Carnegie Mellon University

FrozenHot Cache:

Rethinking Cache Management for Modern Hardware

Ziyue Qiu[*,o,^], Juncheng Yang[^], Juncheng Zhang[*], Cheng Li[*],
Xiaosong Ma[+], Qi Chen[o], Mao Yang[o], Yinlong Xu[*]

[*] University of Science and Technology of China

[o] Microsoft Research

[^] Carnegie Mellon University

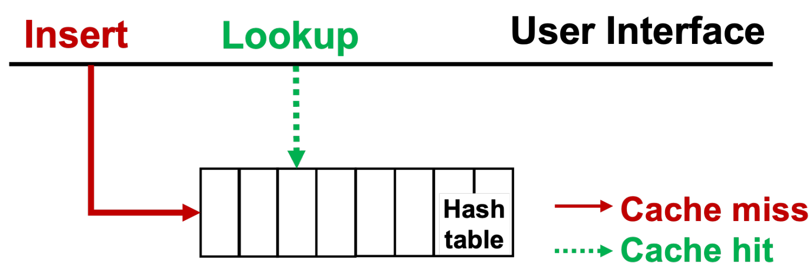
[+] Qatar Computing Research Institute

List-based Software Cache Dominant

- List-based caches are **dominant**, e.g., LRU, FIFO, LFU variants

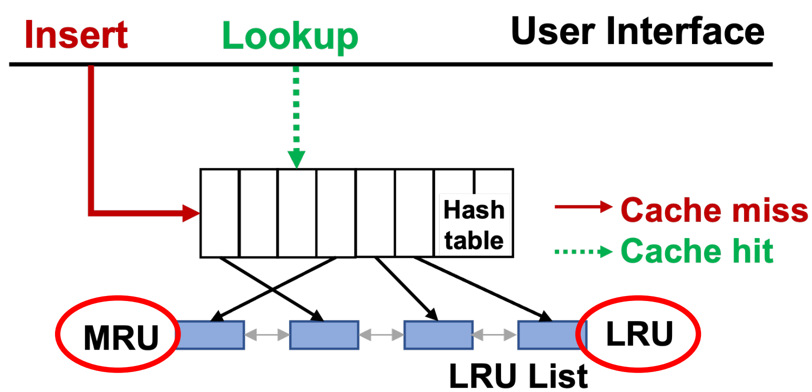
List-based Software Cache Dominant

- List-based caches are **dominant**, e.g., LRU, FIFO, LFU variants
- Main operations incurred by application accesses:
  - lookup
  - insert (write after cache miss)



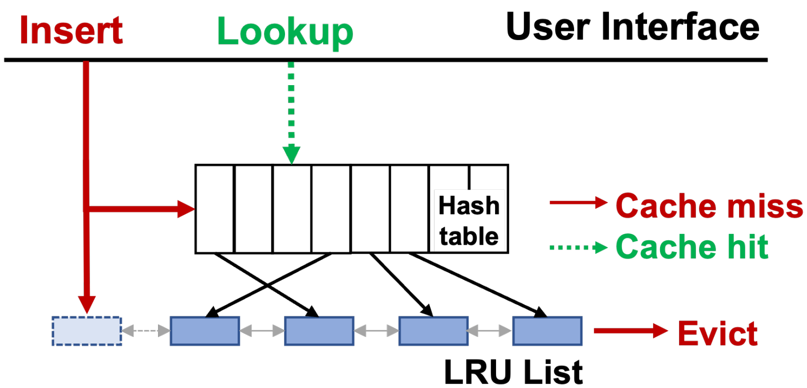
List-based Software Cache Dominant

- List-based caches are **dominant**, e.g., LRU, FIFO, LFU variants
- Main operations incurred by application accesses:
  - lookup
  - insert (write after cache miss)
- Common implementation: hash table + doubly-linked list



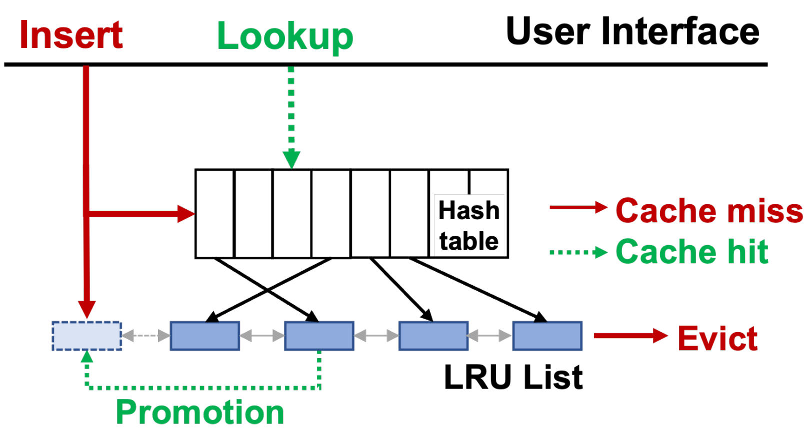
Locks, Locks, Locks Everywhere

- Insert: insert to head and evict the tail **(lock)**



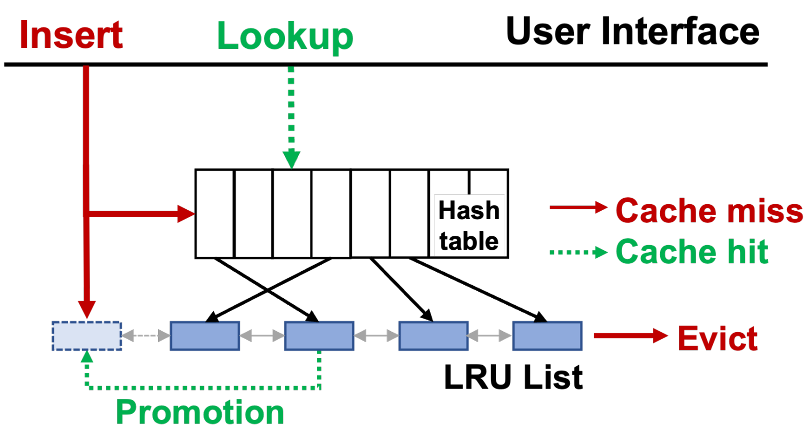
Locks, Locks, Locks Everywhere

- Insert: insert to head and evict the tail **(lock)**
- Lookup: delink and push to front **(also lock!)**



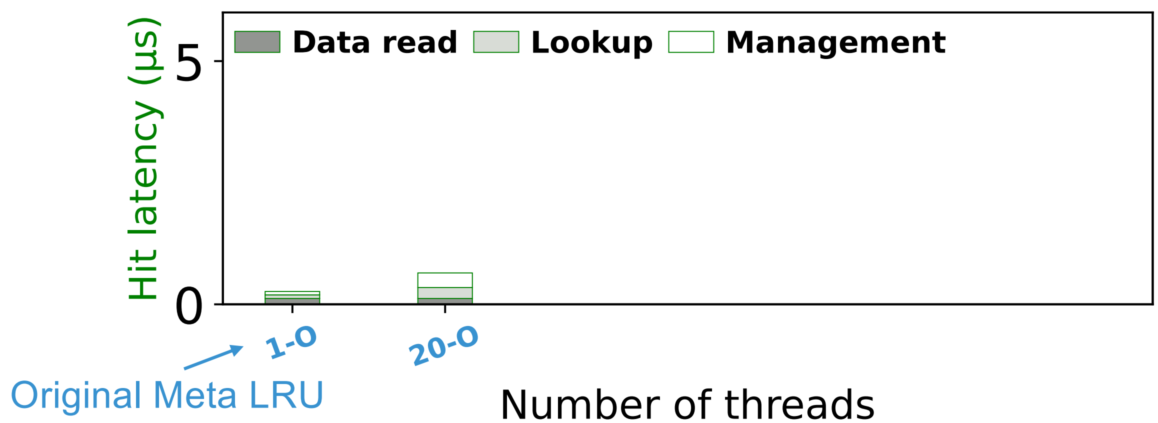
Locks, Locks, Locks Everywhere

- Insert: insert to head and evict the tail **(lock)**
- Lookup: delink and push to front **(also lock!)**
- Cache internal operations: **update-intensive & contention-heavy**
  - Even under **cache-friendly, read-only** workloads



Huge Management Cost

- Before fast SSDs:
  - Low Concurrency: low hit latency (< 1 *μs*)

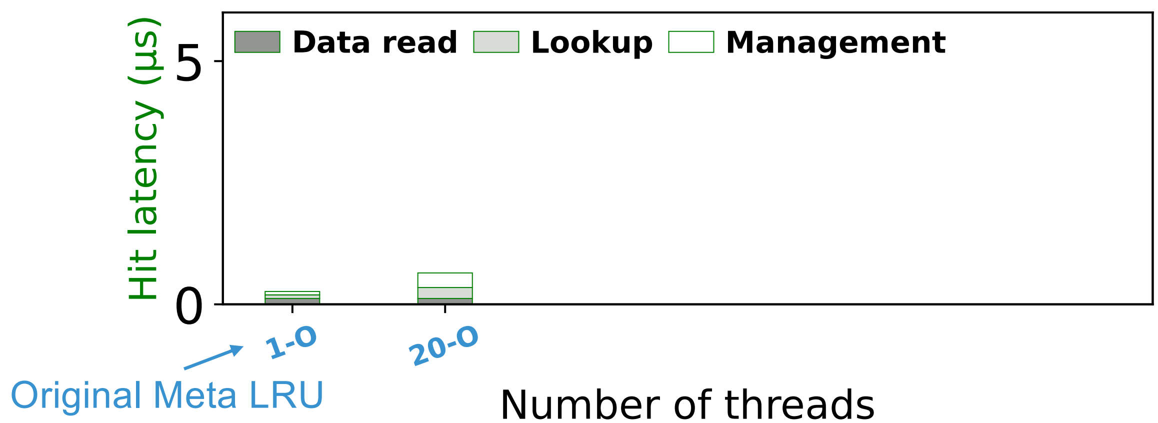


* Run on Meta HHVM LRU Cache

Carnegie Mellon University

Huge Management Cost

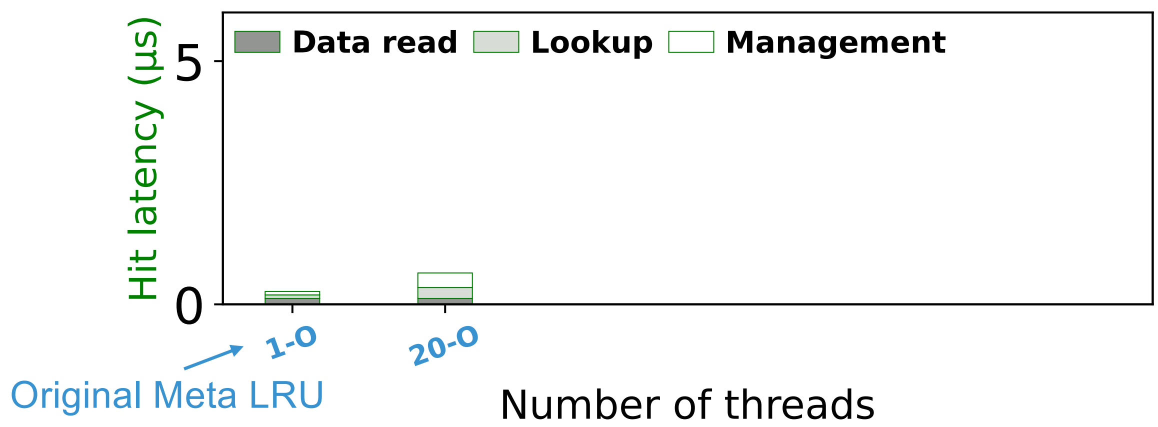
- Before fast SSDs:
  - Low Concurrency: low hit latency (< 1 *μs*)
  - Slow Storage Backend: long latency (> 5 *ms*) and low bandwidth



* Run on Meta HHVM LRU Cache

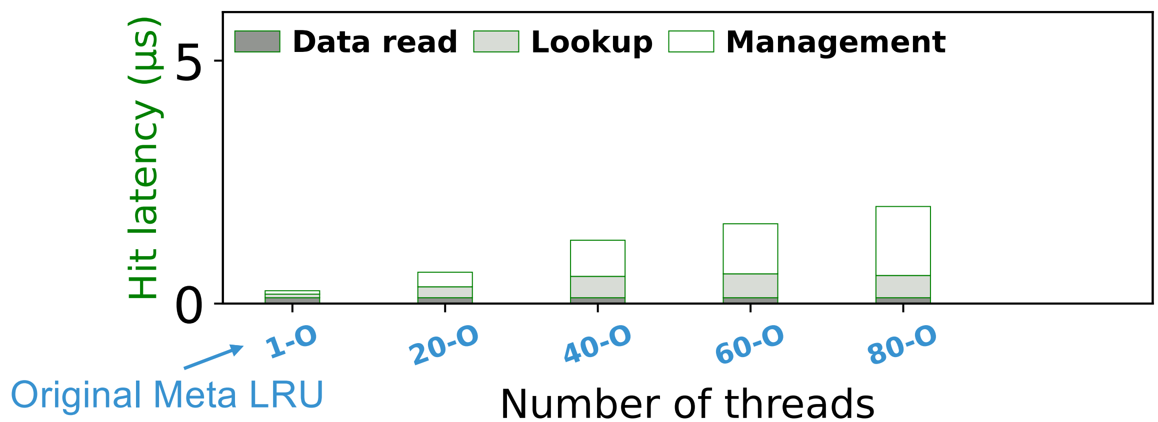
Huge Management Cost

- Now:
  - **Increasing cores** (concurrency): AMD up to 192 threads
  - **Shrinking latency gap**: Intel Optane SSD 5 *μs*,
    Samsung Z-SSD 16 *μs* << HDD 5~10 *ms* (4 KB read)



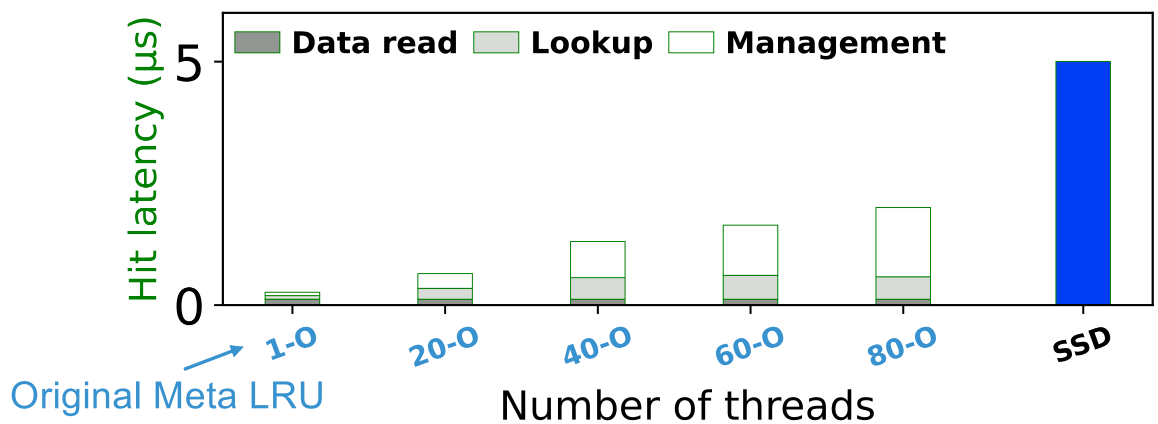
Huge Management Cost

- **Increasing cores** (concurrency): AMD up to 192 threads
- **Shrinking latency gap:** Intel Optane SSD 5 *μs*,
  Samsung Z-SSD 16 *μs* << HDD 5~10 *ms* (4 KB read)
- Latency skyrockets as concurrency increases



Huge Management Cost

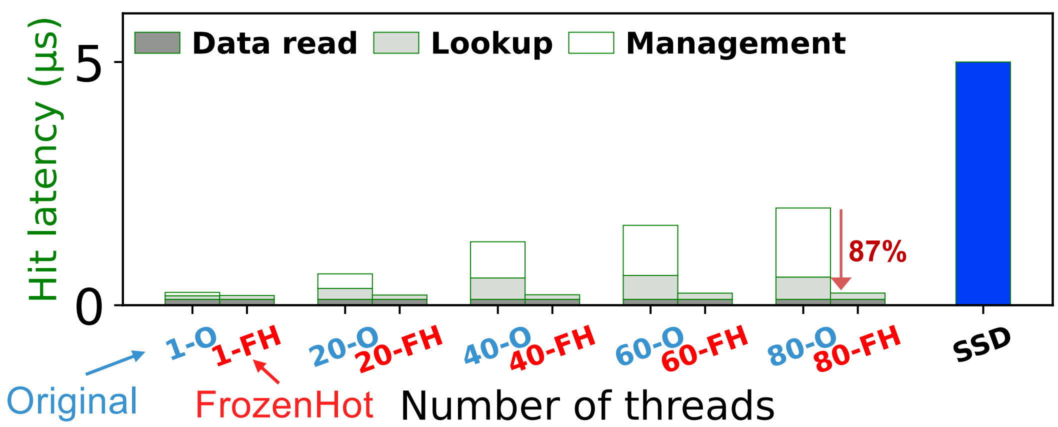
- **Increasing cores** (concurrency): AMD up to 192 threads
- **Shrinking latency gap:** Intel Optane SSD 5 *μs*,
  Samsung Z-SSD 16 *μs* << HDD 5~10 *ms* (4 KB read)
- Latency skyrockets as concurrency increases
- Close to half of Optane SSD read latency



Huge Management Cost

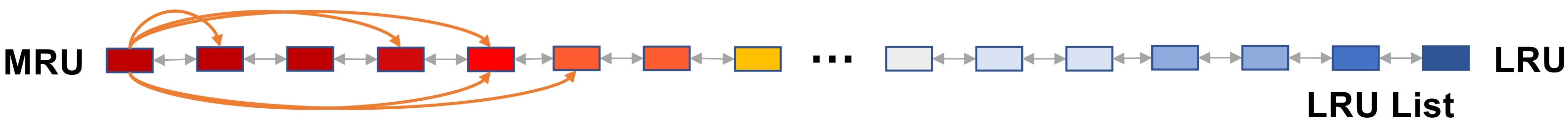
FrozenHot:

new in-memory cache design for scalability



Workload Examination for Cache Redesign

- **Cache-friendly**: random, highly skewed accesses *

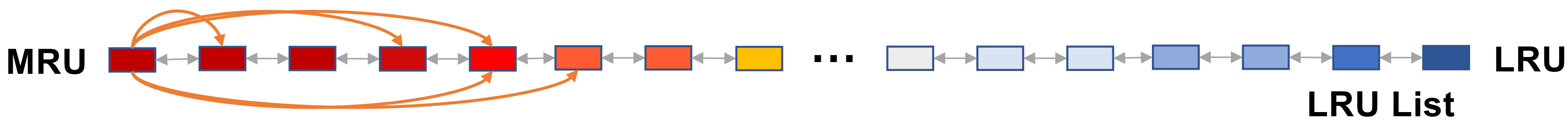


Mostly hits, each incurring promotions => shuffling hottest objects

* LHD: improving cache hit rate by maximizing hit density, NSDI'17

Workload Examination for Cache Redesign

- **Cache-friendly**: random, highly skewed accesses *



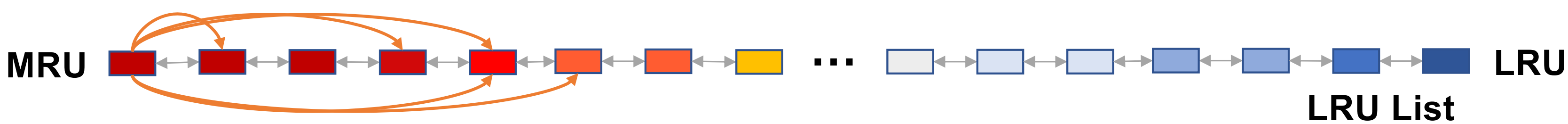
Mostly hits, each incurring promotions => shuffling hottest objects

No need for frequent LRU-like list manipulation

* LHD: improving cache hit rate by maximizing hit density, NSDI'17

Workload Examination for Cache Redesign

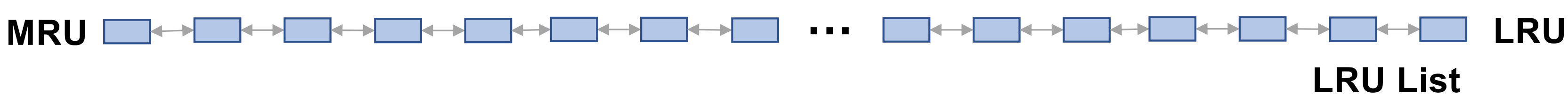
- **Cache-friendly**: random, highly skewed accesses *



Mostly hits, each incurring promotions => shuffling hottest objects

No need for frequent LRU-like list manipulation

- **Cache-unfriendly**: sequential scans *

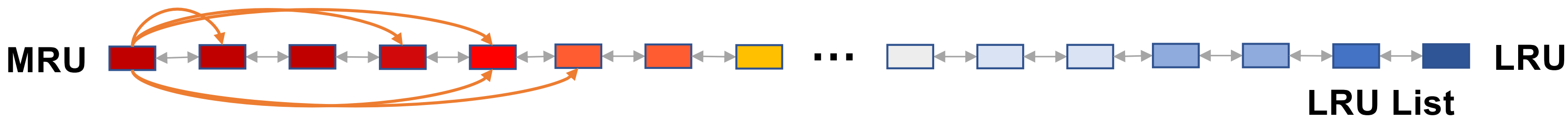


Mostly misses => useless list maintenance

* LHD: improving cache hit rate by maximizing hit density, NSDI’17

Workload Examination for Cache Redesign

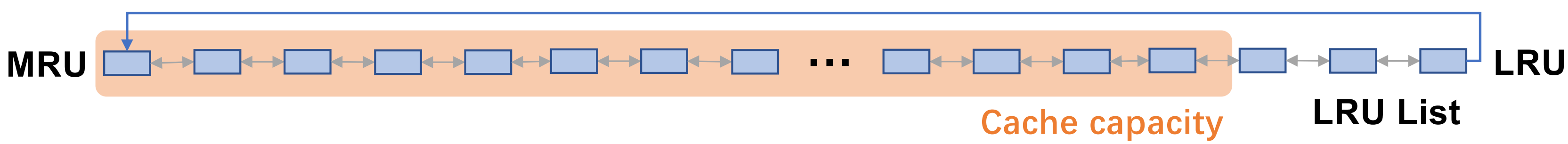
- **Cache-friendly**: random, highly skewed accesses *



Mostly hits, each incurring promotions => shuffling hottest objects

No need for frequent LRU-like list manipulation

- **Cache-unfriendly**: sequential scans *



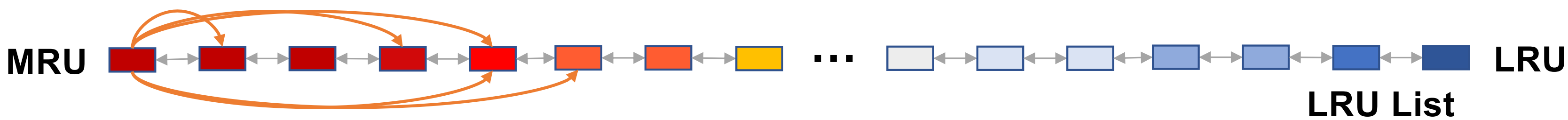
Mostly misses => useless list maintenance

Cyclic scans + large working set => thrashing

* LHD: improving cache hit rate by maximizing hit density, NSDI'17

Workload Examination for Cache Redesign

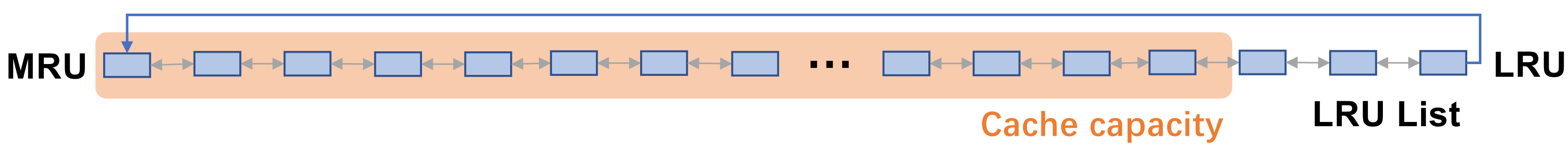
- **Cache-friendly**: random, highly skewed accesses *



Mostly hits, each incurring promotions => shuffling hottest objects

No need for frequent LRU-like list manipulation

- **Cache-unfriendly**: sequential scans *



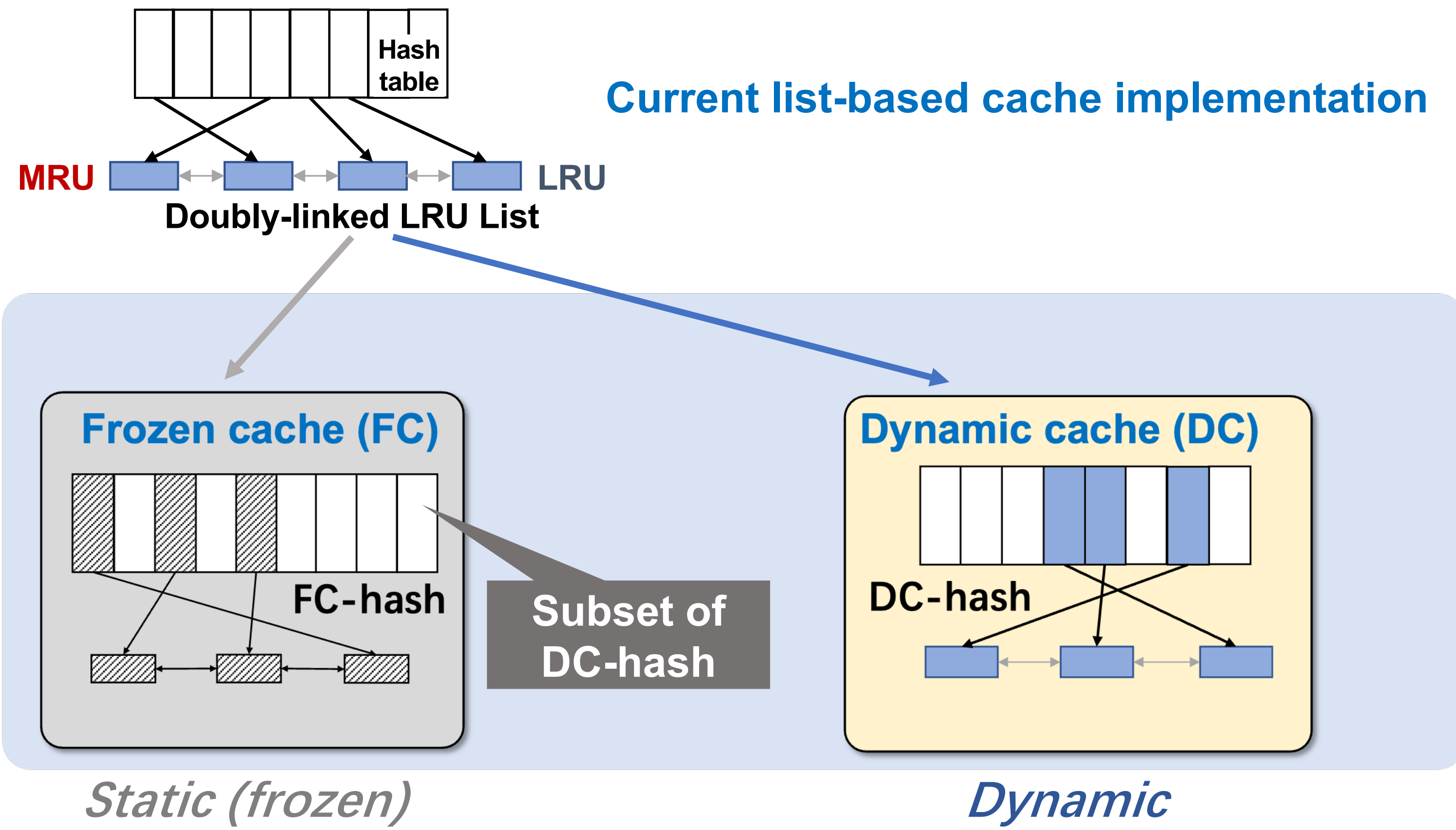
Mostly misses => useless list maintenance

Cyclic scans + large working set => thrashing

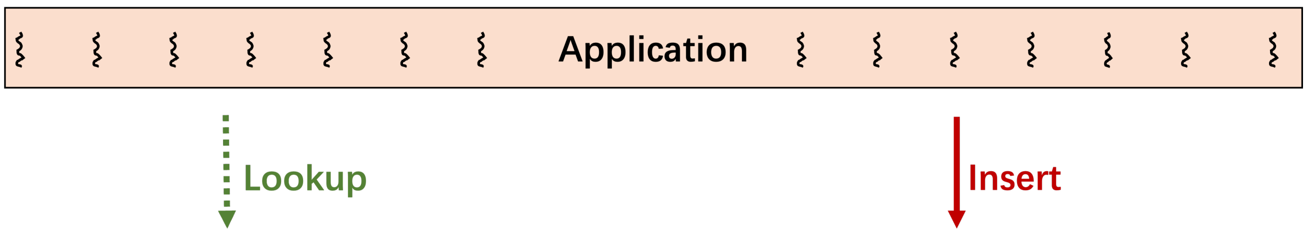
No help or **even harm** (e.g., cache thrashing)

* LHD: improving cache hit rate by maximizing hit density, NSDI'17

FrozenHot Design – Data Structures

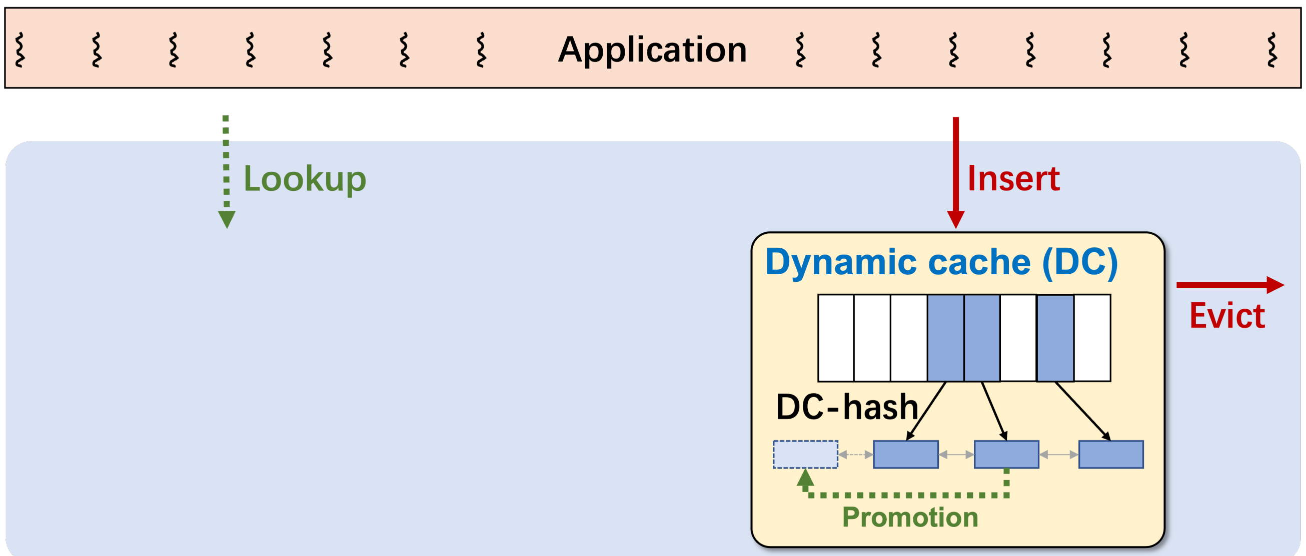


FrozenHot Design – Operations



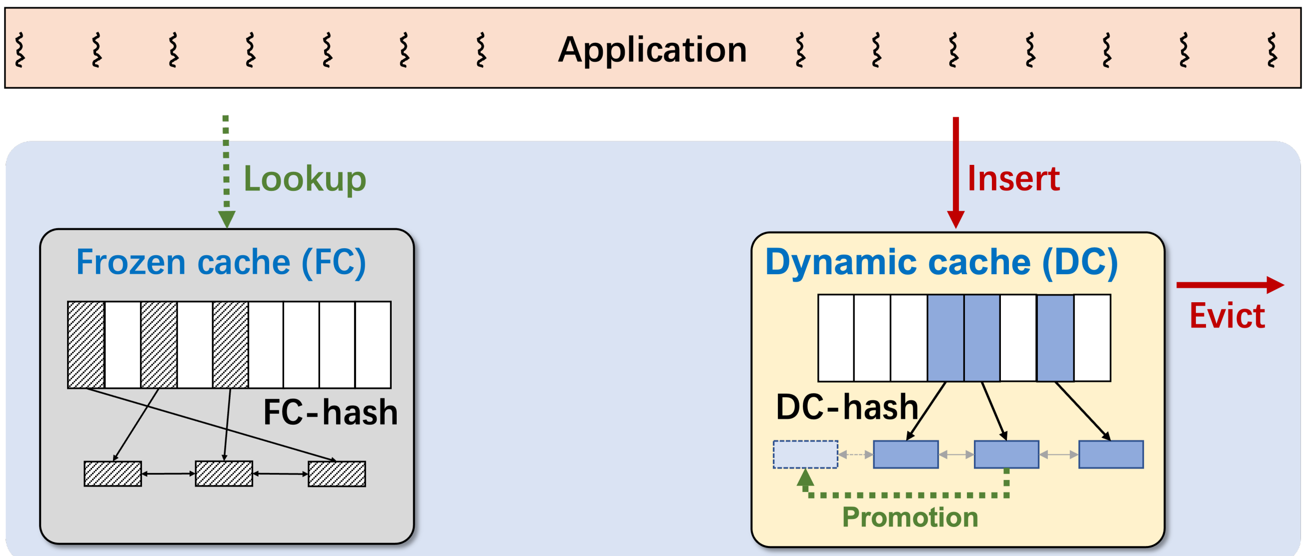
FrozenHot Design – Operations

- Insertions and evictions occur only in Dynamic Cache (DC)



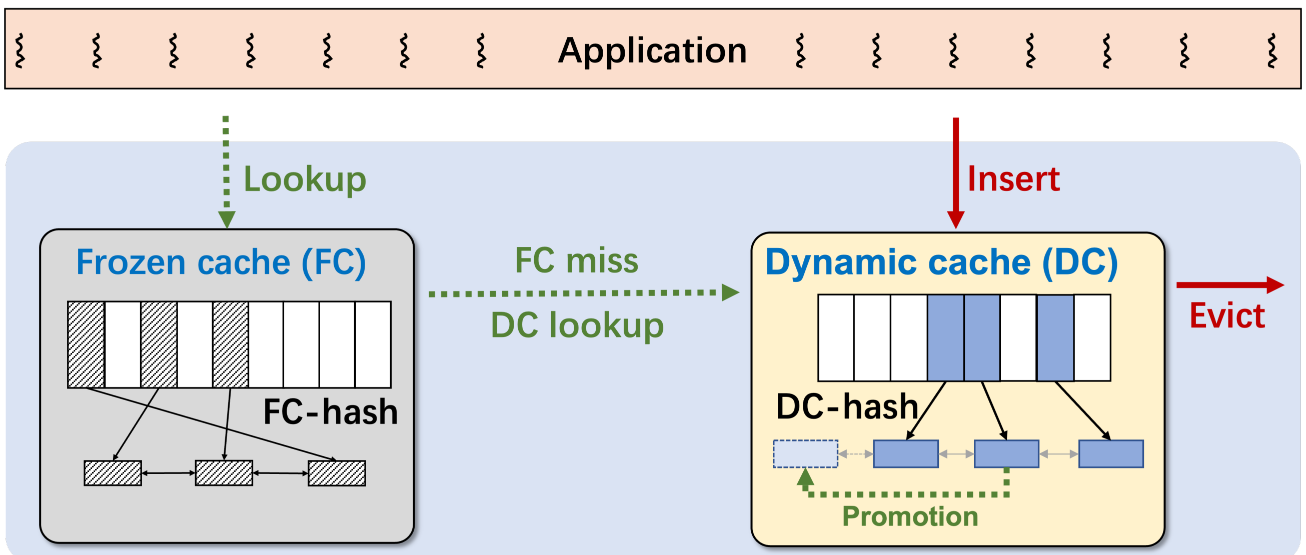
FrozenHot Design – Operations

- A lookup first goes to Frozen Cache (FC)



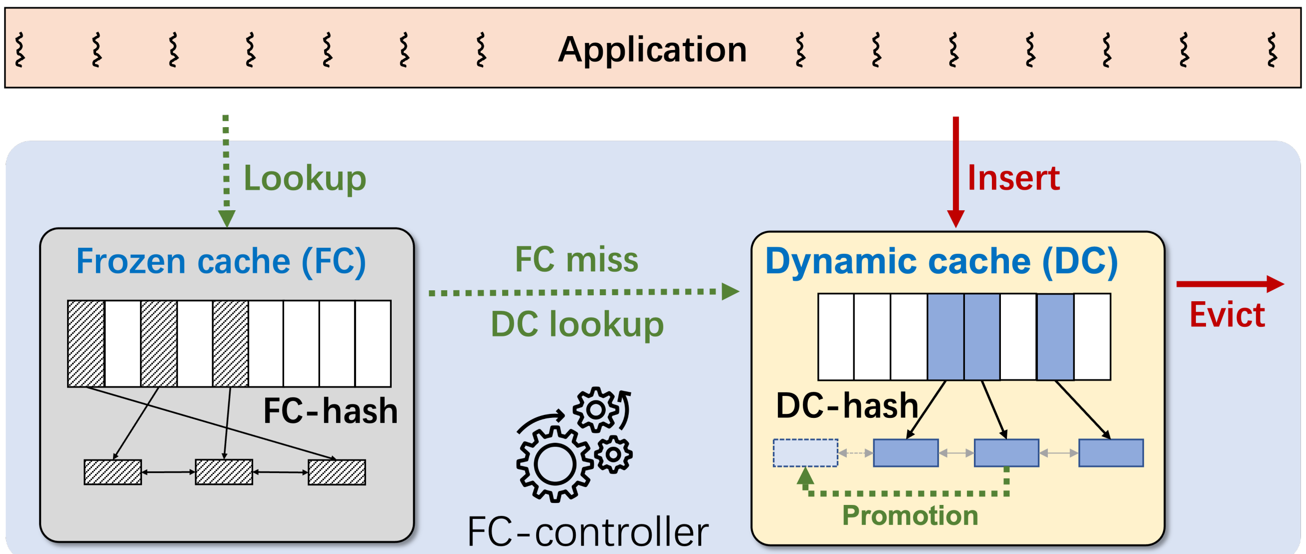
FrozenHot Design – Operations

- If it is a Frozen cache miss, then look up in DC



FrozenHot Design – Performance/Scalability Benefits

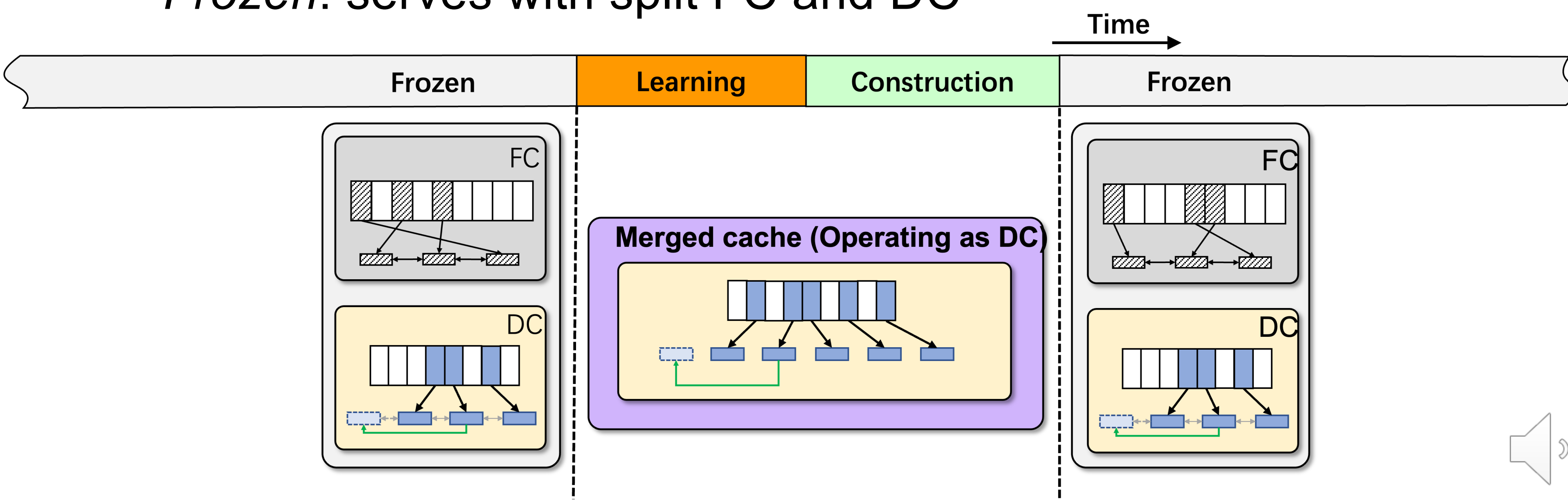
- No cache management on FC accesses (less work)
- No contention either (lock-free)
- Read-only FC-Hash can use faster hash table implementations



Carnegie Mellon University

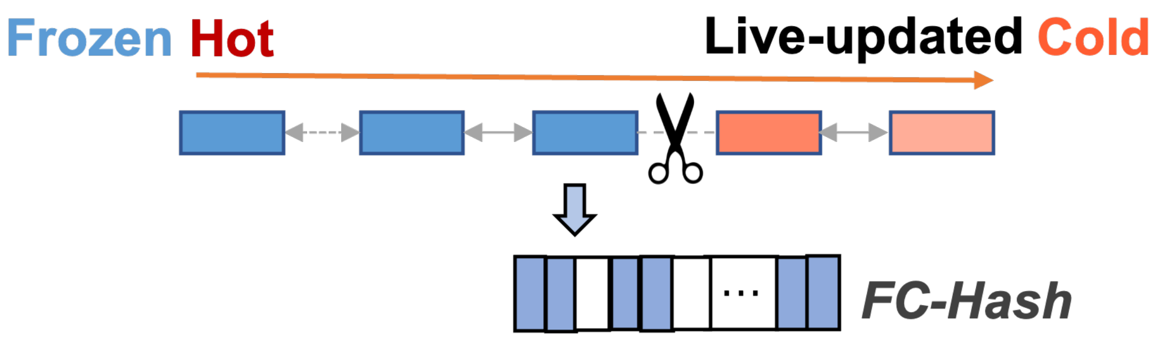
FrozenHot Design – Life Cycles

- FrozenHot alternates through THREE phases:
  - *Learning*: merges FC+DC and observes operations
  - *Construction*: rebuilds FC with learned parameters
  - *Frozen*: serves with split FC and DC



FrozenHot Design – Periodic FC Rebuild

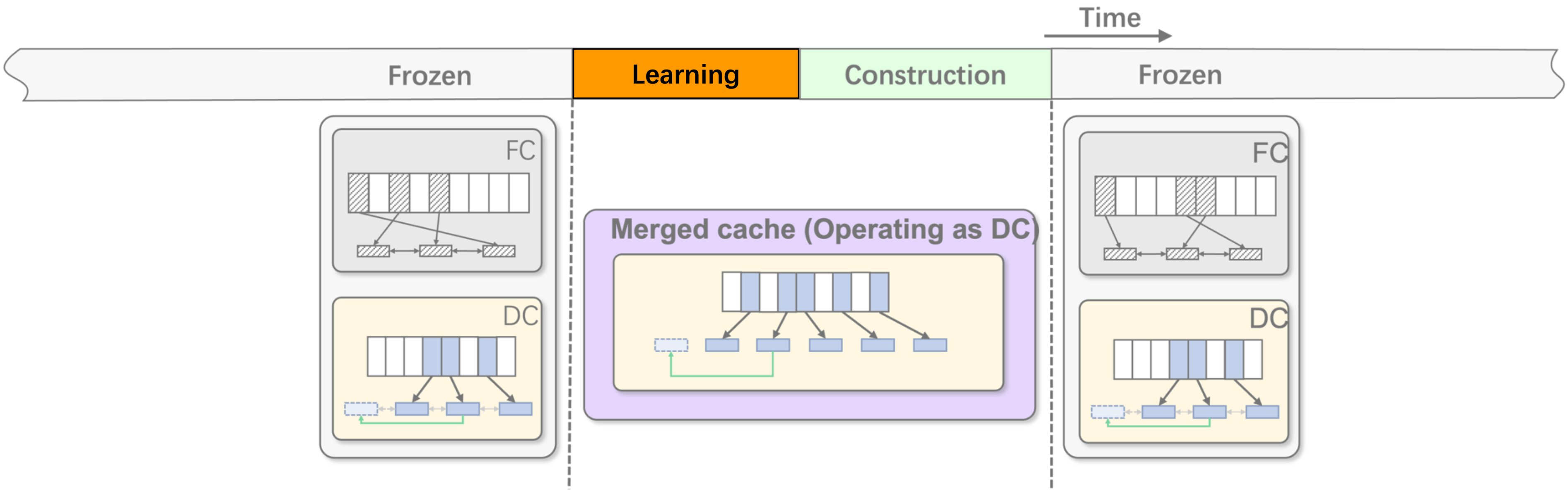
- FC Construction
  - Splitting top-*k* objects, **O(1)** complexity
  - Constructing FC-Hash, O(n) *in background*



- FC Destruction: merging FC+DC lists and removing FC-Hash, **O(1)**
- Support all list-based implementations, e.g., LRU, FIFO, LFU

Carnegie Mellon University

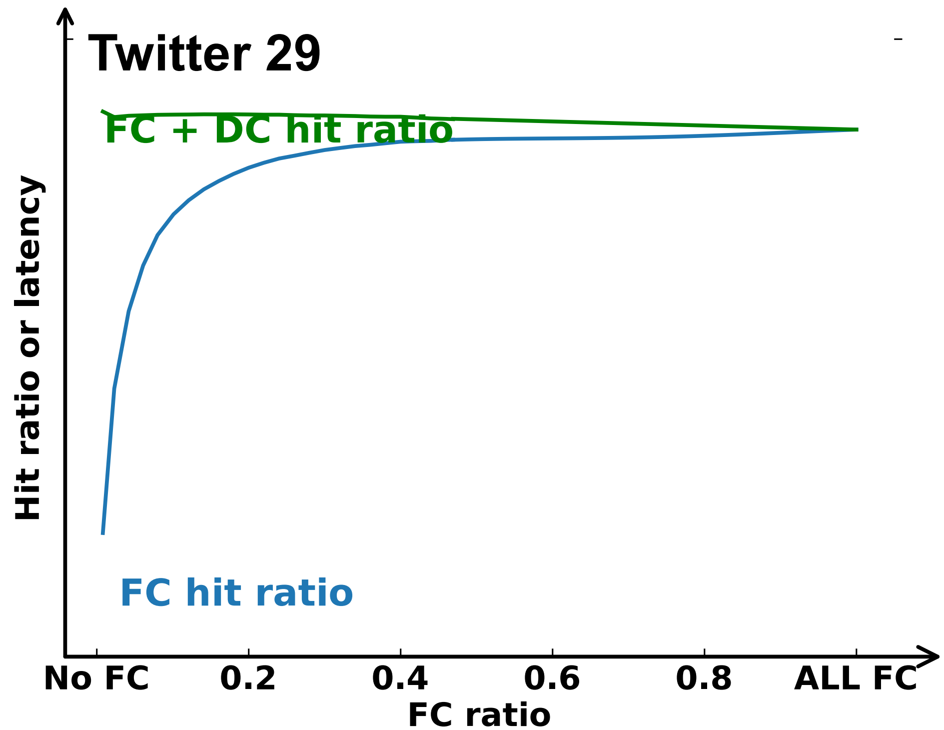
FrozenHot Design – Learning Key Parameters



- Spatial: **how much** and **which** part of the cache should be frozen
- Temporal: **how long** each frozen cache should last

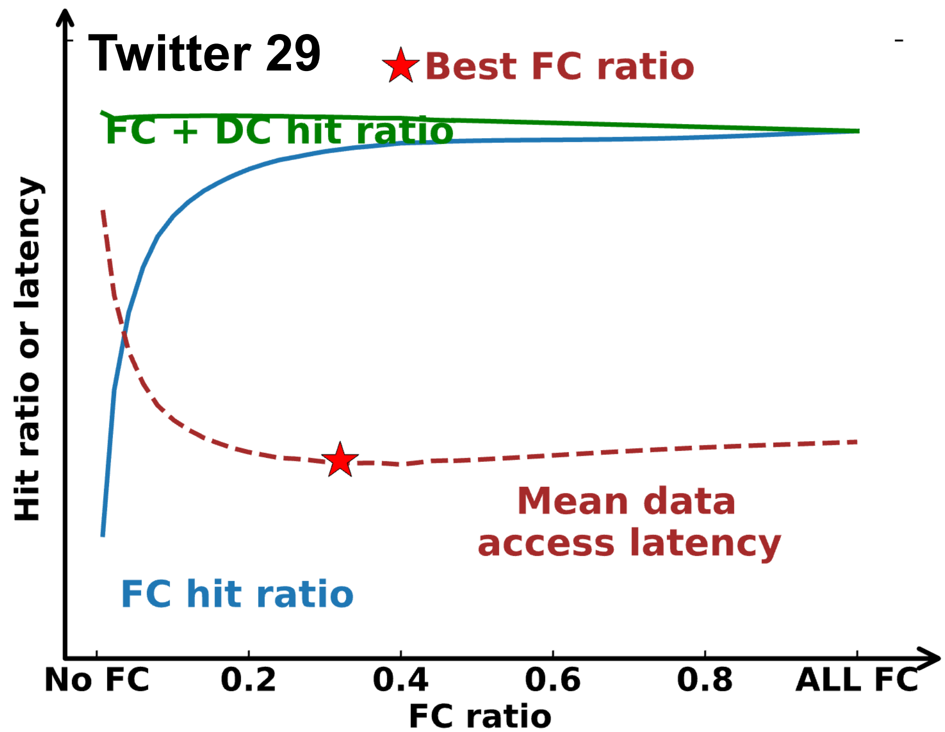
FrozenHot Design – FC Size Auto-configuration

- Value of ***k*** in top-***k*** (list already maintains order)
- More frozen, more hits in FC, gradually more misses in total



FrozenHot Design – FC Size Auto-configuration

- Value of ***k*** in top-***k*** (list already maintains order)
- More frozen, more hits in FC, gradually more misses in total



Carnegie Mellon University

FrozenHot Design – Frozen Phase Length Auto-configuration

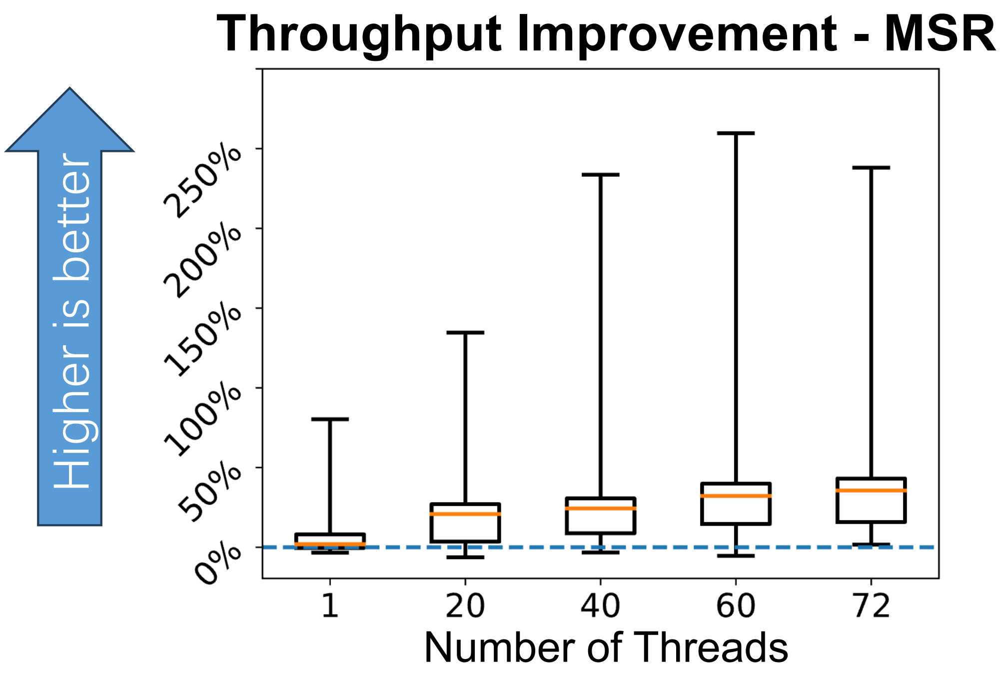
- Controller monitors dynamic performance
- Ends Frozen phase accordingly

Evaluation – Setup

- Compared Systems
  - LRU-FH v.s Relaxed-LRU from Meta HHVM (production)
  - FIFO-FH v.s. FIFO
  - LFU-FH v.s. LFU
- Workloads: 7 Twitter traces and 12 MSR Cambridge traces

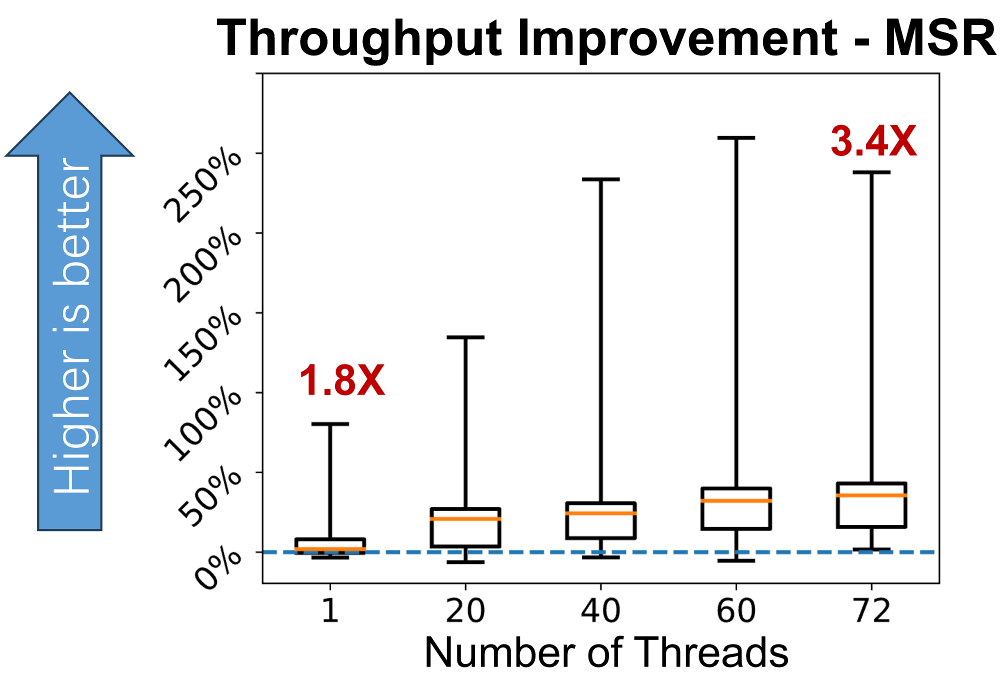
Carnegie Mellon University

Evaluation – Throughput Improvement



Evaluation – Throughput Improvement

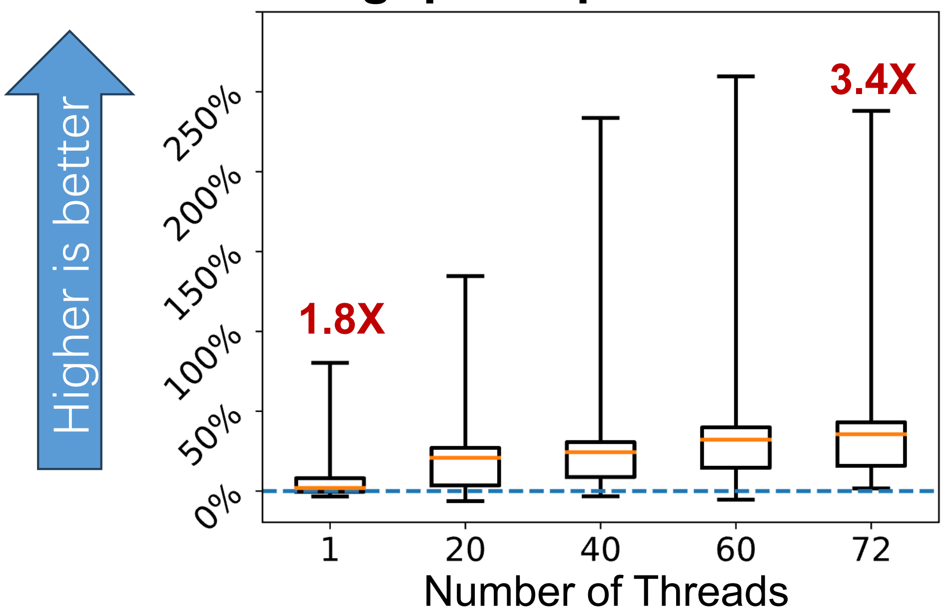
- Increasing gains with growing concurrency level



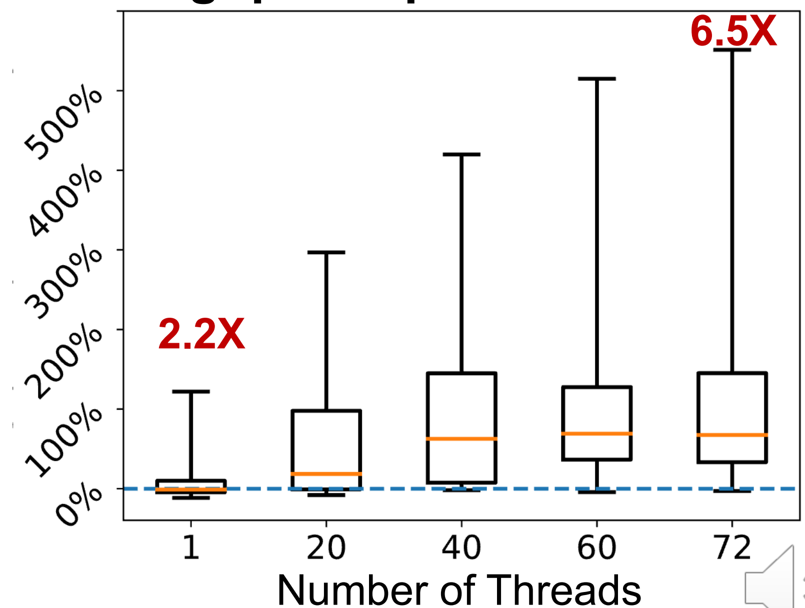
Evaluation – Throughput Improvement

- Increasing gains with growing concurrency level
- Also with workloads having higher locality

Throughput Improvement - MSR



Throughput Improvement - Twitter

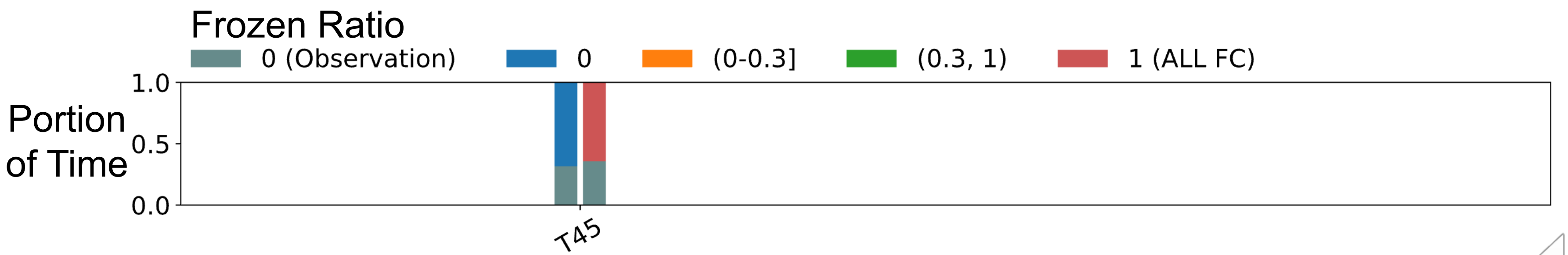


* For the breakdown of the improvement and timelines of different phases, see the paper

Carnegie Mellon University

Evaluation – Frozen Ratio Selection

- Stacked bars show portion of time at each Frozen Ratio range
- *Observation period*: observe accesses to decide internal parameters



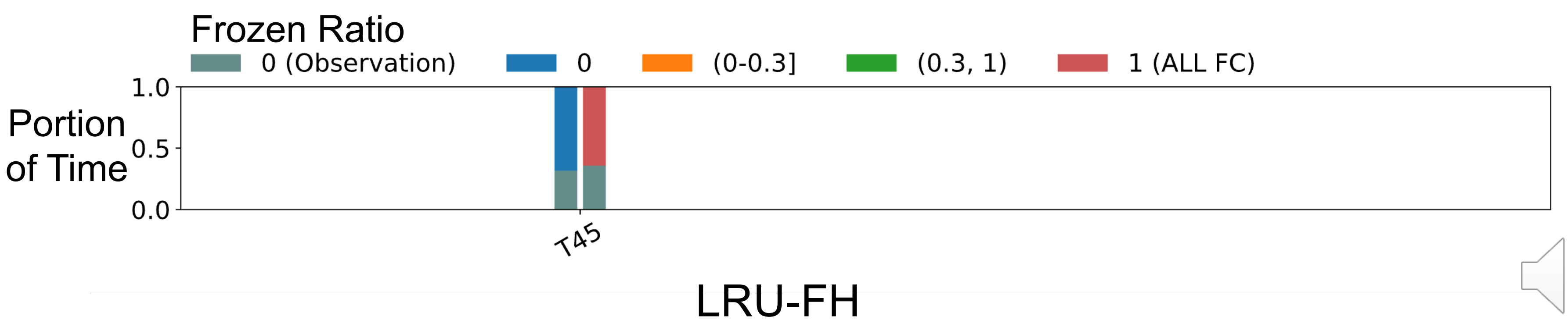
LRU-FH

* Two concurrency levels: 20-thread (left bar) and 72-thread (right bar)

Carnegie Mellon University

Evaluation – Frozen Ratio Selection

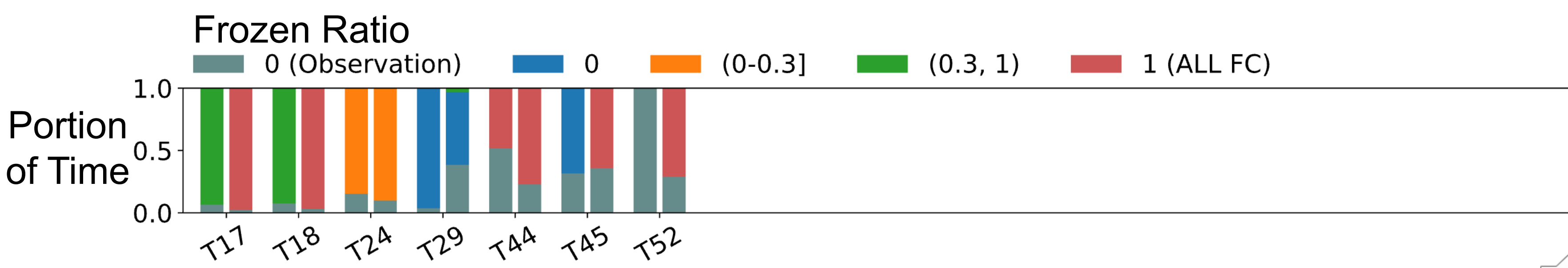
- Stacked bars show portion of time at each Frozen Ratio range
- *Observation period*: observe accesses to decide internal parameters
- Observation 1: higher concurrency, more frozen



* Two concurrency levels: 20-thread (left bar) and 72-thread (right bar)

Evaluation – Frozen Ratio Selection

- Stacked bars show portion of time at each Frozen Ratio range
- *Observation period*: observe accesses to decide internal parameters
- Observation 1: higher concurrency, more frozen
- Observation 2: Frozen Ratio highly depends on workload patterns

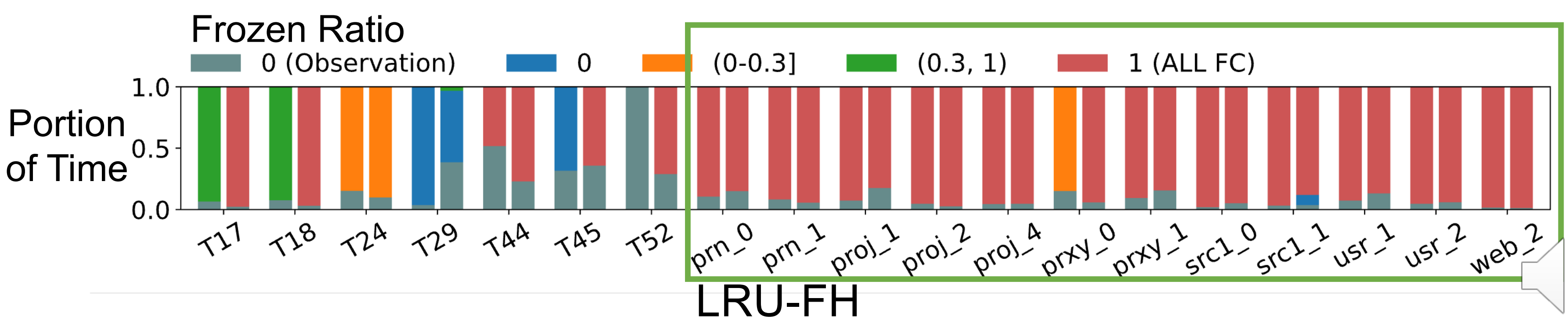


LRU-FH

* Two concurrency levels: 20-thread (left bar) and 72-thread (right bar)

Evaluation – Frozen Ratio Selection

- Stacked bars show portion of time at each Frozen Ratio range
- *Observation period*: observe accesses to decide internal parameters
- Observation 1: higher concurrency, more frozen
- Observation 2: Frozen Ratio highly depends on workload patterns
- Observation 3: 100% Frozen when workloads are cache-unfriendly



* Two concurrency levels: 20-thread (left bar) and 72-thread (right bar)

Conclusion

Key Observation:

- In-memory cache needs redesign
- Continuous, full cache maintenance is **wasteful**

FrozenHot:

periodically-rebuilt **frozen** cache + live-updated **dynamic** cache

Open-sourced: https://github.com/ziyueqiu/FrozenHot.git

ziyueqiu@cs.cmu.edu
https://ziyueqiu.github.io/