FIFO Queues

are all You Need for Cache Eviction

Juncheng Yang

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

Carnegie Mellon University

Carnegie Mellon
Parallel Data Laboratory

EMORY
UNIVERSITY

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
  - simplicity
- A core component of cache design: **eviction**

traffic server™

VARNISH CACHE

Cachelib

Pelikan

ALLUXIO

A long history of research centered around LRU

- Least-recently-used (LRU)
  - maintains objects in a queue with last-access order
  - metadata update (with locking) on each read request
- Problems
  - not scalable
  - not scan-resistant

A long history of research centered around LRU

- Improve LRU's efficiency
  - **add more techniques/queues/metrics**: ARC[FAST'03], LIRS[SIGMETRICS'02], LRU-K[SIGMOD'93], 2Q[VLDB'94], MQ[ATC'01], TinyLFU[TOS'17], LRB[NSDI'20], CACHEUS[FAST'21]…
- Improve LRU's throughput and scalability
  - **reduce #operations per-request**: relaxed LRU, CLOCK variants
- **Existing works: tradeoff between efficiency and throughput**

An alternative: FIFO eviction algorithm

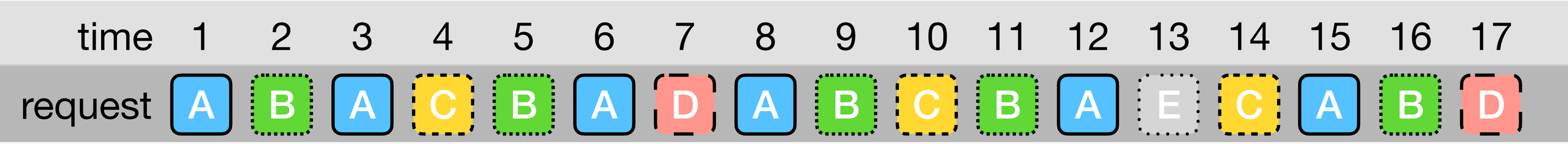
- First-in-first-out (FIFO)
  - simpler
  - fewer metadata
  - less computation
  - more scalable
  - flash friendly

Can we design an efficient FIFO-based algorithm?

Observation

More one-hit wonders than you would have expected

- One-hit wonder: objects appeared once in the sequence
- Zipfian workloads: One-hit-wonder ratio **decreases with sequence length (measured in #obj)**

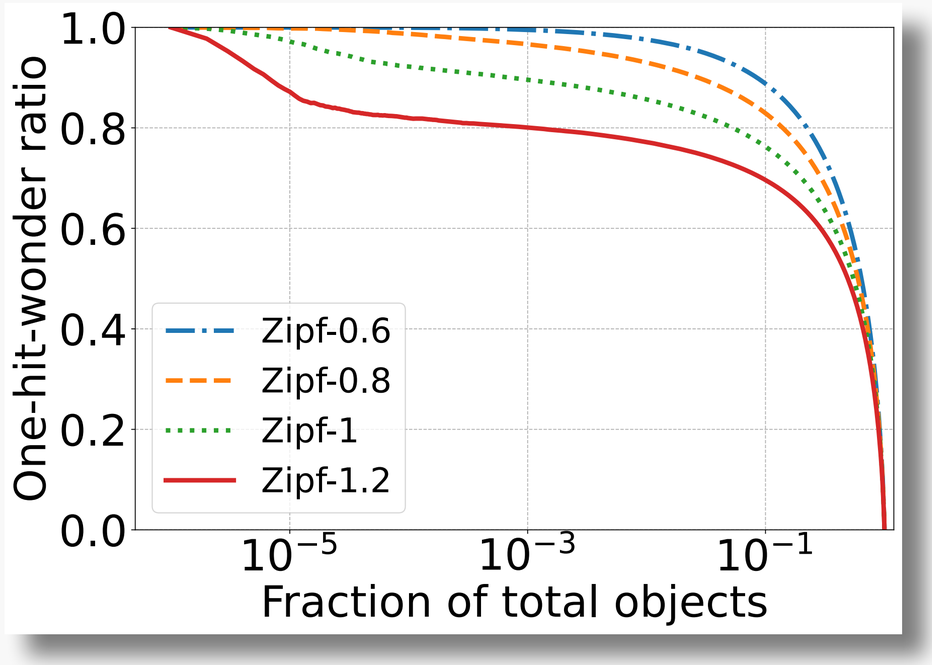


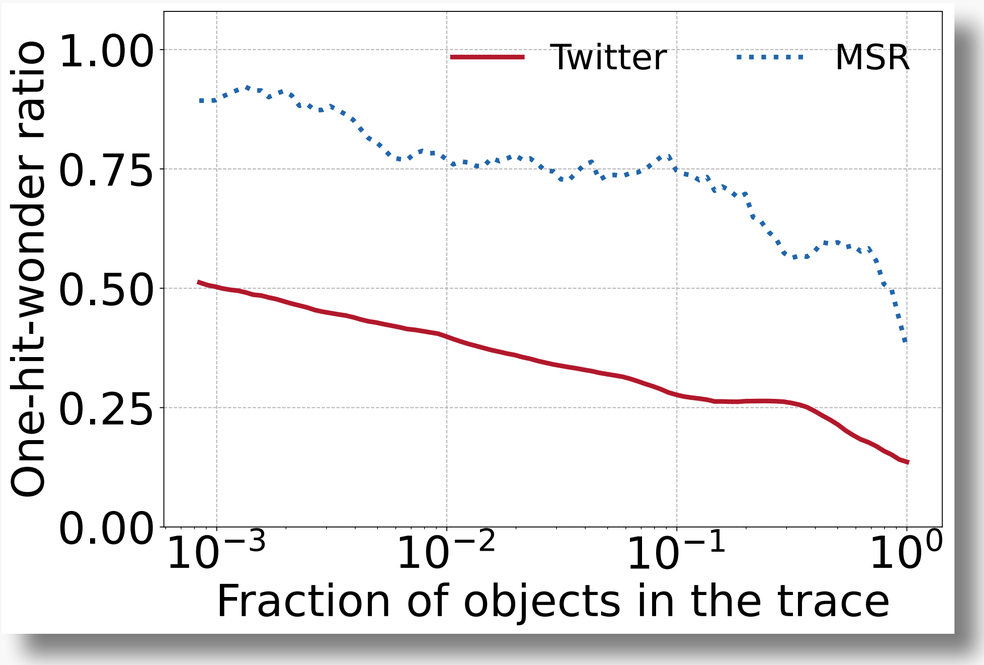
start time	end time	sequence length (# objects)	# one-hit wonder	one-hit wonder ratio
1	17	5	1 (E)	20%
1	7	4	2 (C, D)	50%
1	4	3	2 (B, C)	66%

Observation

More one-hit wonders than you would have expected

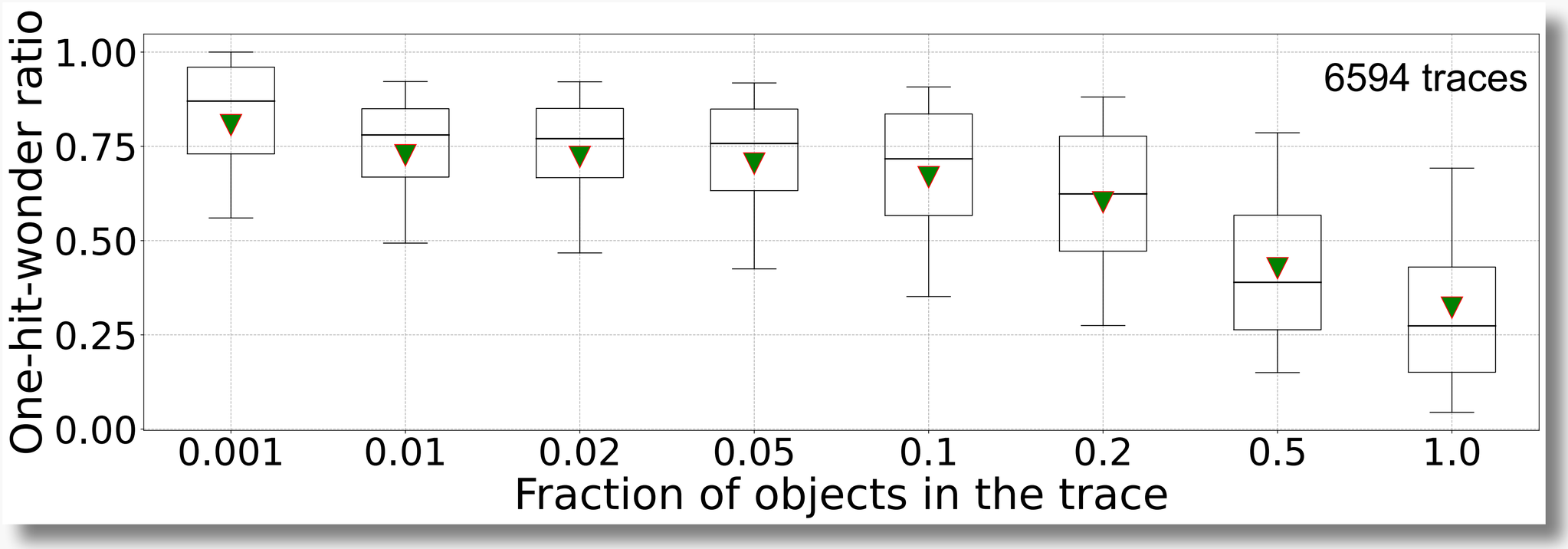
- One-hit wonder: objects appeared once in the sequence
- Zipfian workloads: One-hit-wonder ratio **decreases with sequence length (measured in #obj)**





Observation

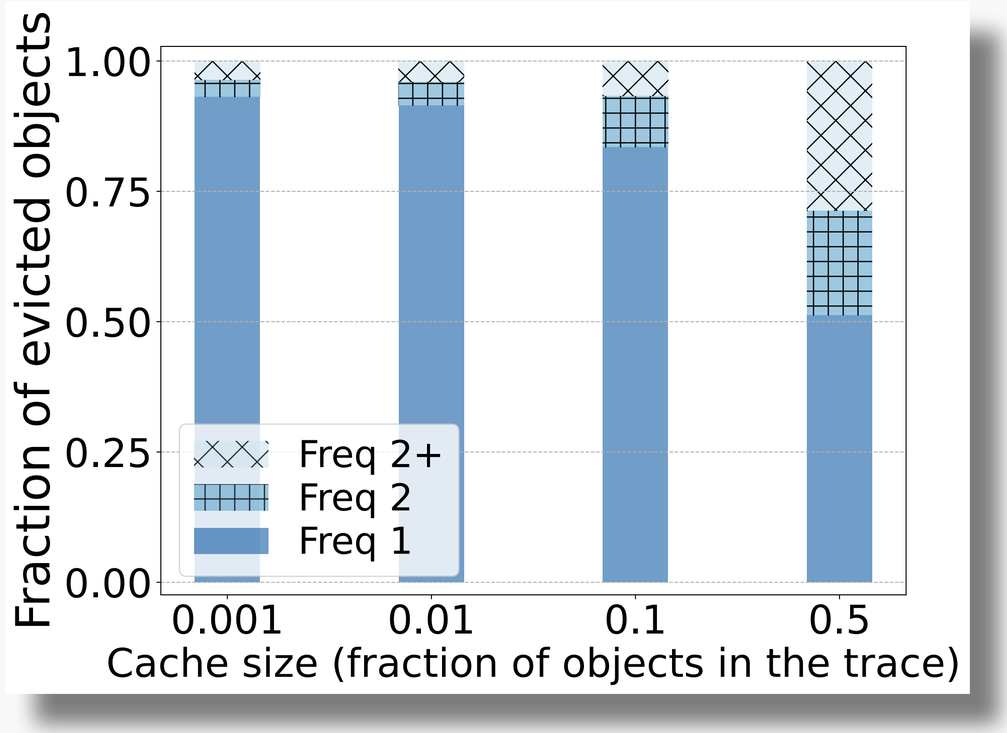
More one-hit wonders than you would have expected



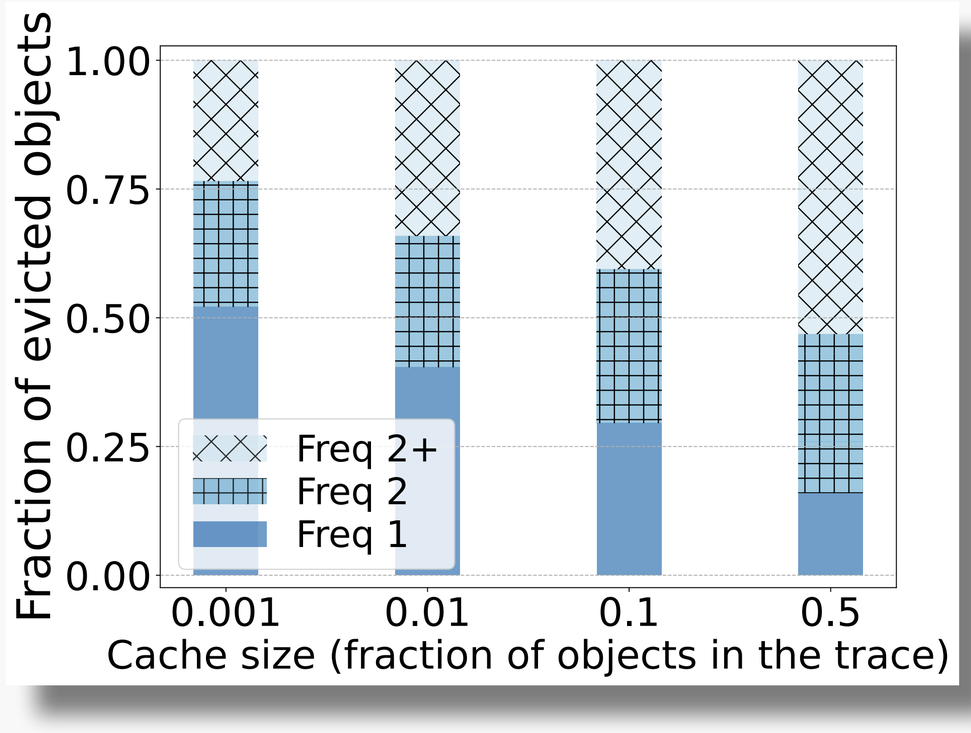
A cache starts eviction after a short request sequence
=> **most objects in the cache are one-hit wonders**

Observation

Many objects in the cache are one-hit-wonders



LRU cache running MSR workload



LRU cache running Twitter workload

A cache starts eviction after a short request sequence
=> **most objects in the cache are one-hit wonders**

S3-FIFO Design

Simple, Scalable eviction algorithm with three Static FIFO queues

https://s3fifo.com

https://blog.s3fifo.com

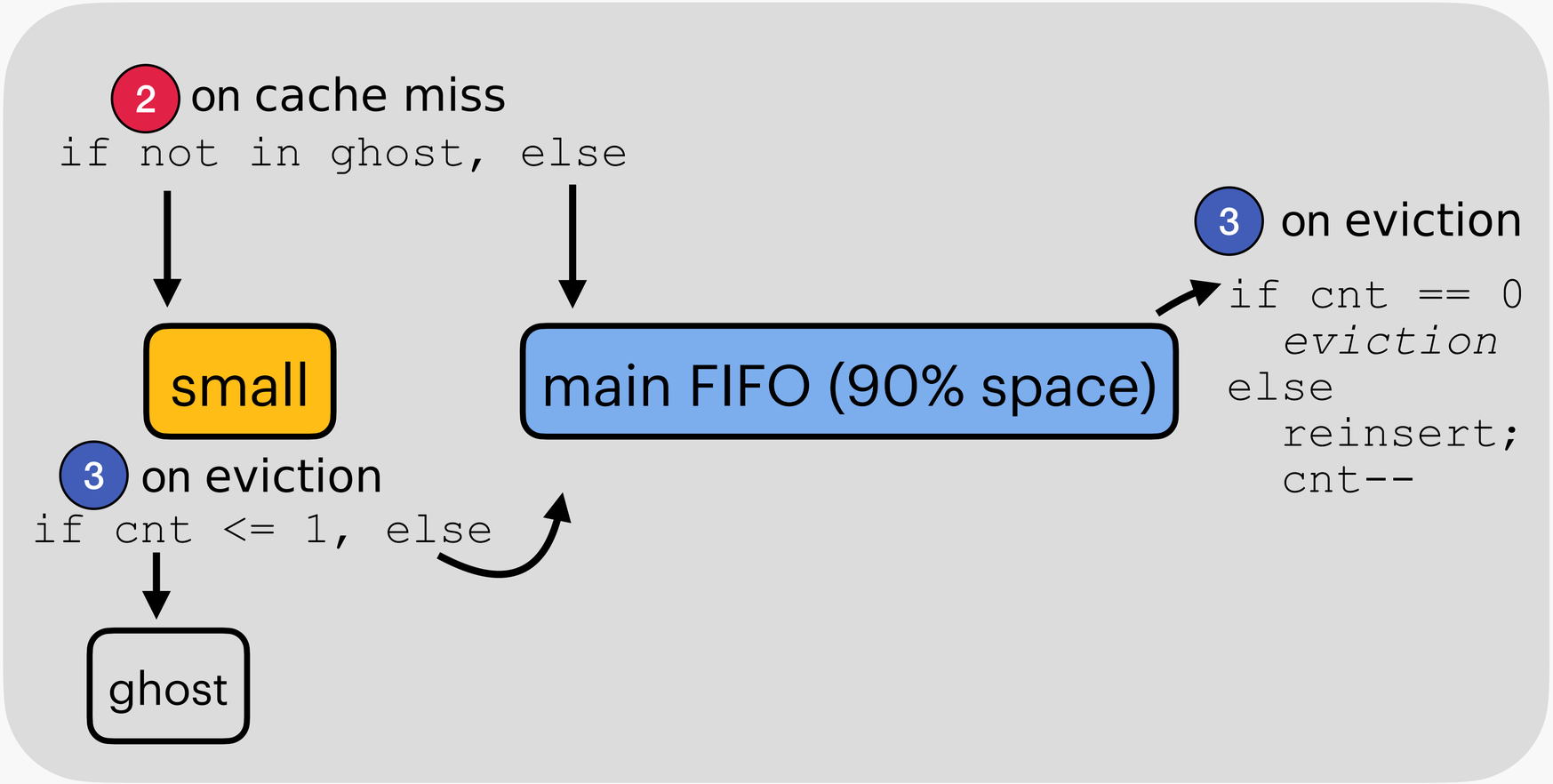
S3-FIFO design

Simple, Scalable eviction algorithm with three Static FIFO queues

```
struct object {
  …
  uint8_t cnt:2;
}
```

1 on cache hit

```
cnt++
```



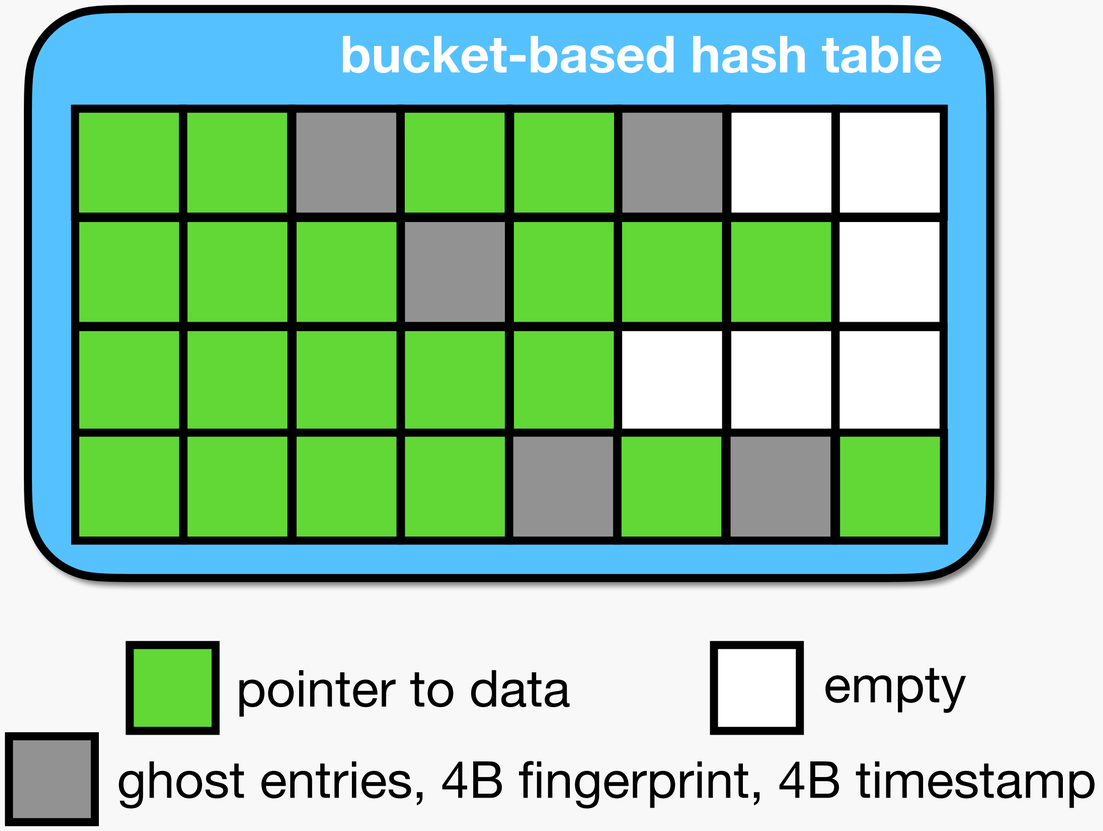
*using 1-bit flag also works for most workloads

S3-FIFO features

- **Simple and robust:** static queues
- **Fast**: no metadata update for most requests
- **Scalable**: no lock
- **Tiny metadata**: 2 bits
- **Flash friendly**: sequential writes

S3-FIFO implementation

- Can implement with one, two or three FIFO queues
- The small and main FIFO queues can be merged
- Ghost FIFO can be implemented as part of the index (e.g., hash table)
  - in the queue => T_{curr} - T_{insert} < S_{ghost}



S3-FIFO evaluation

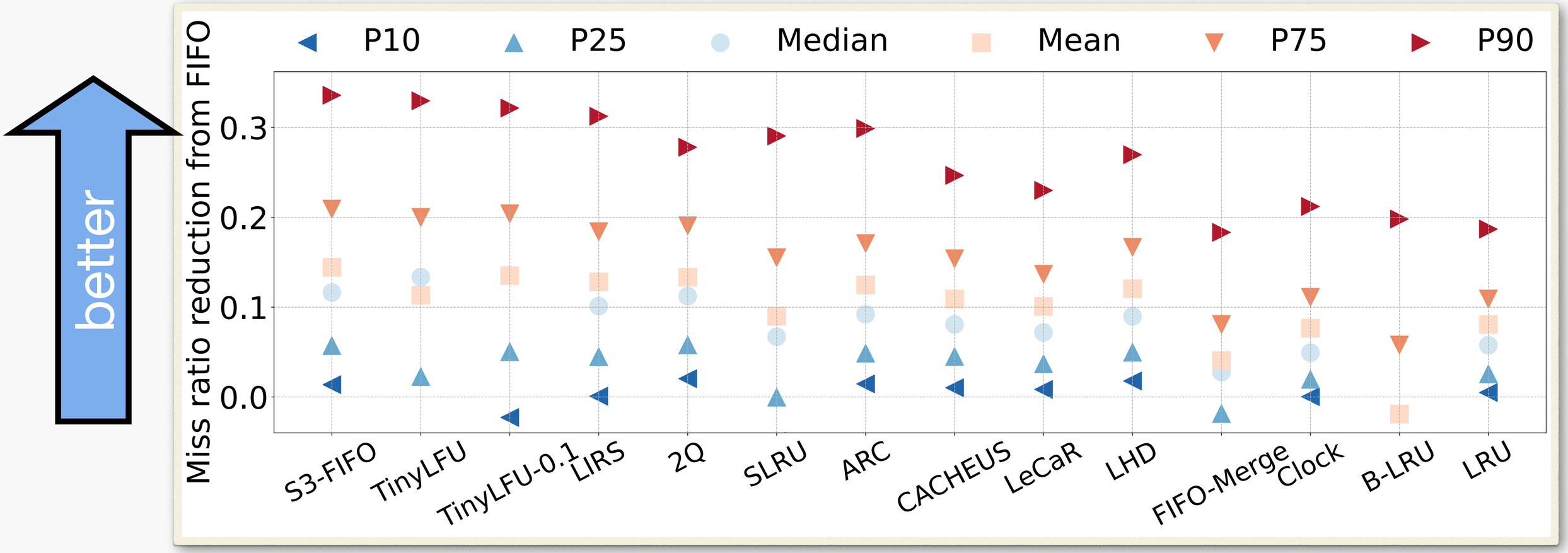
Evaluation setup

- Dataset
  - 14 datasets, 6594 traces from Twitter, Meta, Microsoft, Wikimedia, Tencent, Alibaba, major CDNs…
  - 848 billion requests , 60 billion objects
  - collected between 2007 and 2023
  - block, key-value, object caches
- Platform
  - libCacheSim, cachelib
  - in-house distributed computation system
- Metric
  - miss ratio reduction from FIFO
  - throughput

CloudLab

Efficiency

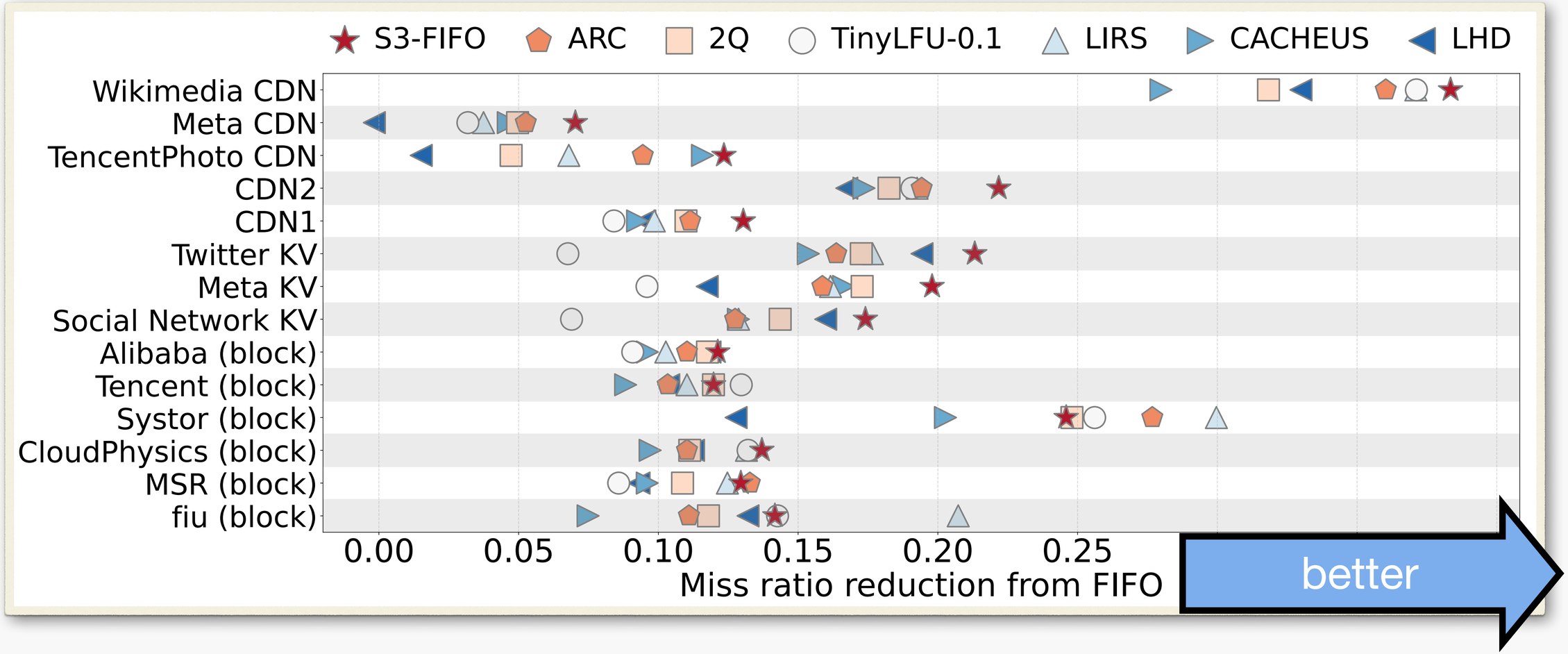
Miss ratio reduction distribution across all traces (large size)



Better than state-of-the-art algorithms, up to 72% lower miss ratio than LRU

Efficiency

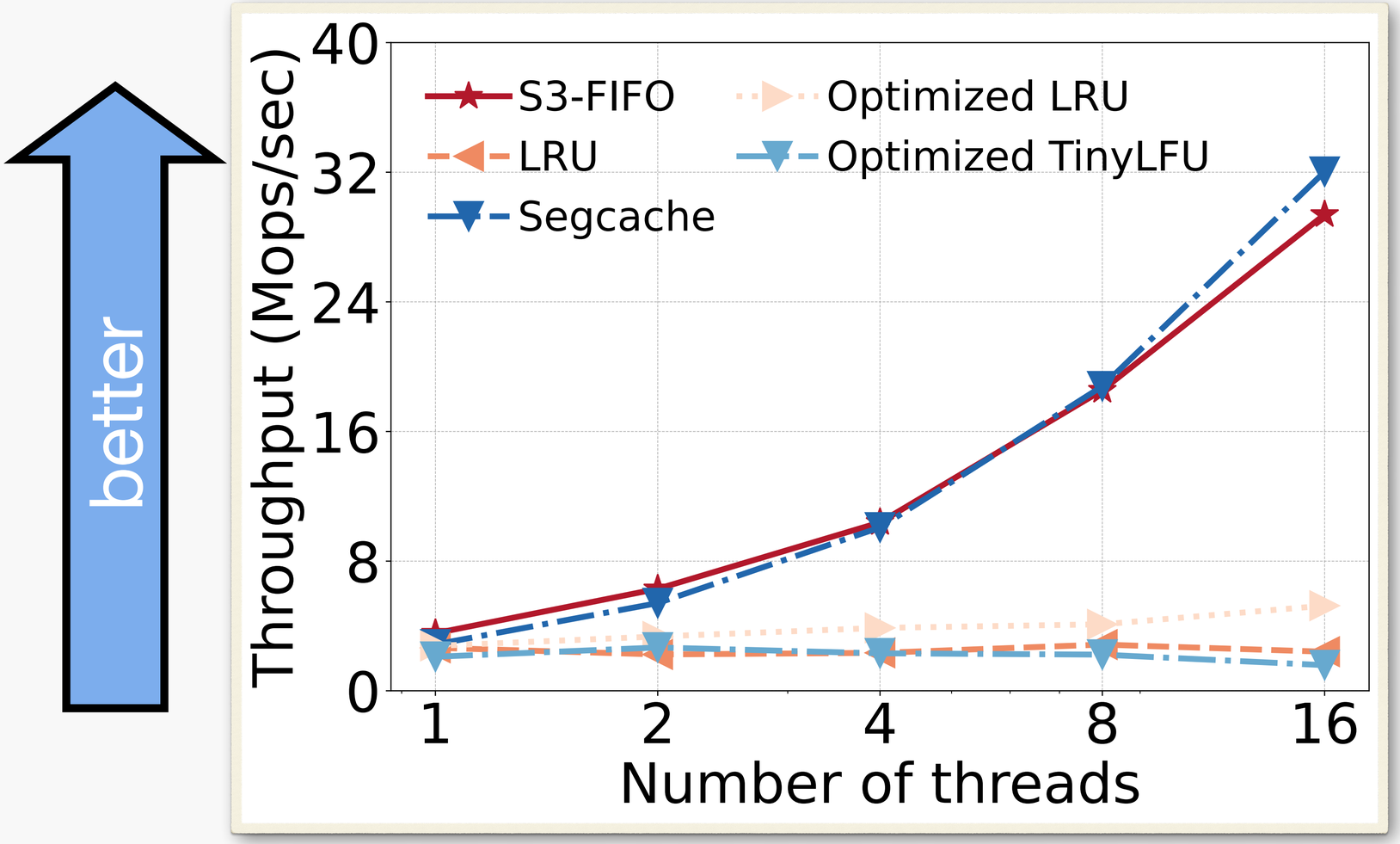
Mean miss ratio reduction on each dataset (large size)



- **efficient**: the largest mean miss ratio reduction or is close to the best
- **robust**: the best on 10 of the 14 datasets

Throughput and scalability

Zipf workloads

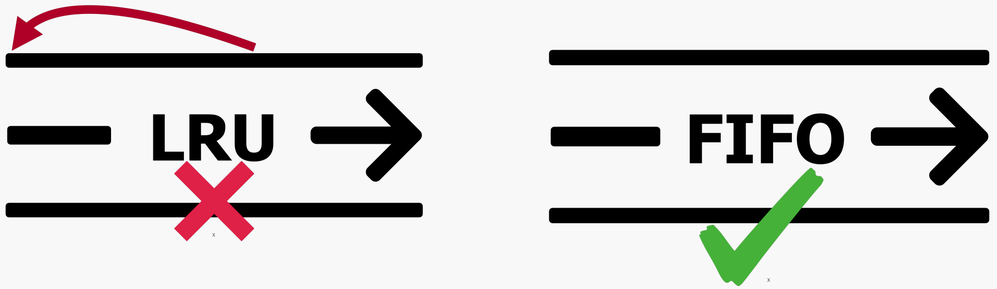


- the fastest on a single thread
- more scalable than optimized LRU, 6x higher throughput
- close to Segcache[NSDI'21]

More in the paper

- Why S3-FIFO is effective
- Implication for flash cache
- Byte miss ratio
- Impact of small FIFO size
- What if we use LRU

Takeaway



- **Cache workloads exhibit high one-hit-wonder ratio**
  - many objects in the cache are not re-accessed
  - critical to remove the one-hit wonders early
- **S3-FIFO**: **s**imple, **s**calable caching with **three** **s**tatic **FIFO** queues
  - reinsertion to keep popular objects
  - a small FIFO queue to *quickly* filter out one-hit wonders
  - interests from Google, VMWare vSAN, Cloudflare, TigerBeetle, Kuaishou, etc.

juncheny@cs.cmu.edu

[HotOS'23] FIFO queues can be better than LRU

https://s3fifo.com
https://blog.s3fifo.com