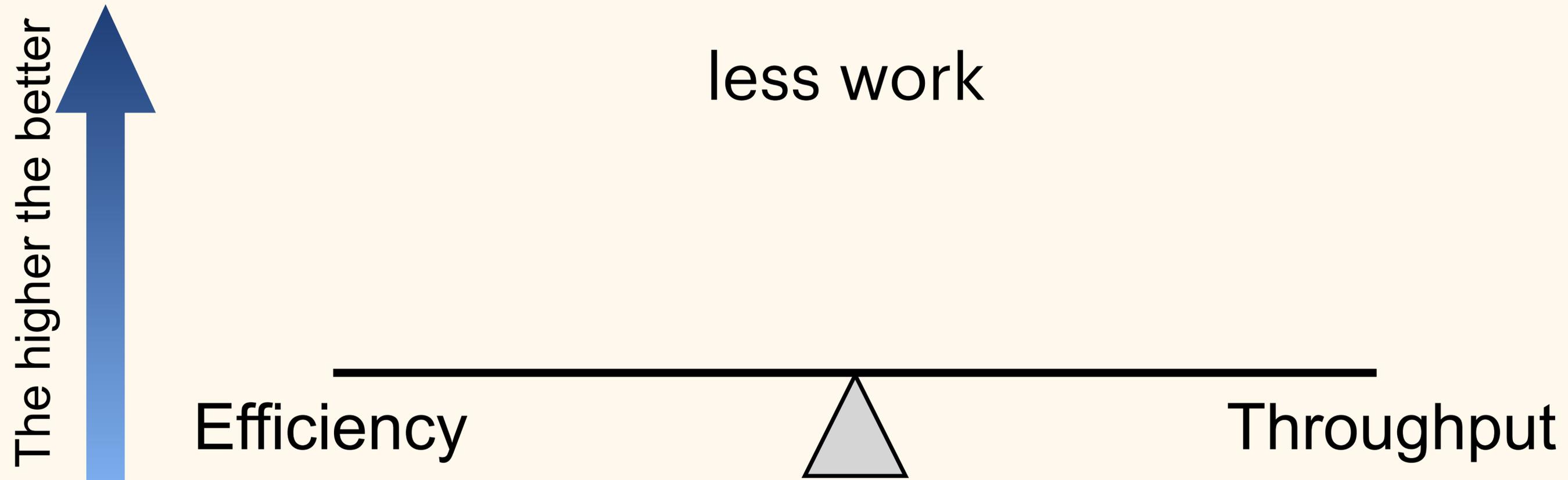
A long history of research centered around LRU

ARC, LIRS, SLRU, MQ, CACHEUS...



A long history of research centered around LRU

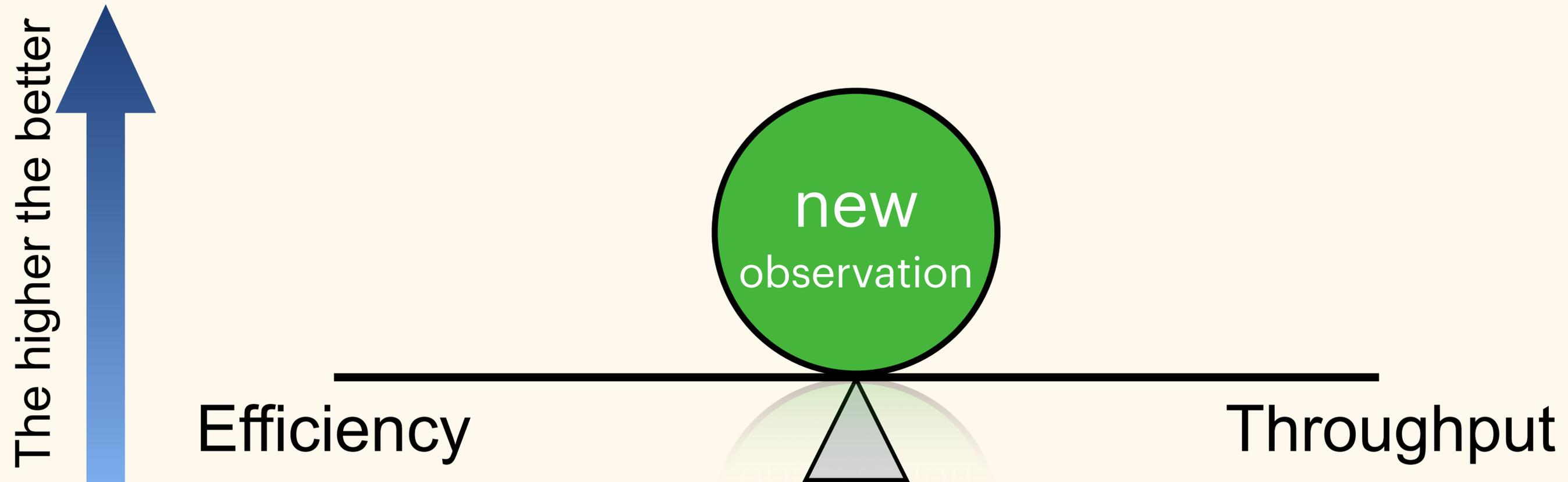
FIFO-Reinsertion, CLOCK variants



Can we have the best of both worlds?

System design is often to make right trade-offs

FIFO can be better than LRU



LRU is ~~bad~~,
let's start with FIFO

Why FIFO?

- Many benefits
 - fewer metadata
 - less computation
 - more scalable
 - flash friendly

The drawback:
FIFO cannot keep **popular**
objects in the cache

What does FIFO need?

- Retain popular objects in the cache

LAZY PROMOTION

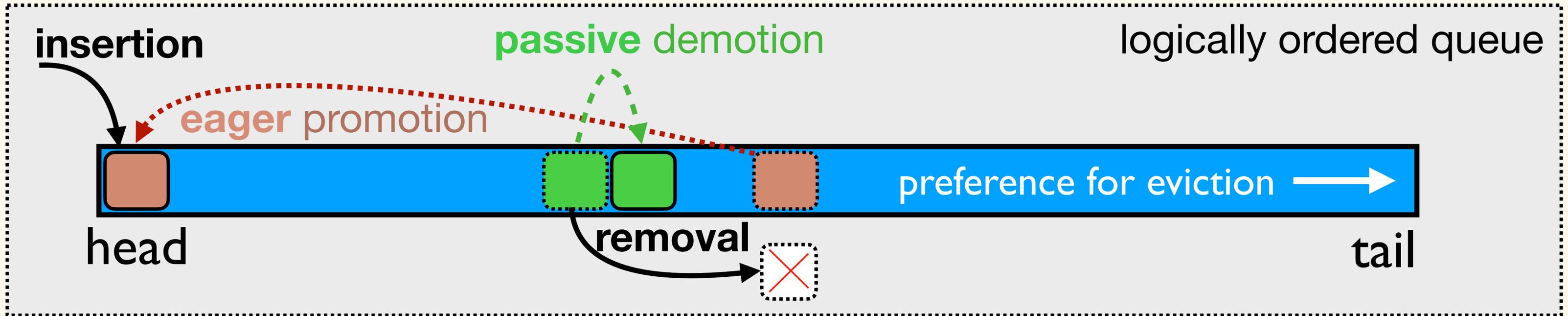


- Evict unpopular objects faster

QUICK DEMOTION

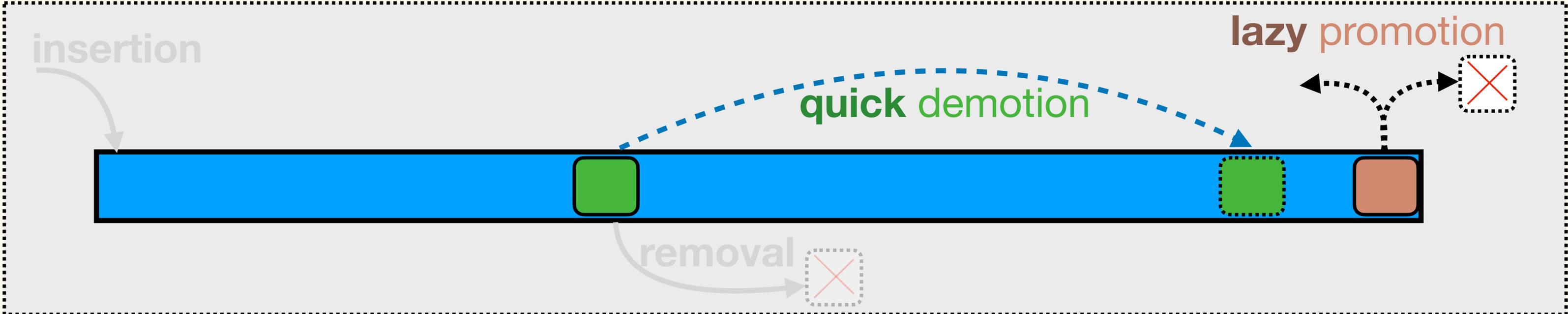
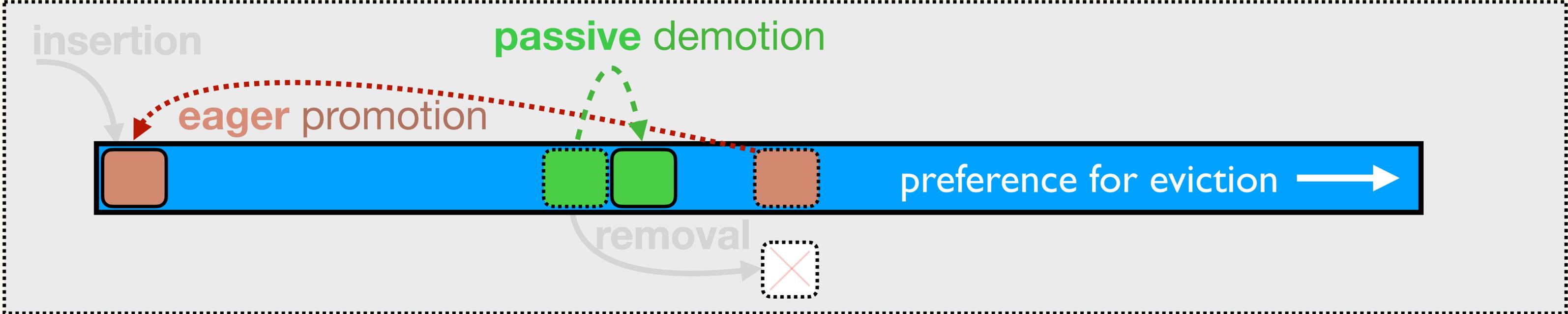


An abstraction of cache



Existing cache eviction algorithms = cache **promotion** algorithms!

An abstraction of cache





LAZY PROMOTION



Promotion only at eviction

LAZY PROMOTION: promotion only at eviction

Retain popular objects with minimal efforts

- Example: FIFO-Reinsertion/CLOCK
 - reinserts an object back during eviction if it has been requested
- Higher throughput (than LRU)
 - fewer operations
 - more scalable
- Higher efficiency (than LRU)
 - more information at eviction time

LAZY PROMOTION: promotion only at eviction

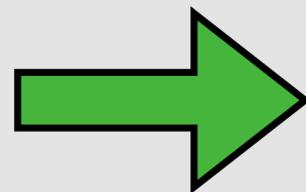
Retain popular objects with minimal efforts

Common wisdom

CLOCK has

~~higher~~
miss ratios

than LRU



We find

lower

Dataset:

- 10 datasets, 5307 traces from 2007-2020
- block, key-value, object
- 814 billion requests , 55.2 billion objects

LAZY PROMOTION: promotion only at eviction

Retain popular objects with minimal efforts

Common wisdom

CLOCK has

~~higher~~ miss ratios

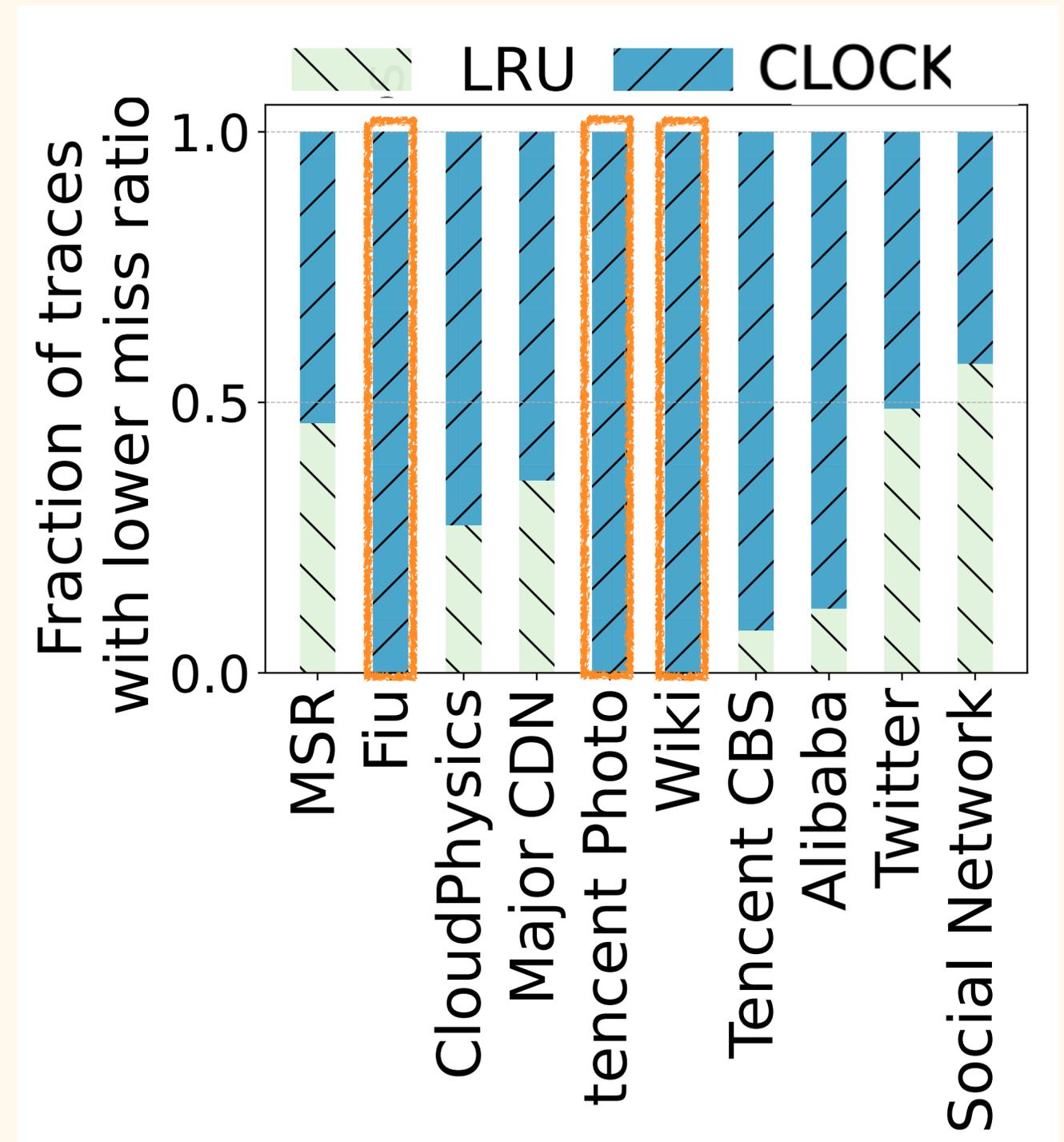
than LRU

→

We find

lower

CLOCK (FIFO + LP) is also simpler, faster, and more scalable





QUICK DEMOTION



Quickly remove most new objects

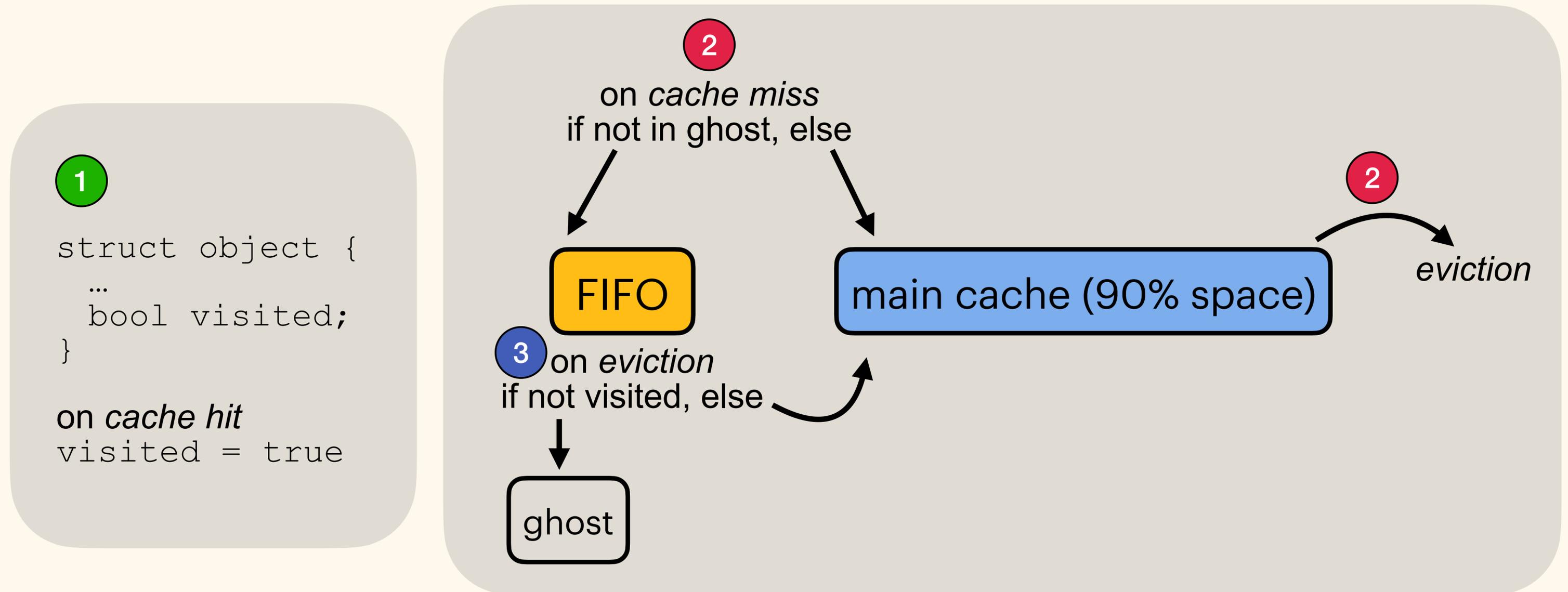
QUICK DEMOTION: quickly remove most new objects

Evict unpopular / short-lived objects faster

- Removing less valuable objects faster is not new
 - Remove scan pages/data
- Why use quick demotion
 - Zipf workloads: unpopular objects are the majority of objects
 - Belady evicts unpopular objects faster

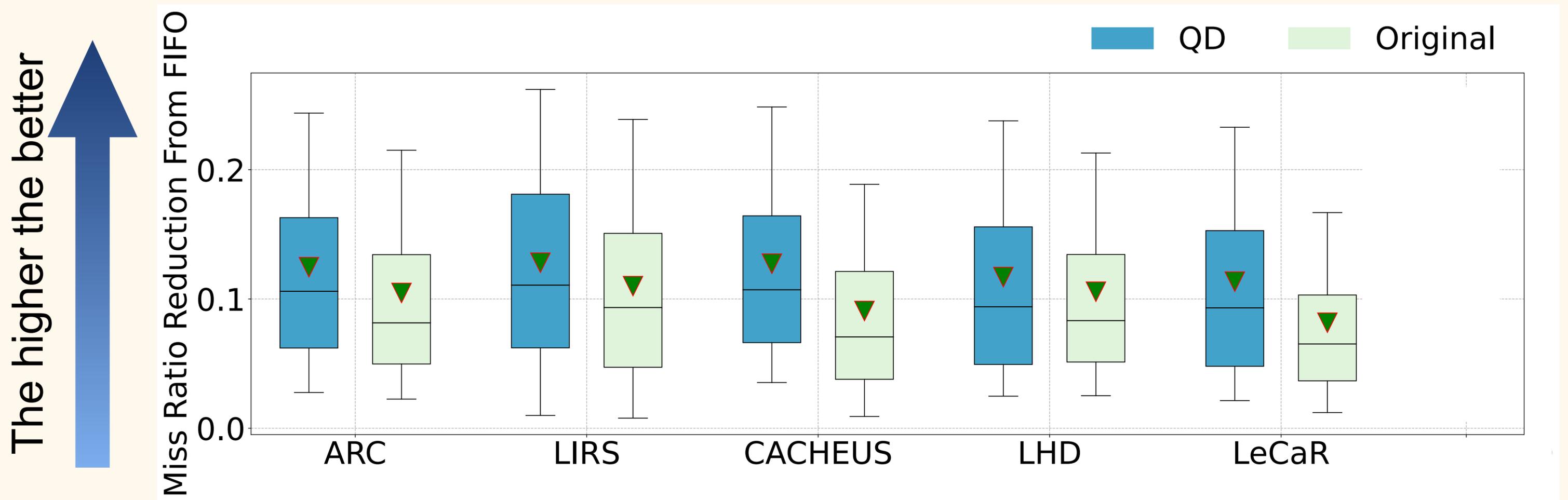
QUICK DEMOTION

A simple design to illustrate the power



QUICK DEMOTION (QD): probationary FIFO quickly removes many new objects

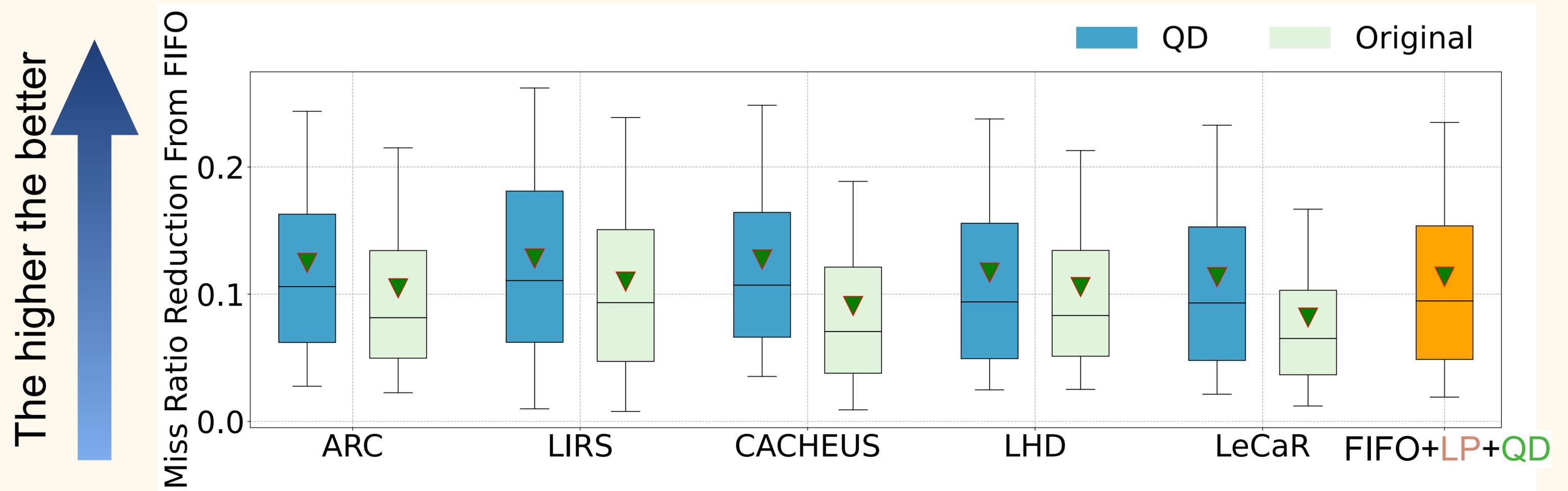
QD improves ALL state-of-the-art algorithms



ARC → QD-ARC: up to 59.8% miss ratio reduction

Across all traces, algorithms and sizes, mean miss ratio reduction: 2.7%

FIFO + LP + QD is better than state-of-the-art



FIFO + LP + QD:

- more **efficient** than state-of-the-art
- **simple, fast, scalable**

More lazy promotion and quick demotion techniques

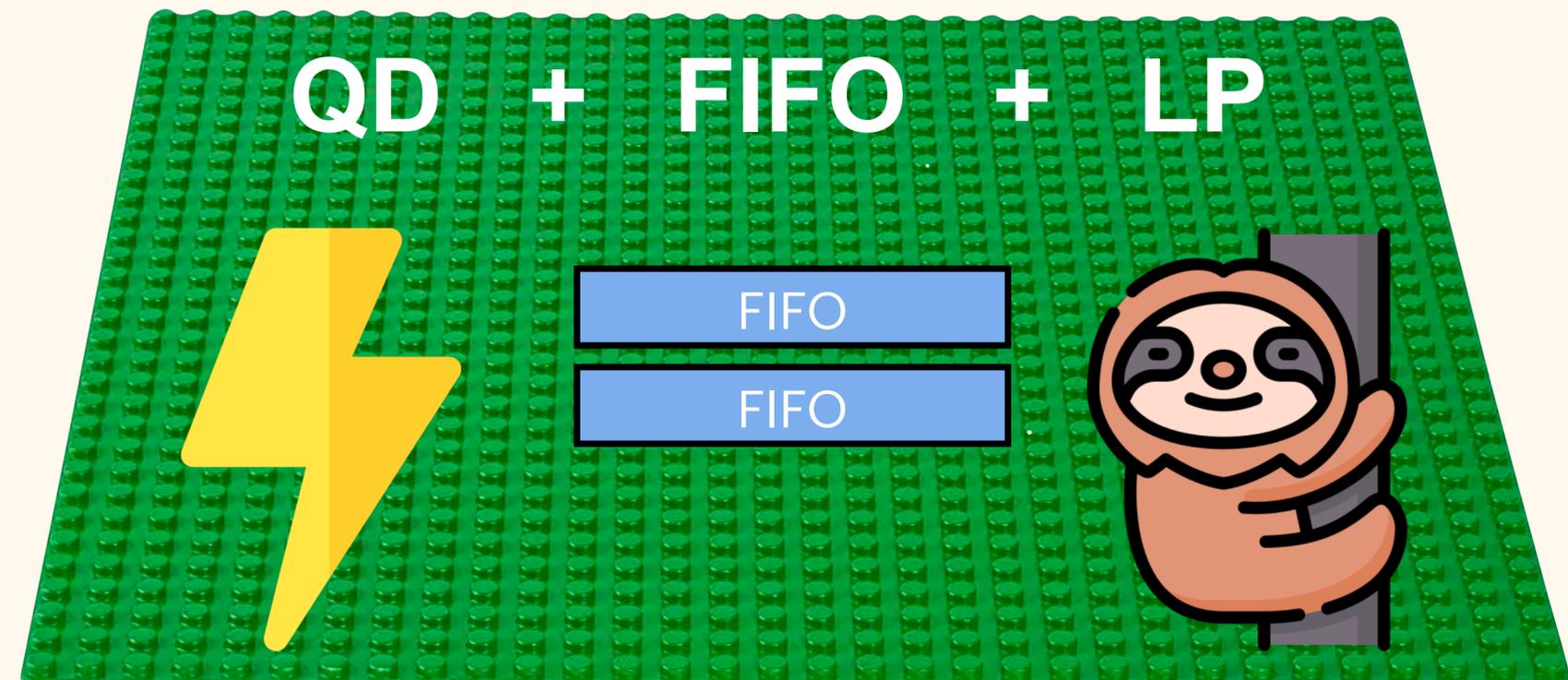
- **LAZY PROMOTION**

- reinsertion
- merge (Segcache)
- periodic (FrozenHot)
- batched (CliqueMap)
- probabilistic

- **QUICK DEMOTION**

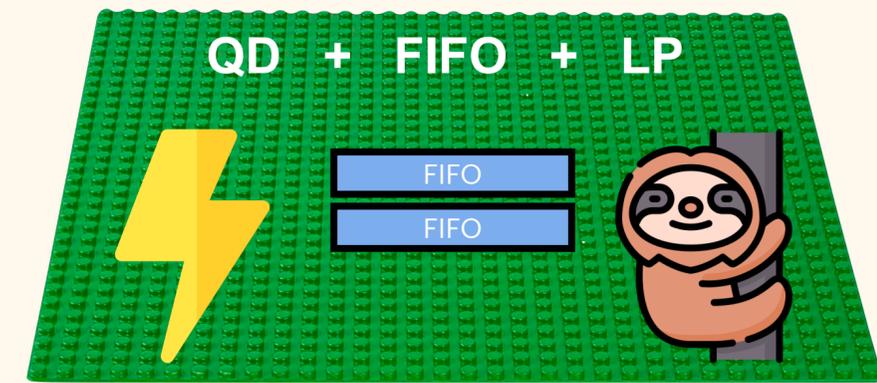
- small FIFO
- new metric

Design cache eviction algorithms like building LEGOs



Takeaways

FIFO is better than you would have expected



- **FIFO + LAZY PROMOTION** can be more efficient than LRU
- **QUICK DEMOTION** enables state-of-the-art efficiency
- Design new eviction algorithms using

FIFO + LAZY PROMOTION + QUICK DEMOTION

simple, fast, scalable, yet efficient

<https://github.com/TheSys-lab/HotOS23-QD-LP>

Acknowledgement

- Open-sourced traces
- Cloudlab, PDL

<https://jasonry.me>
juncheny@cs.cmu.edu
[@1a1a11a](#)

<https://jasonry.me/slides/hotos23-qdlp.pdf>

