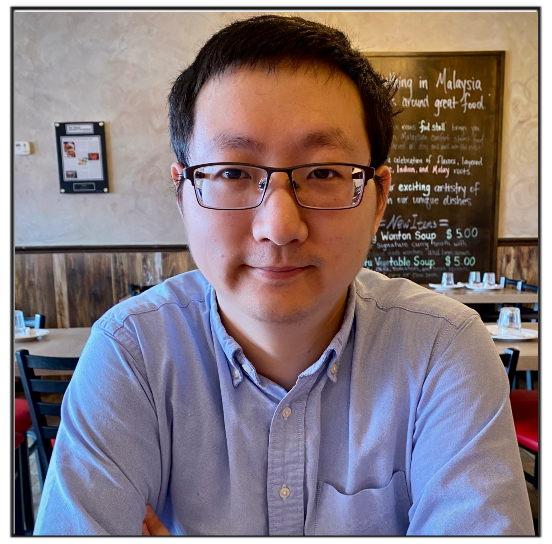
SIEVE is Simpler than LRU:

An Efficient Turn-Key Eviction Algorithm for Web Caches

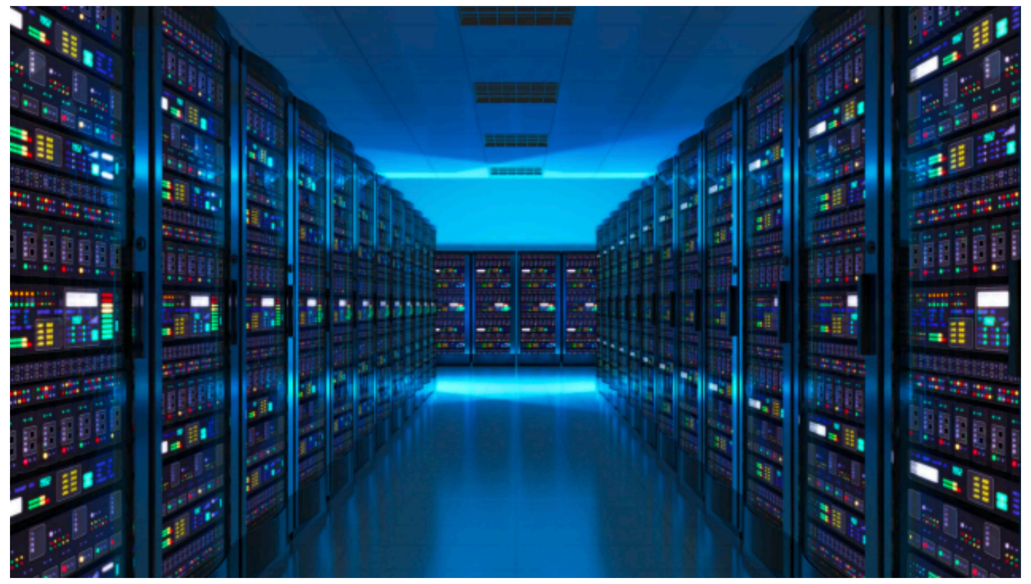




Yazhuo Zhang, Juncheng Yang, Yao Yue, Ymir Vigfusson, K.V. Rashmi

Emory University, Carnegie Mellon University, Pelikan Foundation

Caching System is Important

Page Cache



MEMCACHED

redis

Cachelib

Pelican

traffic server™

Web Caches

Limited Space!

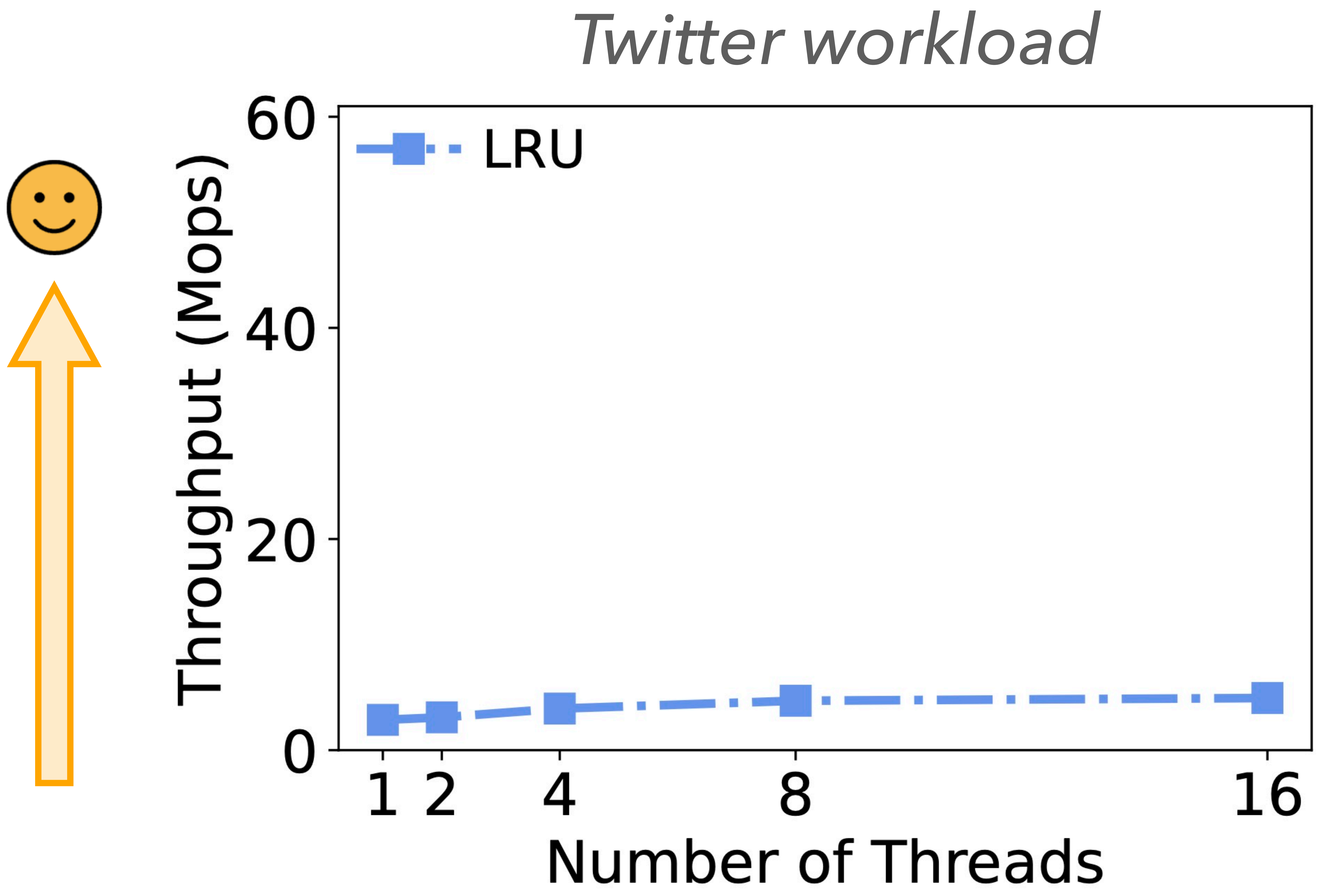
Core: Eviction Algorithm

Cache Metrics

Efficiency — *Cache Miss Ratio*

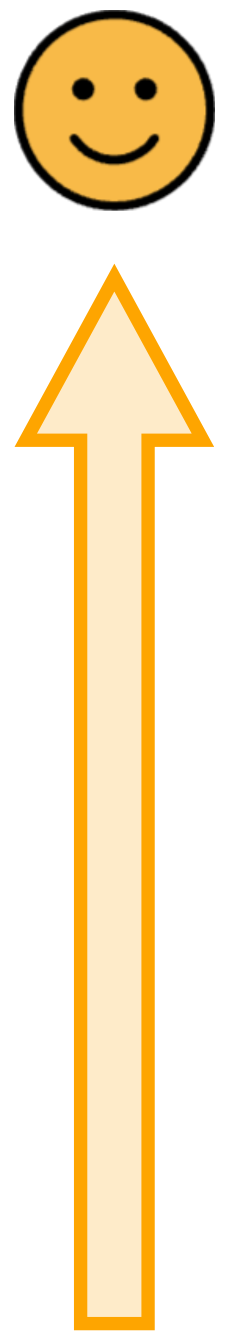
Scalability — *Reqs/Second*

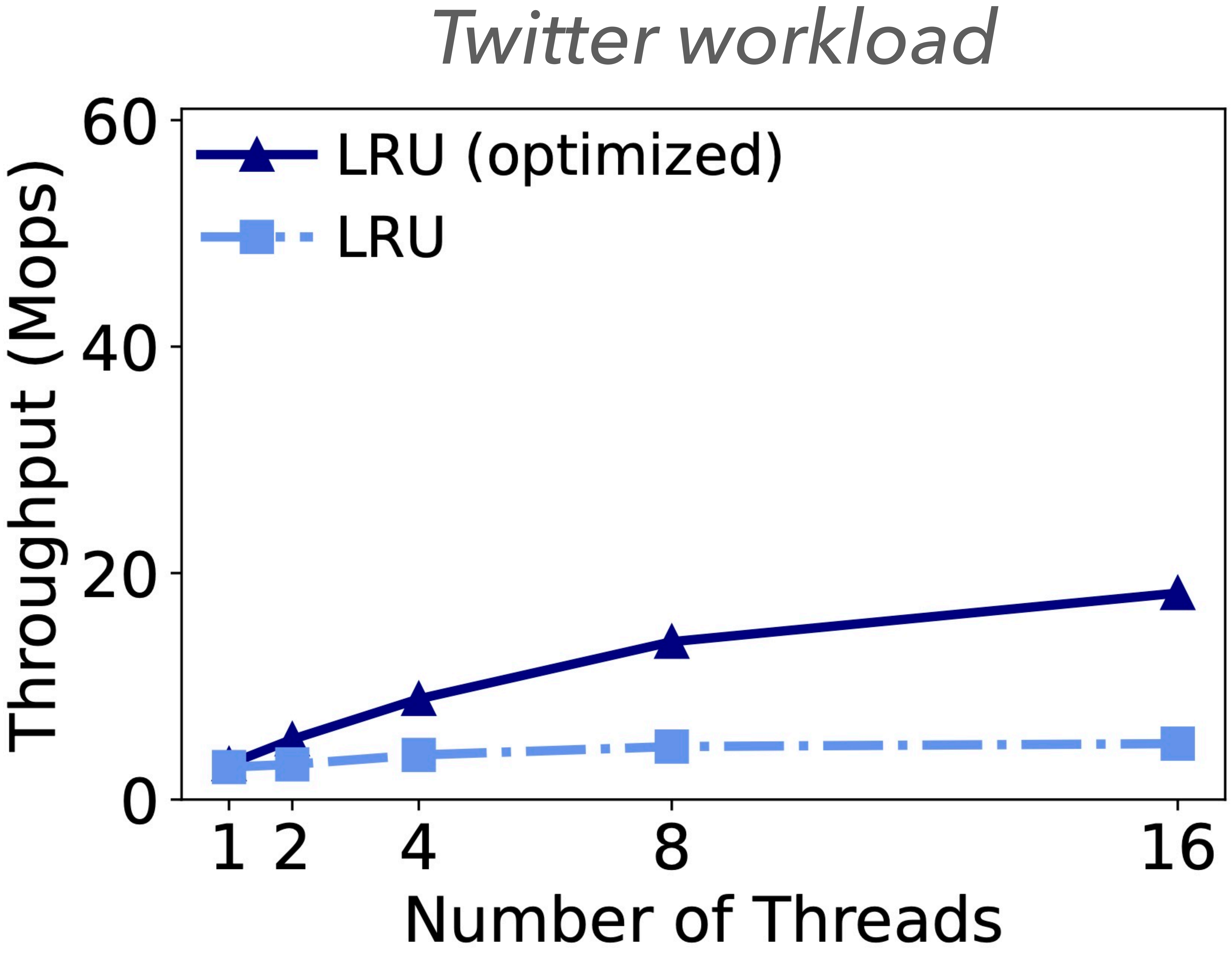
Throughput Measured in Cachelib



[OSDI'20] The CacheLib Caching Engine: Design and Experiences at Scale

Throughput Measured in Cachelib





[OSDI'20] The CacheLib Caching Engine: Design and Experiences at Scale

Cache Metrics

Efficiency

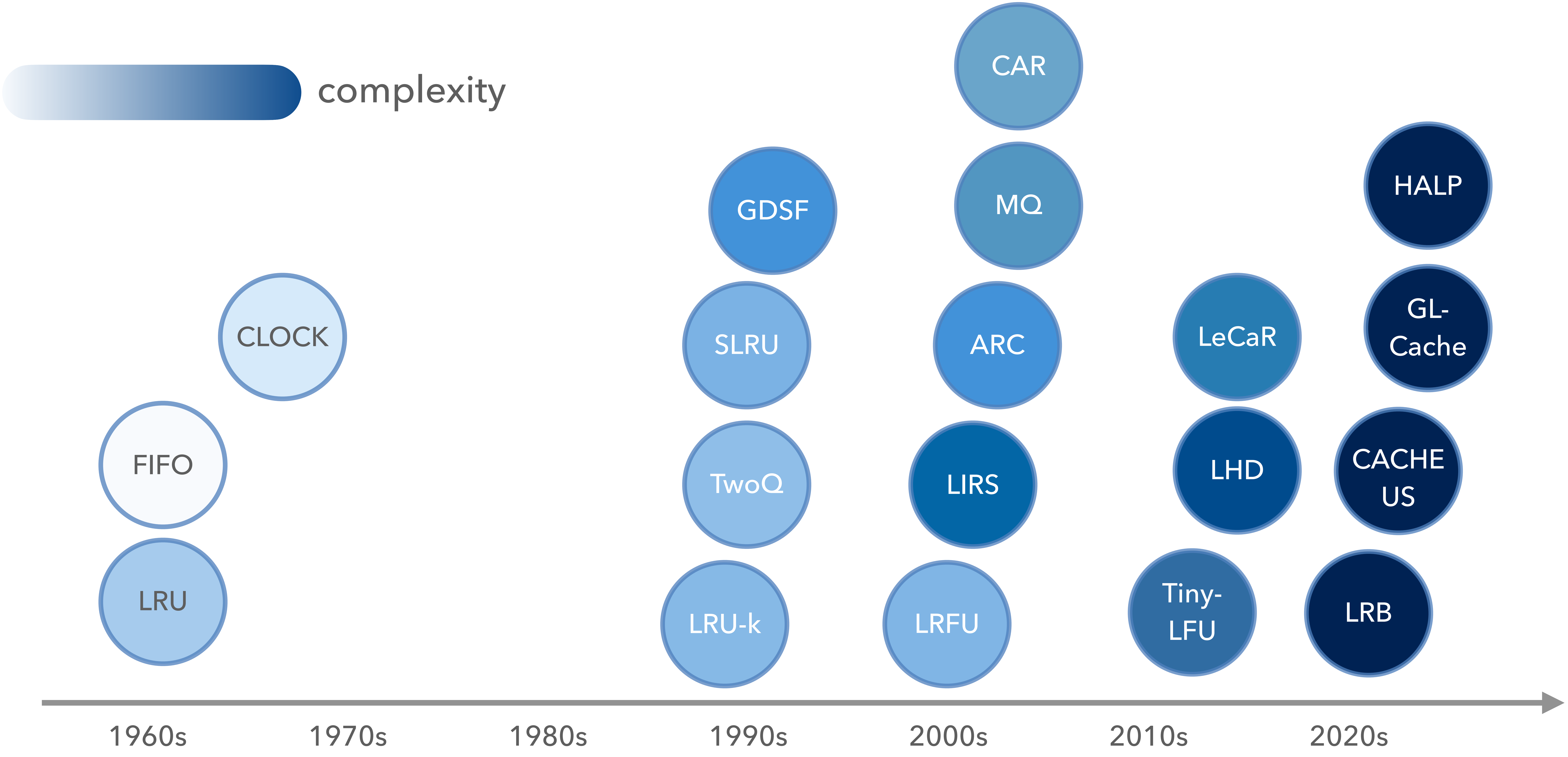
Cache Miss Ratio

Scalability

Reqs/Second

Simplicity

A Rich Literature of Eviction Algorithms



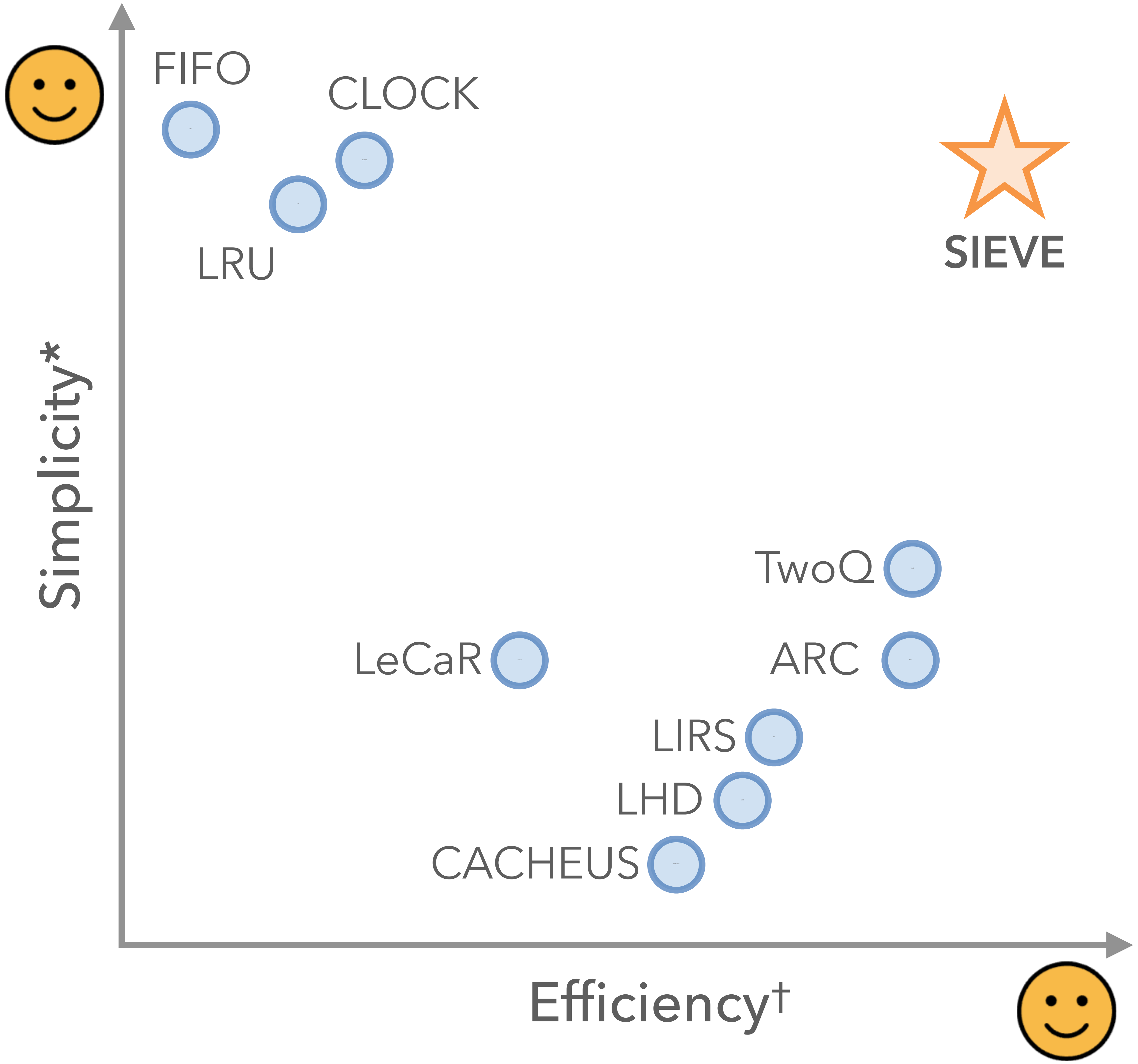
The Trouble with Complexity

- Difficult to debug and maintain
- Difficult to tune the parameters

*"Predicting which pages will be accessed in the near future is tricky, and the kernel has evolved many mechanisms to improve its chances of guessing right. But the kernel **not only often gets it wrong**, but also spends a lot of CPU time to make the incorrect choice."*

-- Linux kernel developer

SIEVE: a Simple and Efficient Cache Eviction Algorithm



* *Measured by lines of code*

† *Measured by average object miss ratio reduction from FIFO*

SIEVE Design

The Secret to Designing Efficient Eviction Algorithms

Lazy promotion

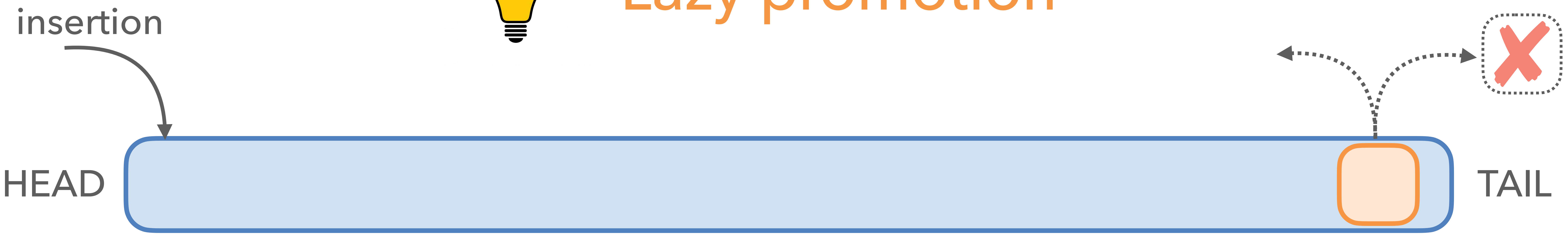
Quick demotion

[HotOS’23] FIFO queues can be better than LRU: the Power of Lazy Promotion and Quick Demotion

[SOSP’23] FIFO Queues are all You Need for Cache Eviction

The Secret to Designing Efficient Eviction Algorithms

Lazy promotion



Retain popular objects with minimal effort

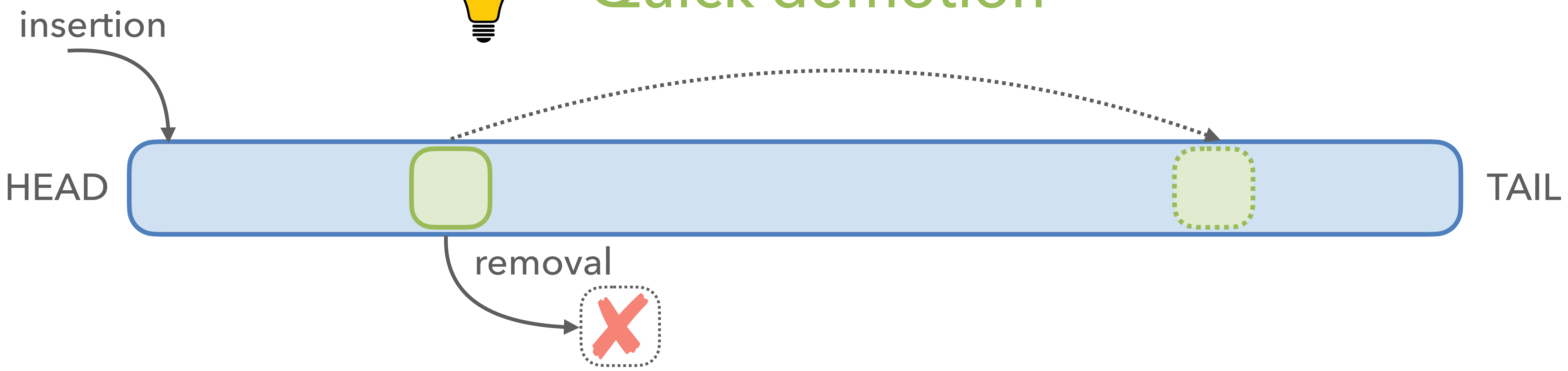
- Improve throughput due to less computation
- Improve efficiency due to more information at eviction

[HotOS'23] FIFO queues can be better than LRU: the Power of Lazy Promotion and Quick Demotion

[SOSP'23] FIFO Queues are all You Need for Cache Eviction

The Secret to Designing Efficient Eviction Algorithms

Quick demotion



Quickly remove most new objects,
such as one-hit-wonders (no request after insertion)

[HotOS'23] FIFO queues can be better than LRU: the Power of Lazy Promotion and Quick Demotion

[SOSP'23] FIFO Queues are all You Need for Cache Eviction

The Secret to Designing Efficient Eviction Algorithms

Lazy promotion

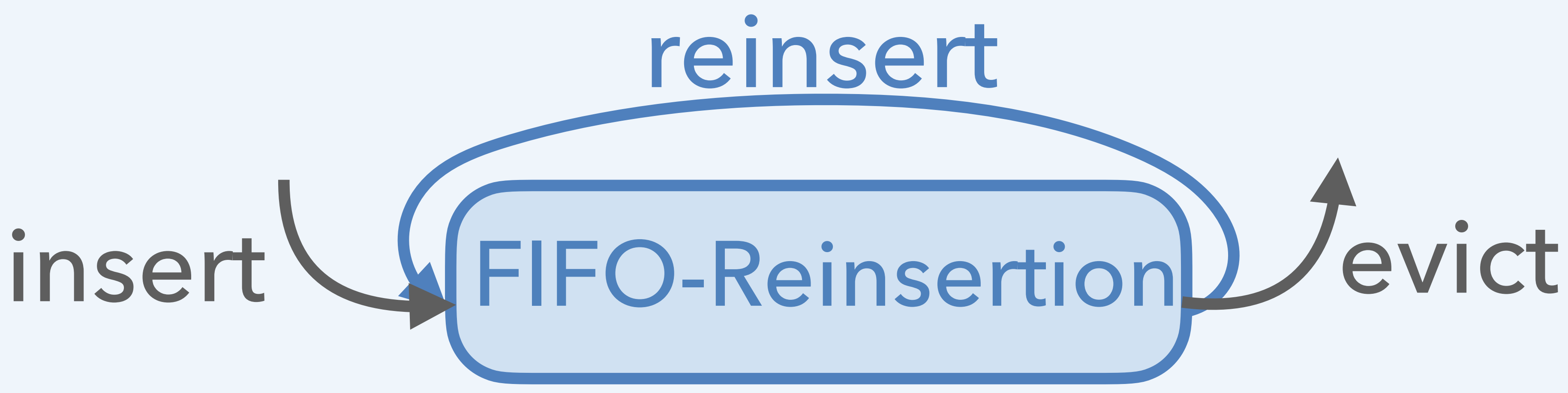
Retain popular objects with minimal effort

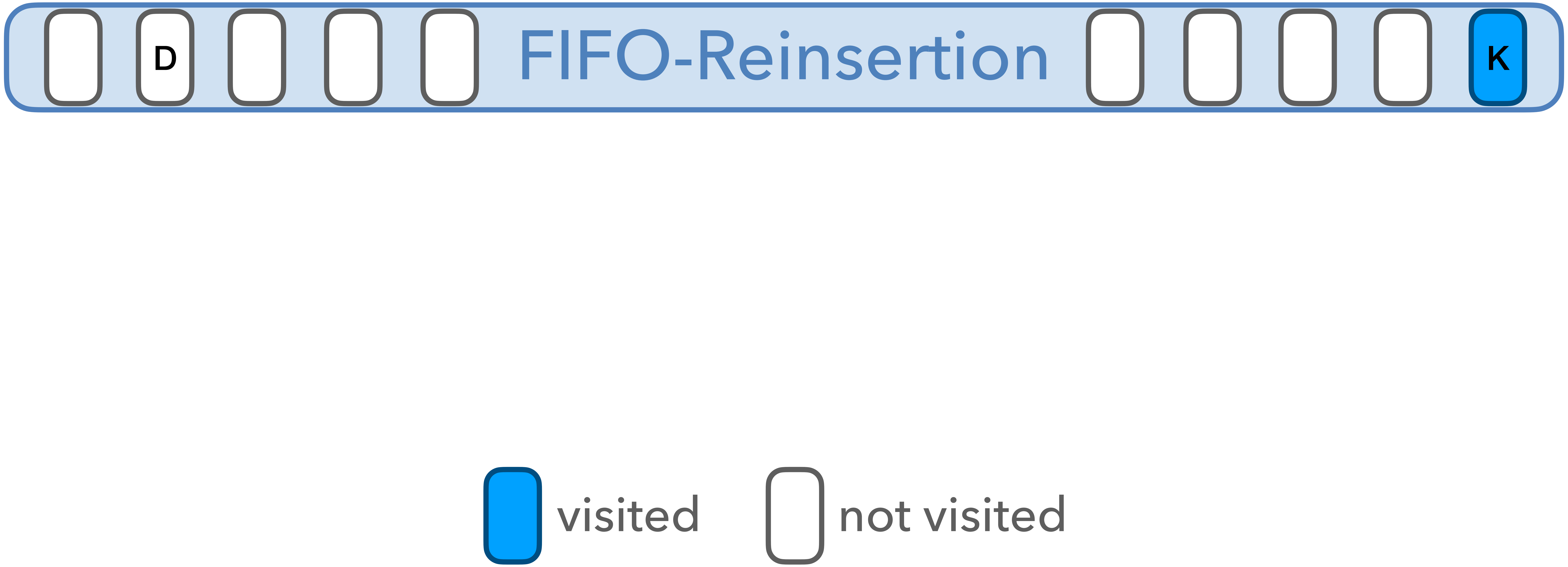
Quick demotion

Remove unpopular objects fast, such as one-hit-wonders

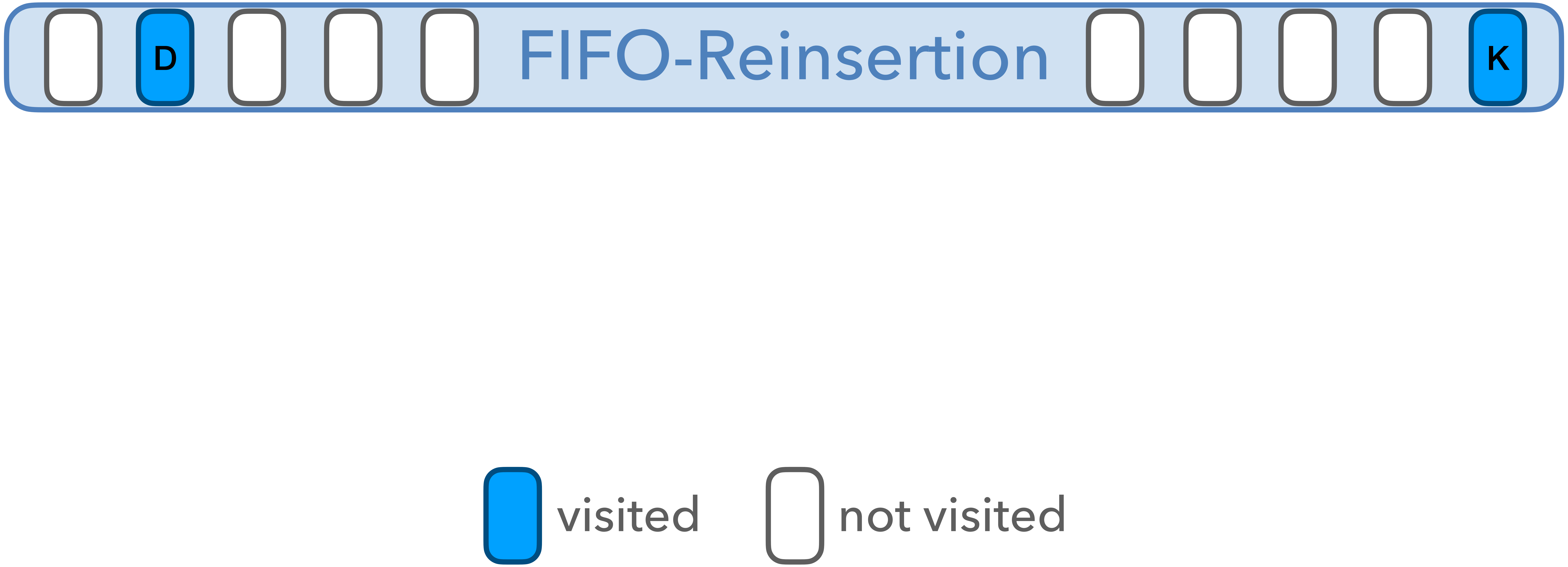
[HotOS’23] FIFO queues can be better than LRU: the Power of Lazy Promotion and Quick Demotion
[SOSP’23] FIFO Queues are all You Need for Cache Eviction

		Efficiency	Scalability
	Eager promotion No quick demotion	★★½	½
	No promotion No quick demotion	★	★★★★★

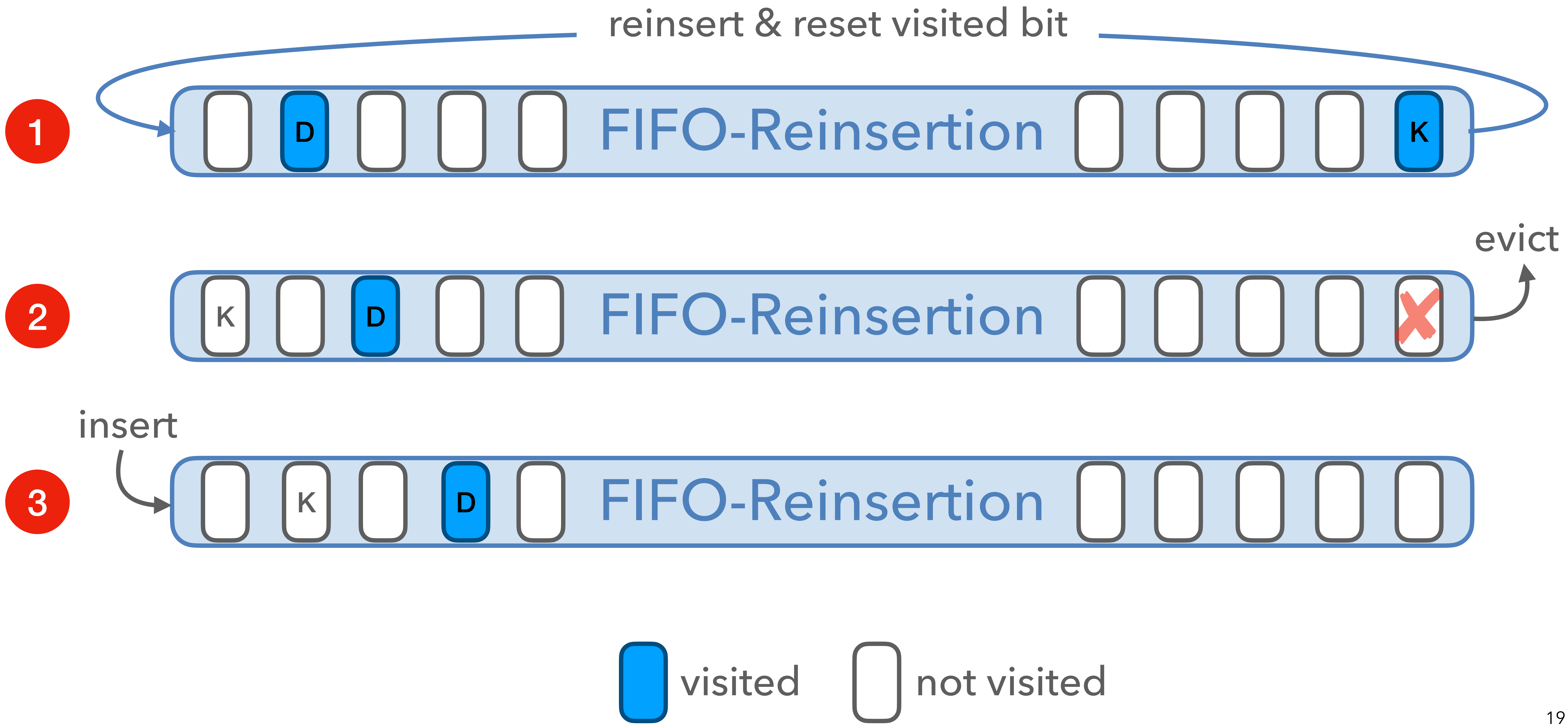
		Efficiency	Scalability
	Eager promotion No quick demotion		
	No promotion No quick demotion		
	Lazy promotion No quick demotion		


D
FIFO-Reinsertion
K
visited
not visited

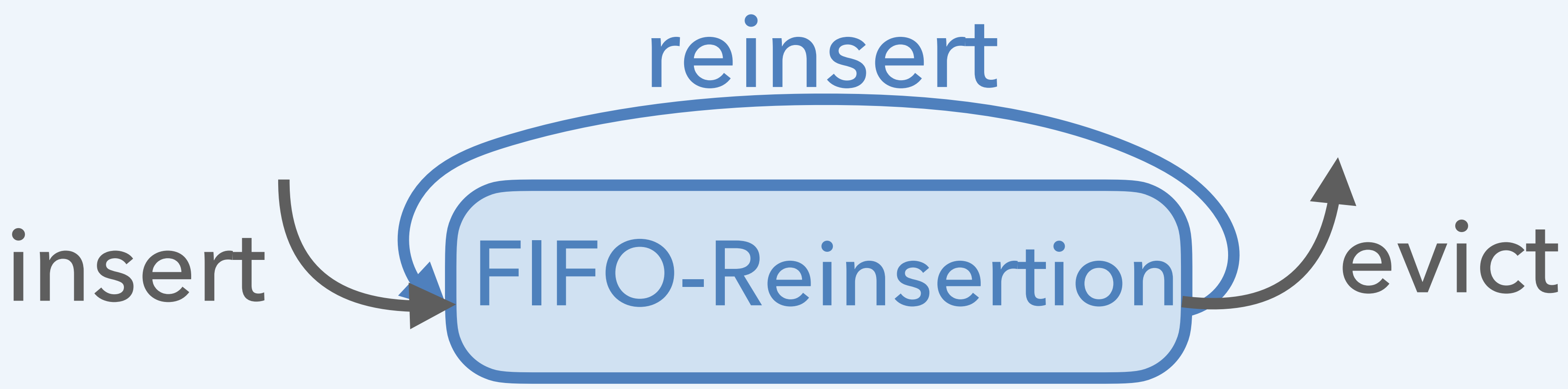
cache hit on D



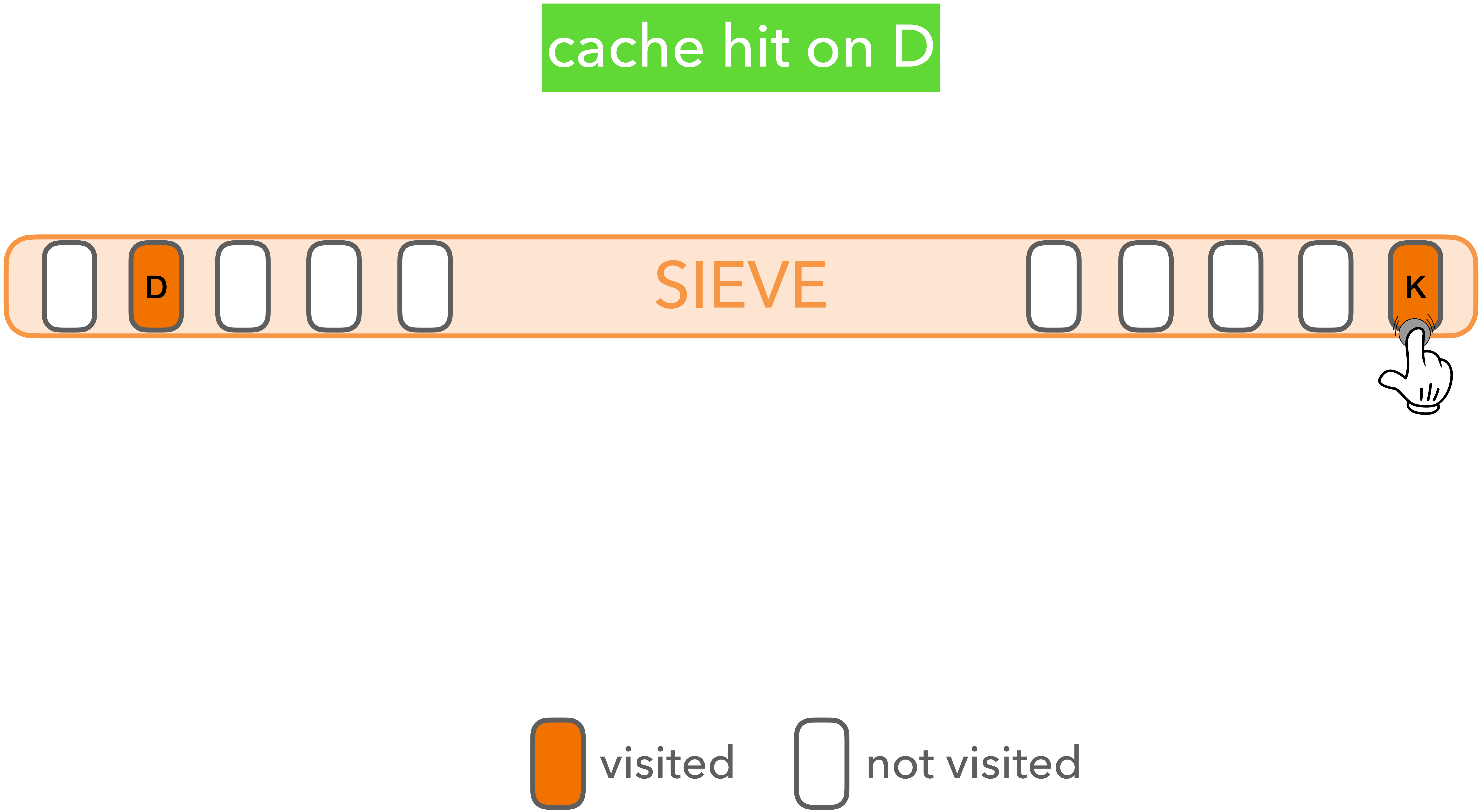
cache miss

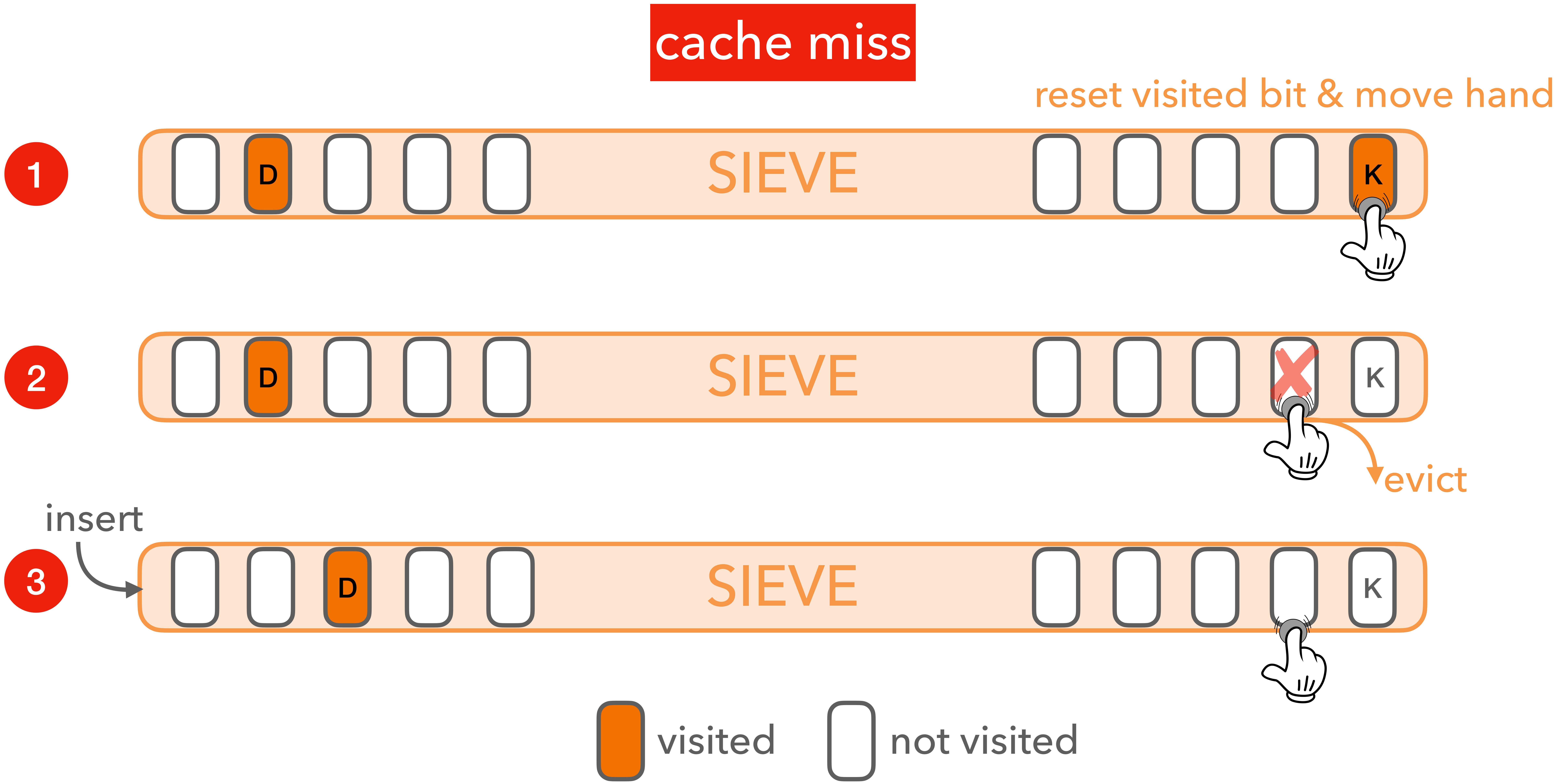

reinsert & reset visited bit
1
D
FIFO-Reinsertion
K
evict
2
K
D
FIFO-Reinsertion
insert
3
K
D
FIFO-Reinsertion
visited
not visited


Efficiency
Scalability
promote
insert
LRU
evict
Eager promotion
No quick demotion
insert
FIFO
evict
No promotion
No quick demotion
reinsert
insert
FIFO-Reinsertion
evict
Lazy promotion
No quick demotion

		Efficiency	Scalability
	Eager promotion No quick demotion	★★½	½
	No promotion No quick demotion	★	★★★★★
	Lazy promotion No quick demotion	★★★	★★★★½
			

visited not visited

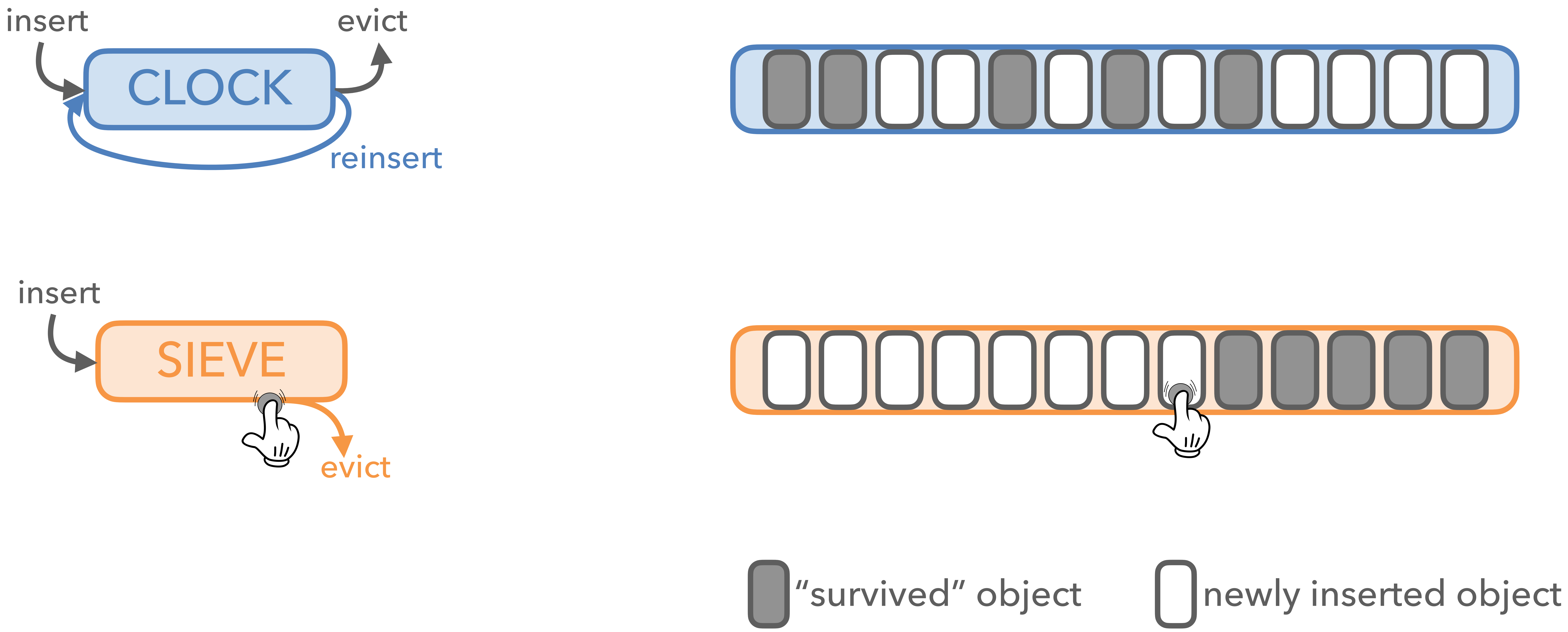

cache hit on D
D
SIEVE
K
visited
not visited


cache miss
reset visited bit & move hand
1
D
SIEVE
K
2
D
SIEVE
K
evict
insert
3
D
SIEVE
K
visited
not visited


Efficiency
Scalability
promote
insert
LRU
evict
Eager promotion
No quick demotion
insert
FIFO
evict
No promotion
No quick demotion
reinsert
insert
FIFO-Reinsertion
evict
Lazy promotion
No quick demotion
insert
SIEVE
evict
Lazy promotion

Quickly remove new objects

Separate new and old objects



[NSDI'21] SegCache: memory-efficient and high-throughput DRAM cache for small objects


Efficiency
Scalability
promote
insert
LRU
evict
Eager promotion
No quick demotion
insert
FIFO
evict
No promotion
No quick demotion
reinsert
insert
FIFO-Reinsertion
evict
Lazy promotion
No quick demotion
insert
SIEVE
evict
Lazy promotion
Quick demotion

SIEVE Evaluation

Web Cache Workloads

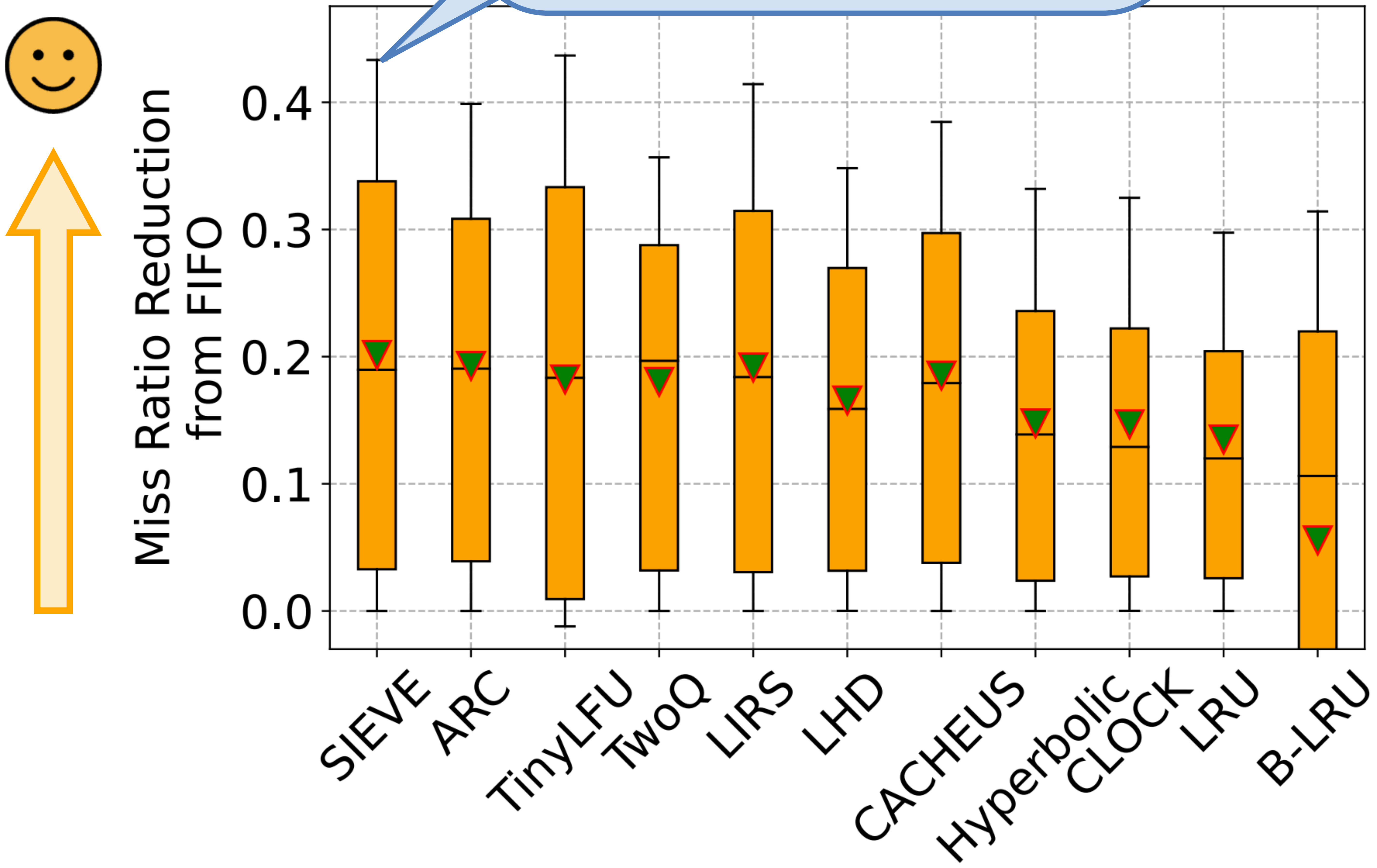
- Simulator: libCacheSim
- Prototype: Cachelib
- Testbed: Cloudlab

1559 traces | 247,017 million requests |14, 852 million objects

trace collection	collection time	#traces	cache type	# request (million)	# object (million)
CDN1	2021	1273	object	37,460	2,652
CDN2	2018	219	object	3,728	298
Tencent Photo	2018	2	object	5,650	1,038
Wiki CDN	2019	3	object	2,863	56
Twitter KV	2020	54	KV	195,441	10,560
Meta KV	2022	5	KV	1,644	82
Meta CDN	2023	3	object	231	76

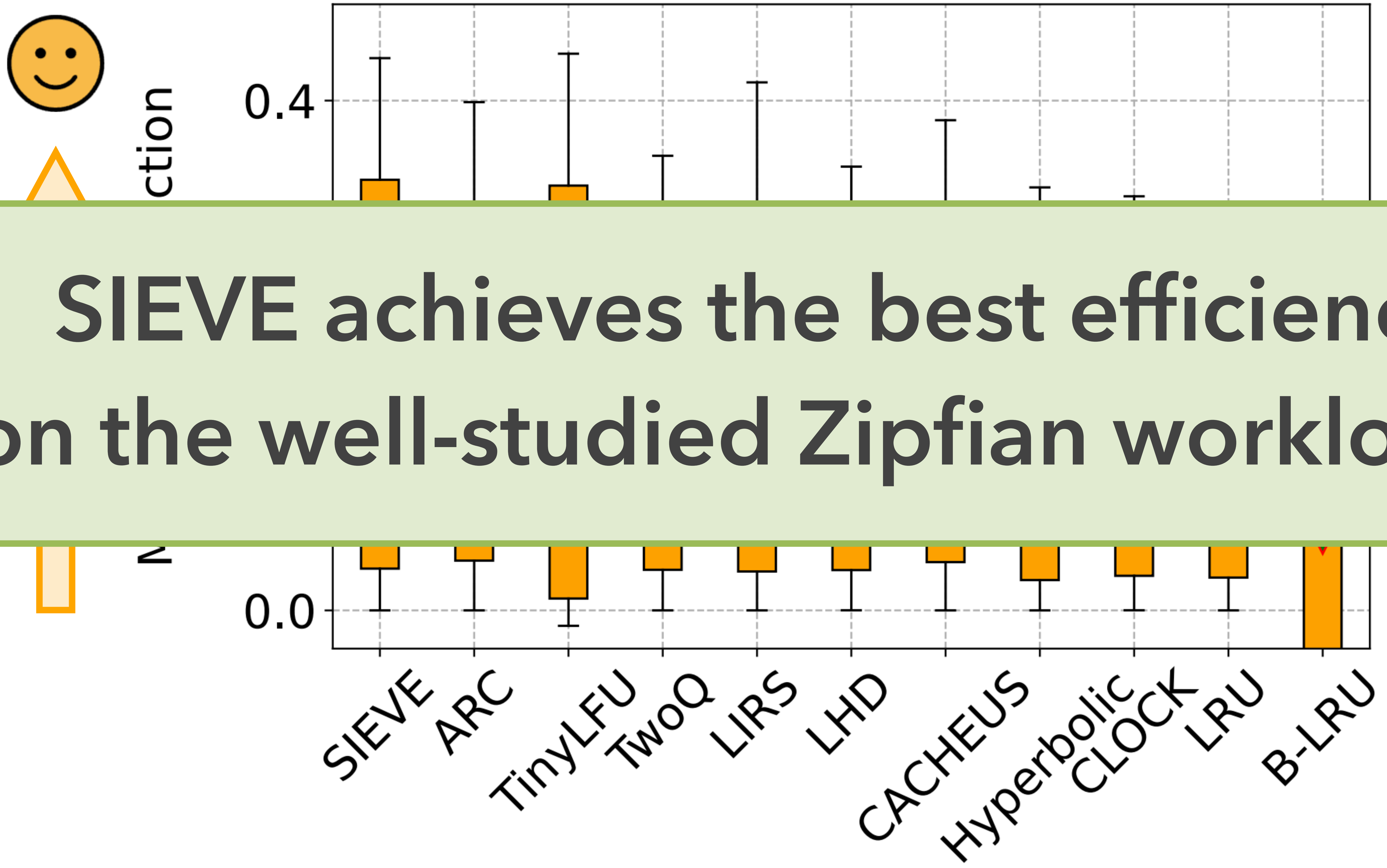
SIEVE: Efficiency

SIEVE reduces FIFO's miss ratio by more than 42% on 10% of the traces (top whisker) with a mean of 21%



CDN1, 1273 traces (37,460 million requests)

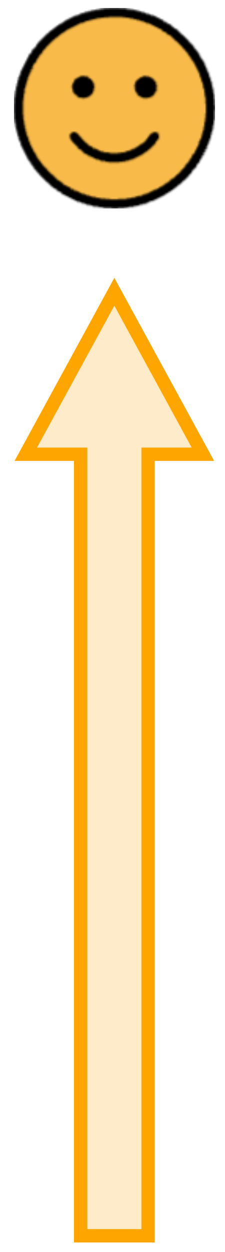
SIEVE: Efficiency

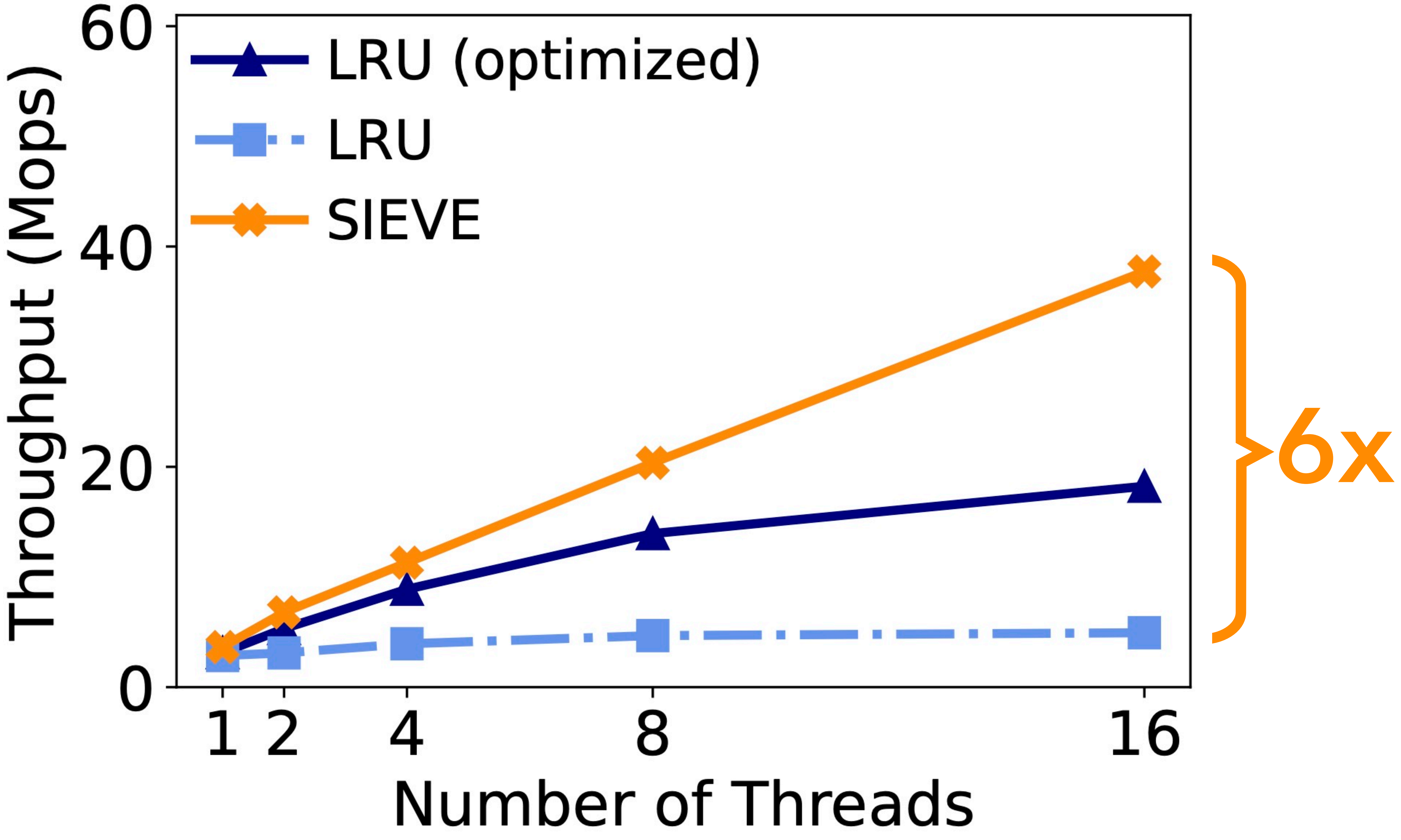


SIEVE achieves the best efficiency
on the well-studied Zipfian workloads

CDN1, 1273 traces (37,460 million requests)

SIEVE: Throughput





SIEVE: Simplicity

Cache library	Language	Lines of change
groupcache	Golang	21
mnemonist	Javascript	12
lru-rs	Rust	16
lru-dict	Python + C	21

SIEVE: Simplicity

Cache library	Language	Lines of change
groupcache	Golang	21
mnemonist	Javascript	12
lru-rs	Rust	16
lru-dict	Python + C	21

Adoption

Large systems: Pelikan Nyrkiö SkiftOS DragonFly DNSCrypt-proxy encrypted-dns-resolver

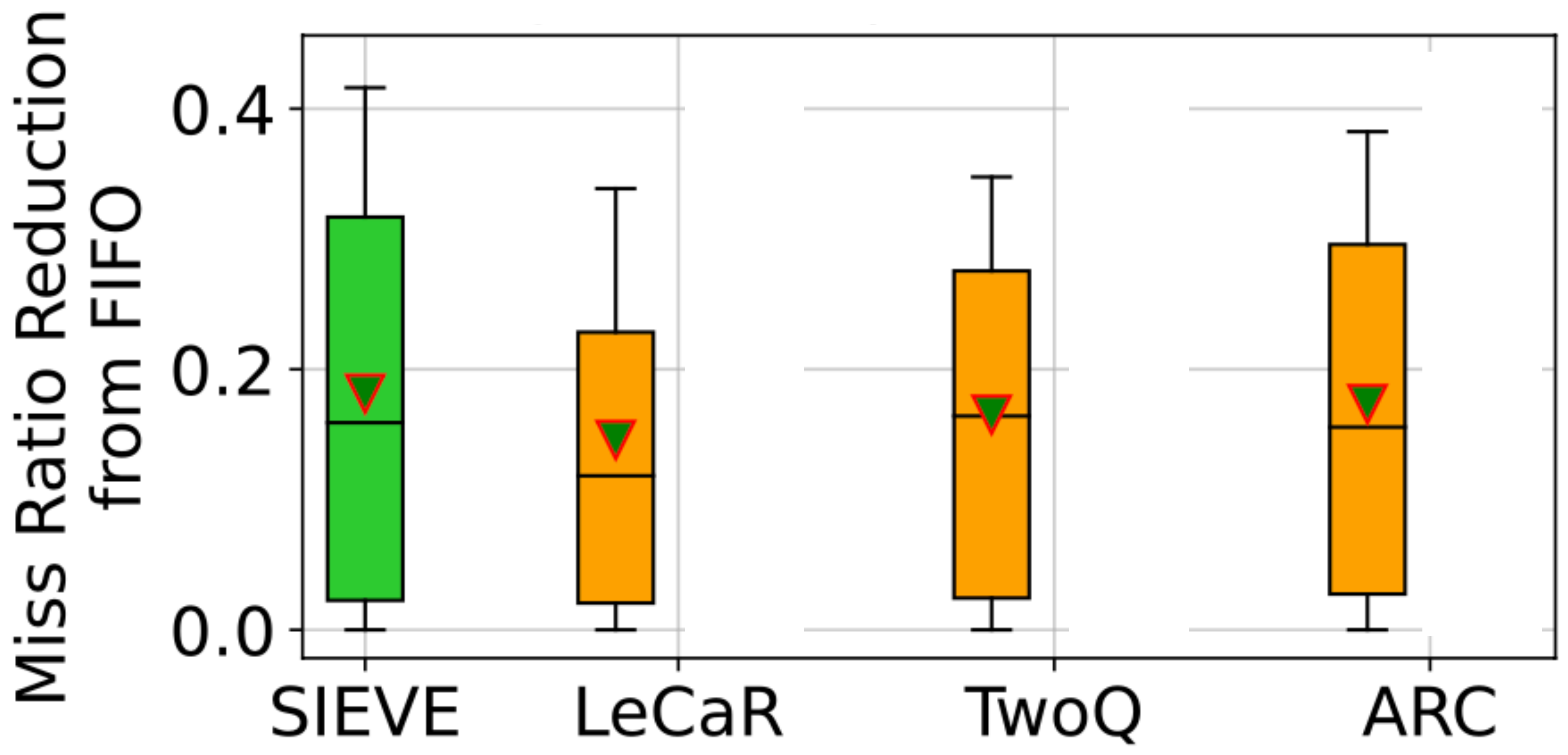
Cache libraries: golang-fifo js-sieve rust-sieve-cache go-sieve sieve_cache (Ruby) zig-sieve (Zig) sieve (Swift) sieve (JavaScript) sieve (Elixir) sieve (Nim) sieve-cache (Java) sieve (Python) sieve-cache-in-rust sieve-cache (JavaScript) gosieve, sieve (typescrpt)

SIEVE: Primitive

LeCaR: LRU + LFU + ML

TwoQ: LRU + FIFO

ARC: LRU + LRU + 2 ghost queues



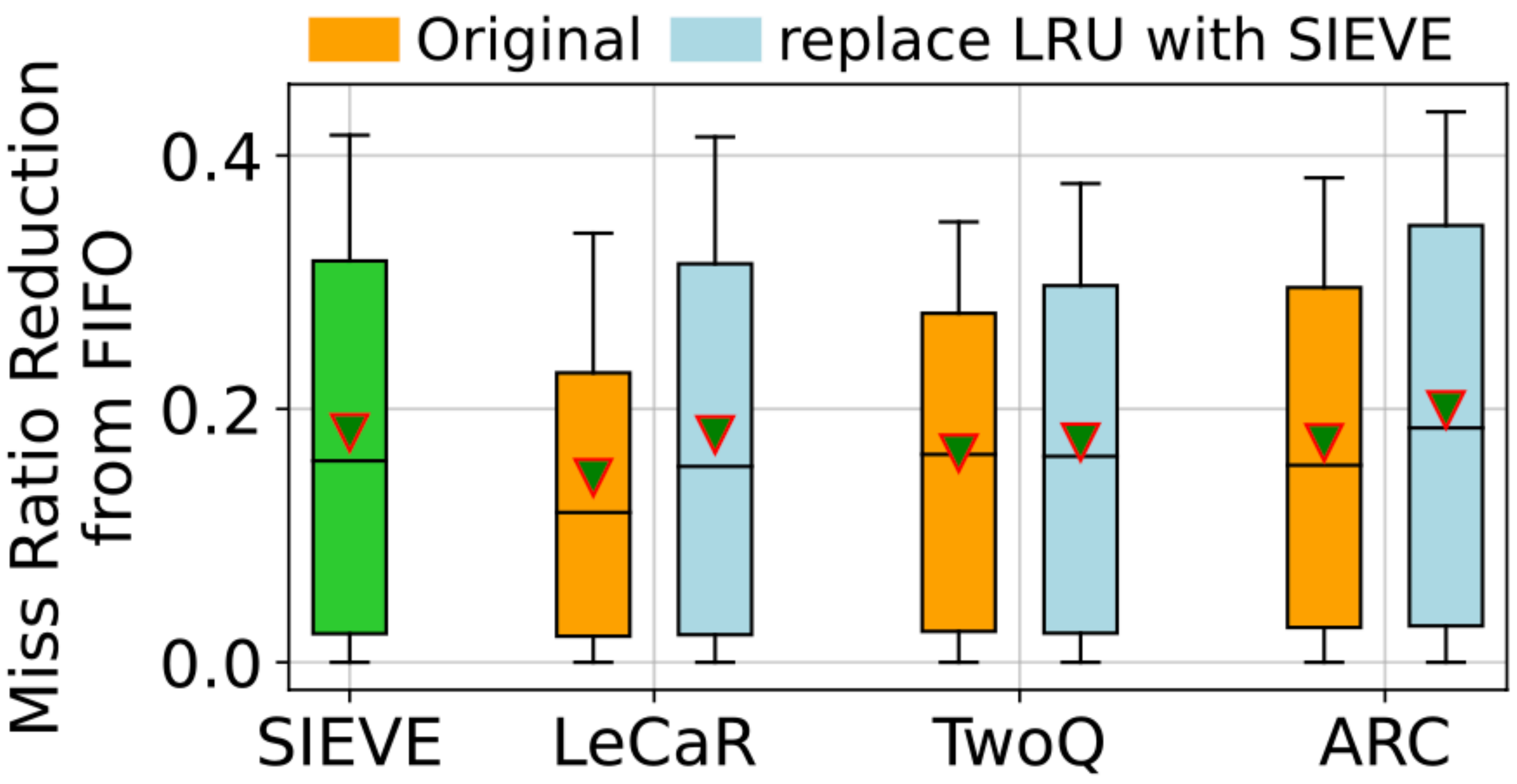
SIEVE: Primitive

LeCaR: LRU + LFU + ML

TwoQ: LRU + FIFO

ARC: LRU + LRU + 2 ghost queues

Replace LRU with SIEVE

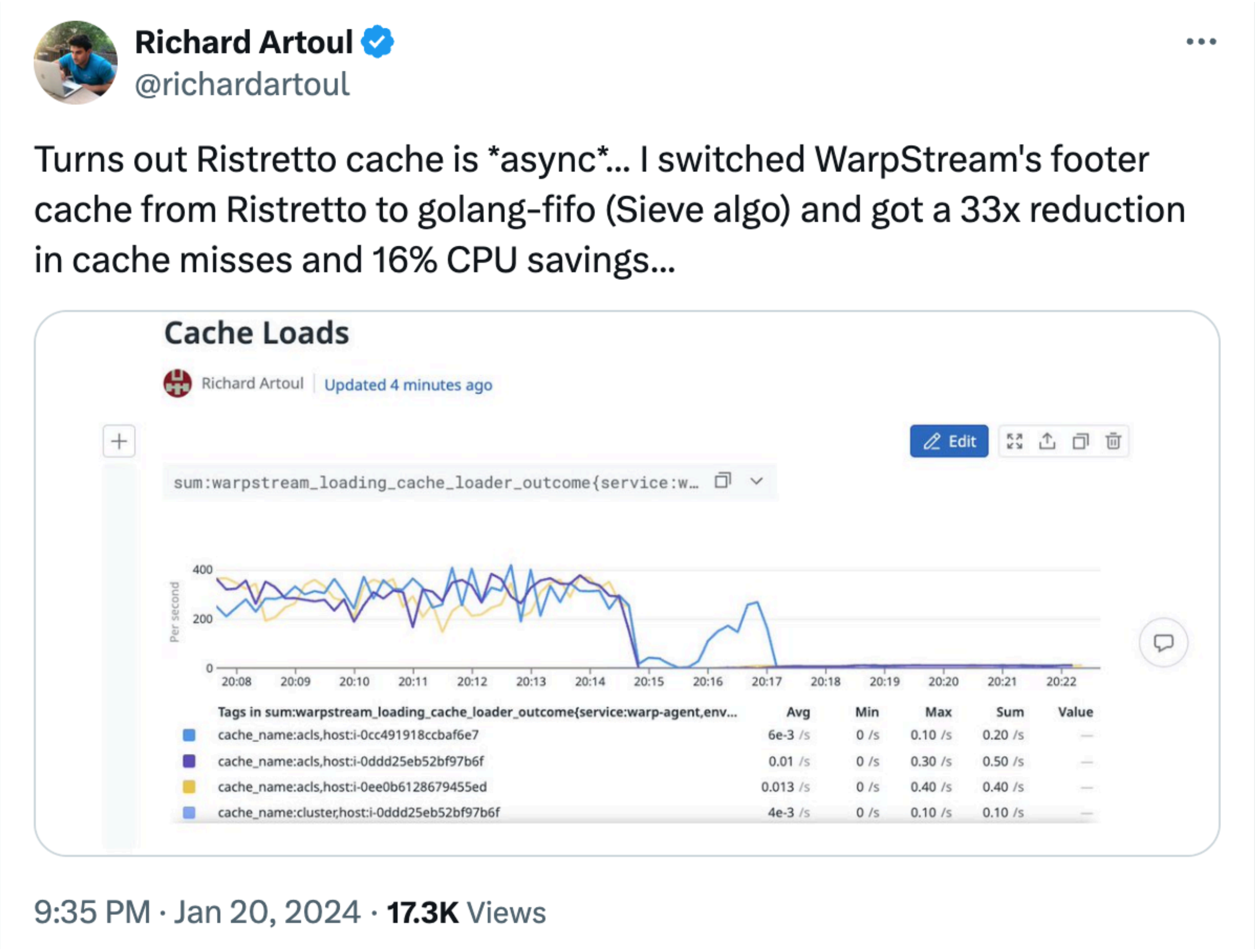


More in the paper

- Why SIEVE is effective
- Byte miss ratio
- When SIEVE is not effective
- Comparison to ML algorithms

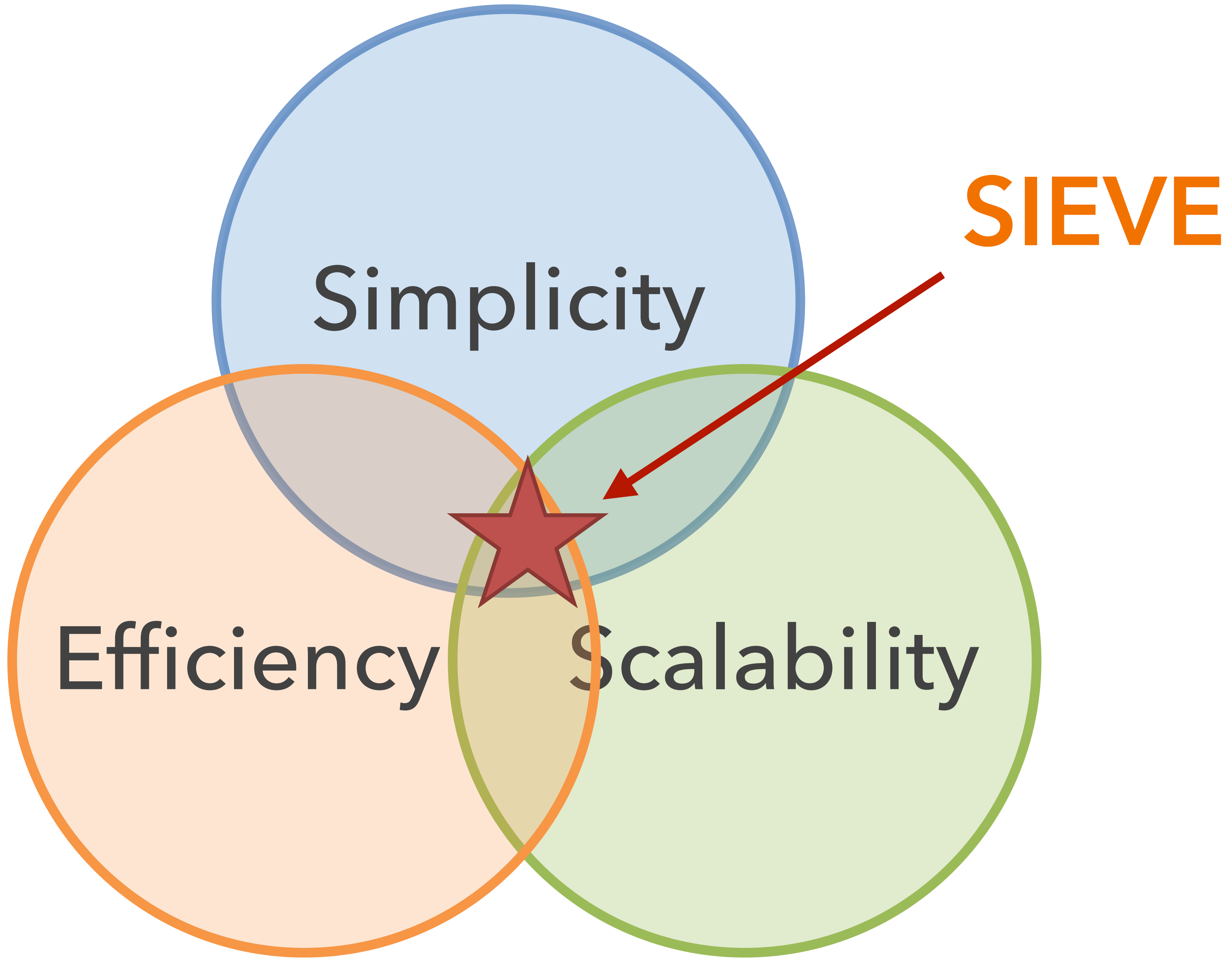
SIEVE Adoption

- SIEVE is available in over **20** cache libraries with **10+** programming languages
- Production systems start integrating SIEVE: Pelican, SkiftOS, DragonFly, and etc



Takeaway

- Lazy promotion and quick demotion are key to efficient eviction algorithm
- SIEVE uses a moving hand to 1) retain popular objects in place, and 2) remove unpopular objects quickly
- The simplest algorithm with state-of-the-art efficiency and scalability



https://sievecache.com

Backup Slides

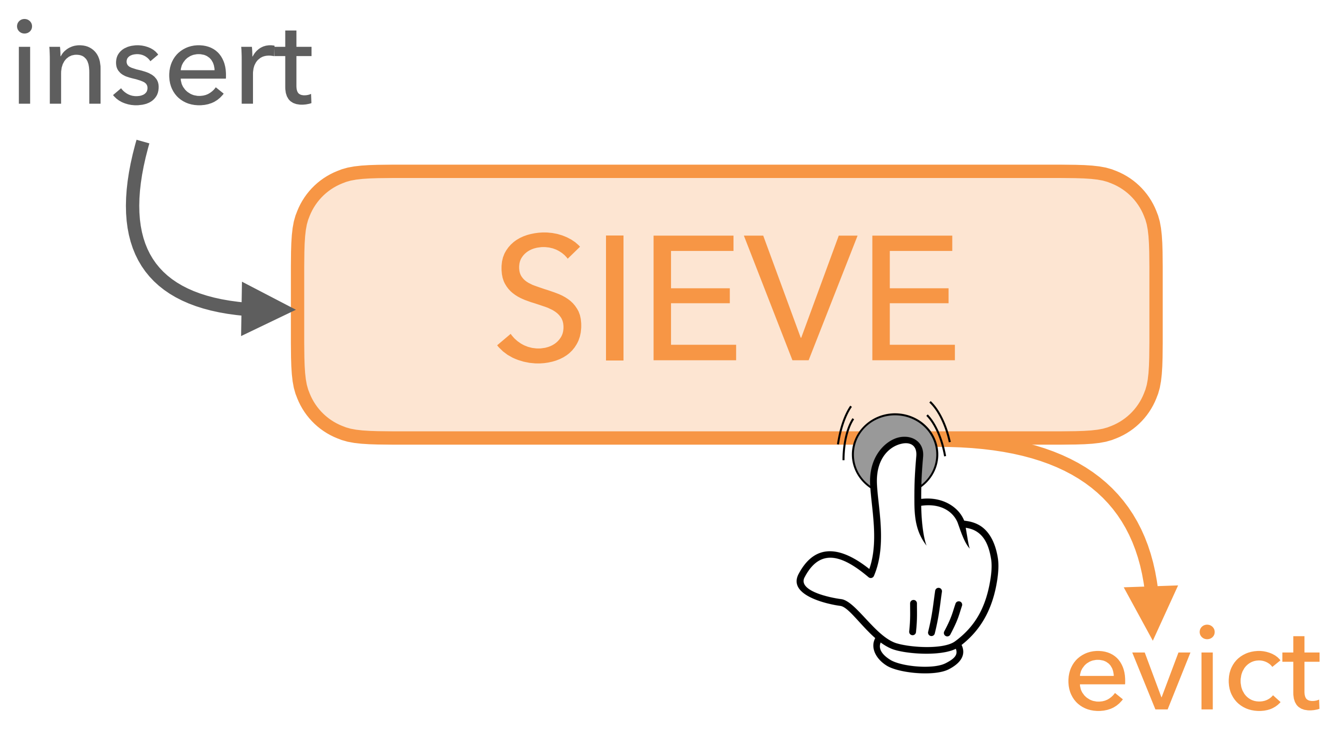
No promotion

No quick demotion

FIFO **cannot keep popular** objects in the cache

Lazy promotion

ClOCK **retains popular** objects by reinserting them



Lazy promotion

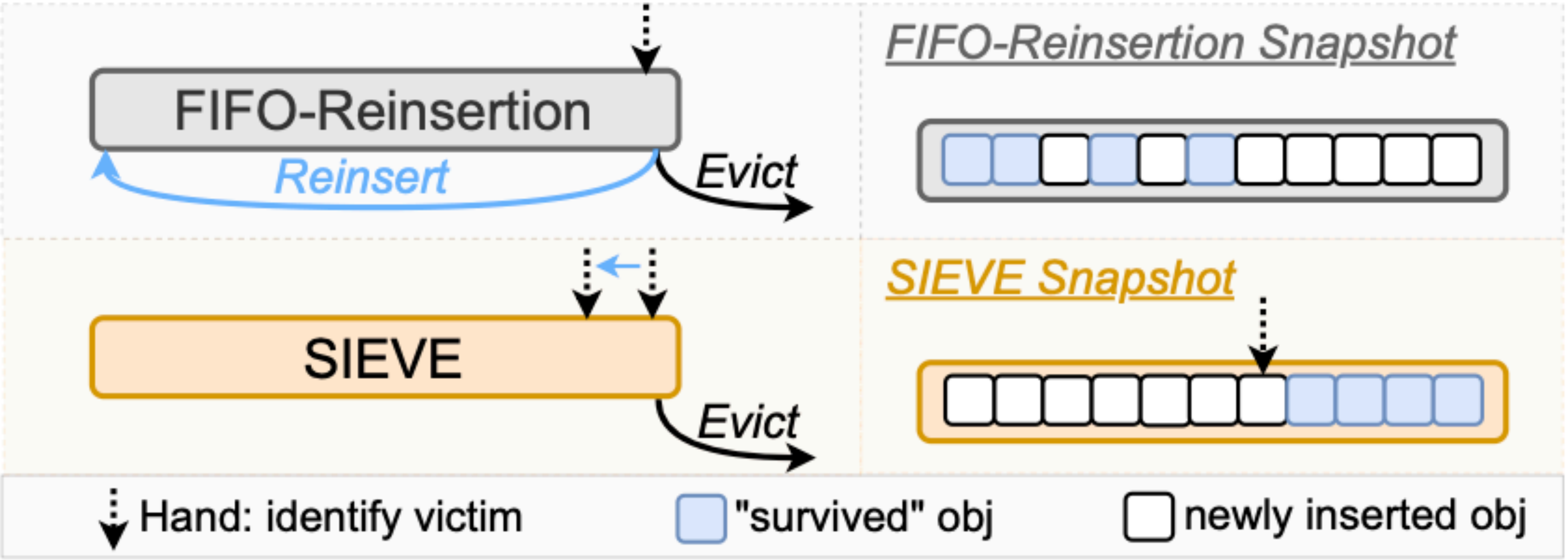
Quick demotion

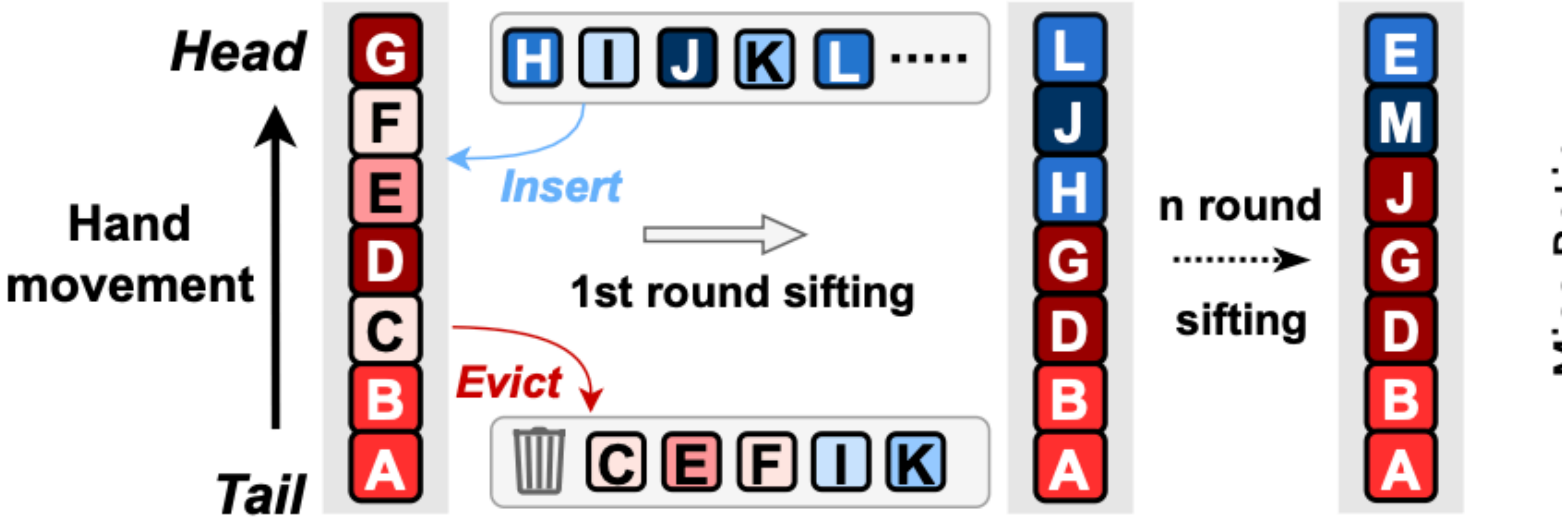
SIEVE **retains popular objects** by keeping them in place and **quickly removes unpopular objects** by using a moving hand

```
|head(new)--[hand]--tail(old)|

# FIFO-Resertion / CLOCK
obj = tail
while obj.visited:
    obj.visited = false
    prev = obj.prev
    move obj to head
    obj = prev

# SIEVE
obj = hand
while obj.visited:
    obj.visited = false
    # skip obj, do nothing
    obj = obj.prev
hand = obj.prev
```





(a) Density of colors indicates inherent object popularity (blue: newly inserted objects; red: old objects in each round), and the letters represent object IDs. The first queue captures the state at the start of the first round, and the second queue captures the state at the end of the first round.

FIFO

CLOCK

LRU

ARC

LIRS

LHD

CACHE US

SIEVE is Simpler than LRU:

An Efficient Turn-Key Eviction Algorithm for Web Caches

Yazhuo Zhang, Juncheng Yang, Yao Yue, Ymir Vigfusson, K.V. Rashmi