# **Elastic Queue:** A Universal SSD Lifetime Extension Plug-in for Cache Replacement Algorithms

Yushi Liang, Yunpeng Chai, Ning Bao, Huanyu Chen, Yaohong Liu

Key Laboratory of Data Engineering and Knowledge Engineering, Ministry of Education,

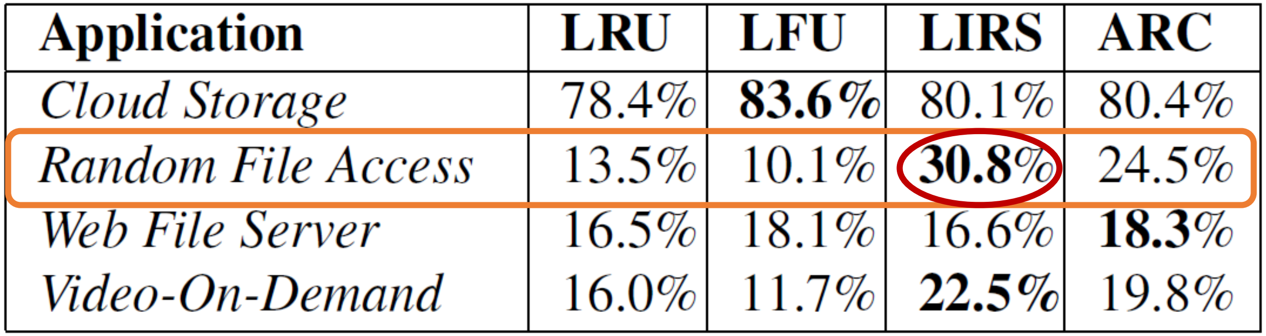
School of Information,

Renmin University of China

# Traditional Cache Algorithm

- Plenty of researches
  - Different way of qualifying locality
- Adaptability to applications
  - Free to choose the most suitable one for certain senario

LRU
LIRS
LFU
ARC
2Q
…



| **Application** | **LRU** | **LFU** | **LIRS** | **ARC** |
|---|---|---|---|---|
| *Cloud Storage* | 78.4% | **83.6%** | 80.1% | 80.4% |
| *Random File Access* | 13.5% | 10.1% | **30.8%** | 24.5% |
| *Web File Server* | 16.5% | 18.1% | 16.6% | **18.3%** |
| *Video-On-Demand* | 16.0% | 11.7% | **22.5%** | 19.8% |

# SSD-based cache

- Solid State Drives
  - Lower price (vs. DRAM)
  - Higher IOPS, excellent random I/O bandwidth (vs. HDD)

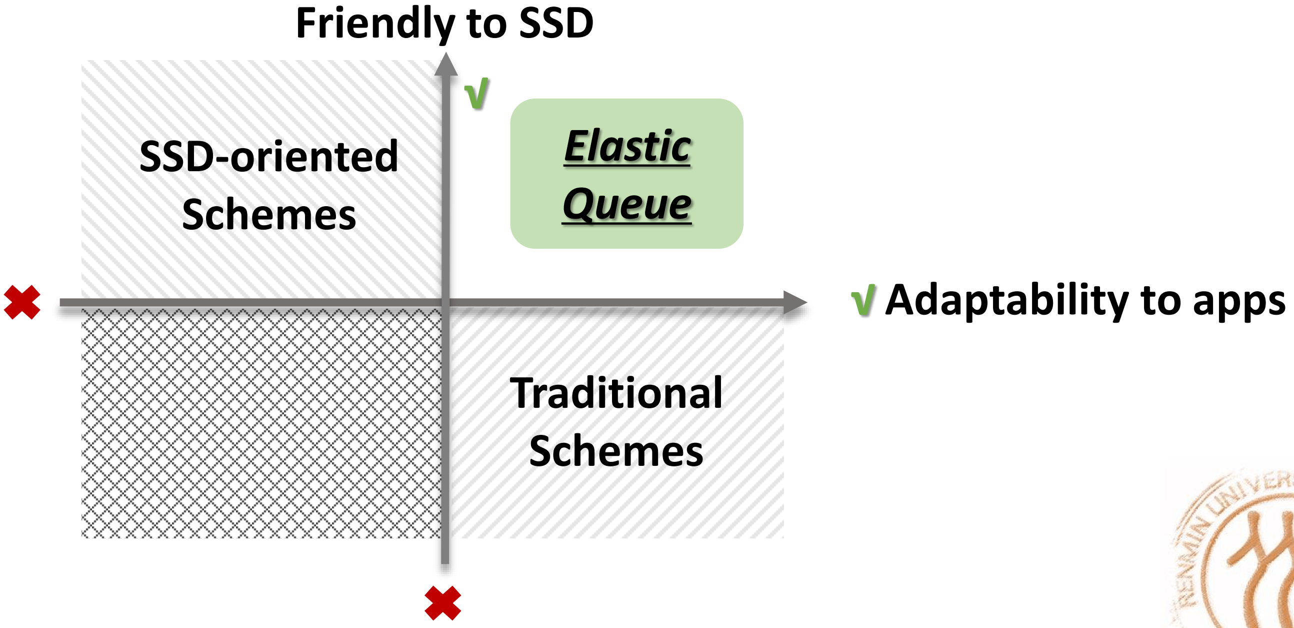
- Challenges
  - Limited times of re-writing for each unit
  - Unbalanced read / write performance

# SSD-oriented Cache Algorithm

- Friendly to SSD lifetime
  - LARC, L2ARC, Sievestore, WEC, ETD-Cache….
- Fixed strategy
  - Few choices
  - Diverse application feature

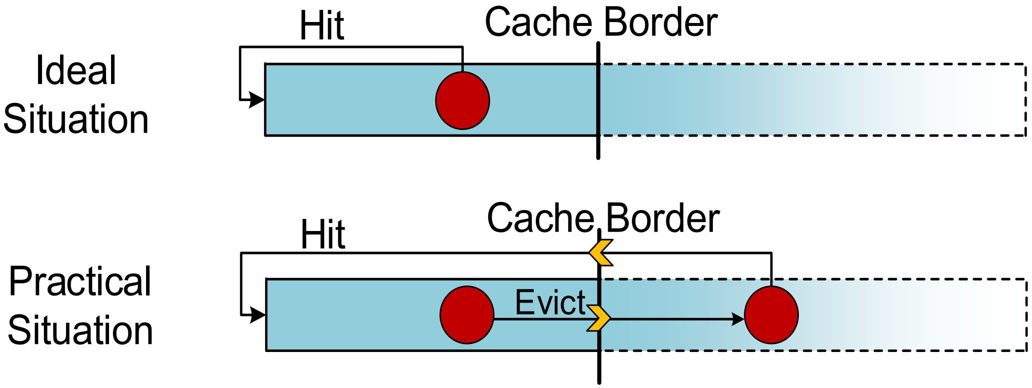
# Our Solution

- Elastic Queue
  - Cover the “blank zone”
  - Cooperate with any other cache algorithm
  - Provide protection to reduce SSD writes



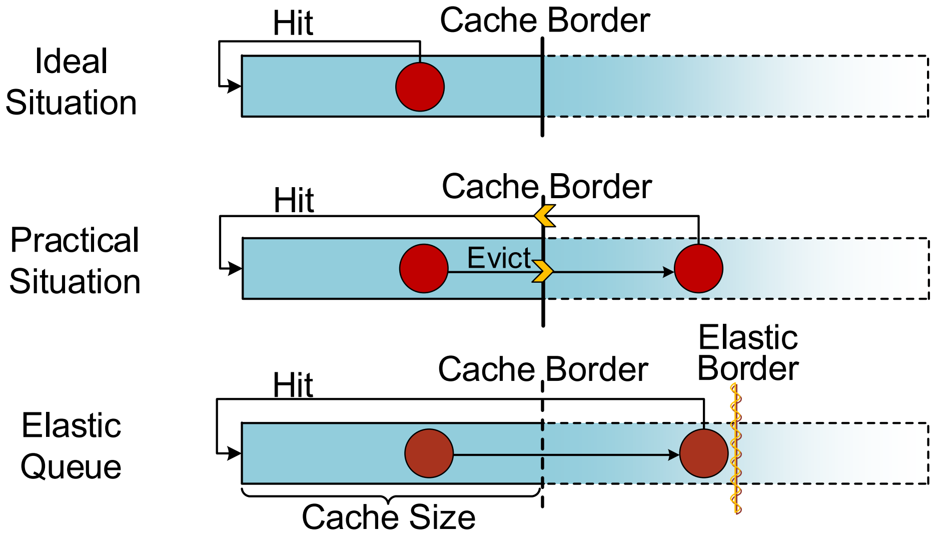
# Unified Priority Queue Model

- Unified queue model of cache algorithms
  - Blocks prioritized by qualified locality
- Common problem
  - Unstable access intervals ([Y. Chai+TOC 2015])
  - Too much unnecessary traversal on the cache border
  - Lead to SSD worn-out rapidly



# Elastic Queue Principle

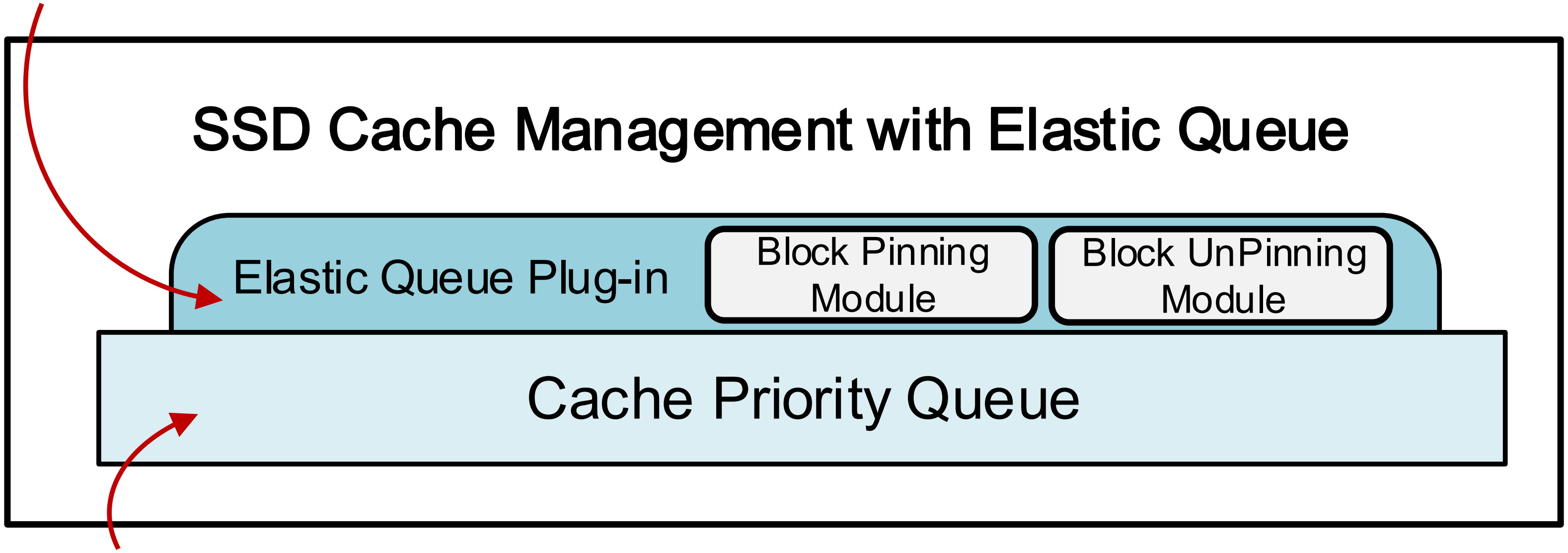
- Prevent hot blocks from early eviction
  - Pin blocks in SSD
  - Assigns Elastic Border (EB)
  - Enhance SSD endurance



# Elastic Queue Architecture

- 1 Queue + 2 Modules

Provide protection



SSD Cache Management with Elastic Queue

Elastic Queue Plug-in

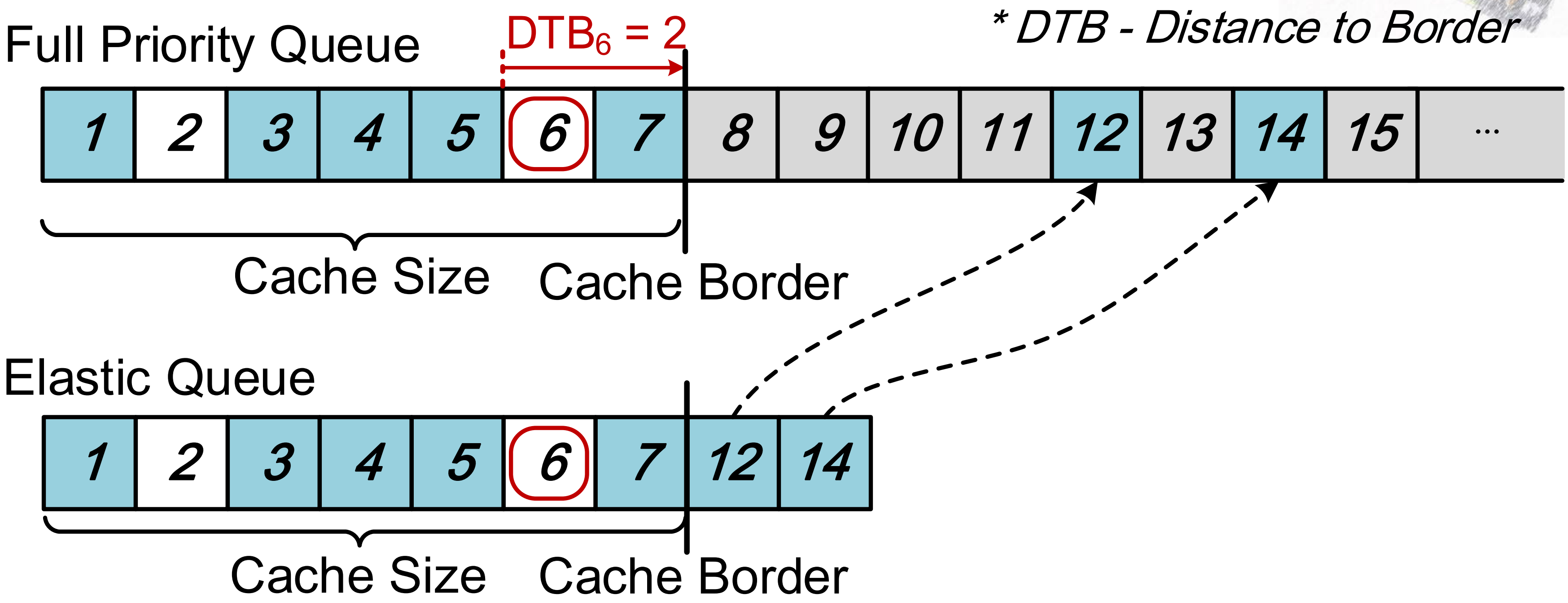
Block Pinning Module

Block UnPinning Module

Cache Priority Queue

Any cache policy

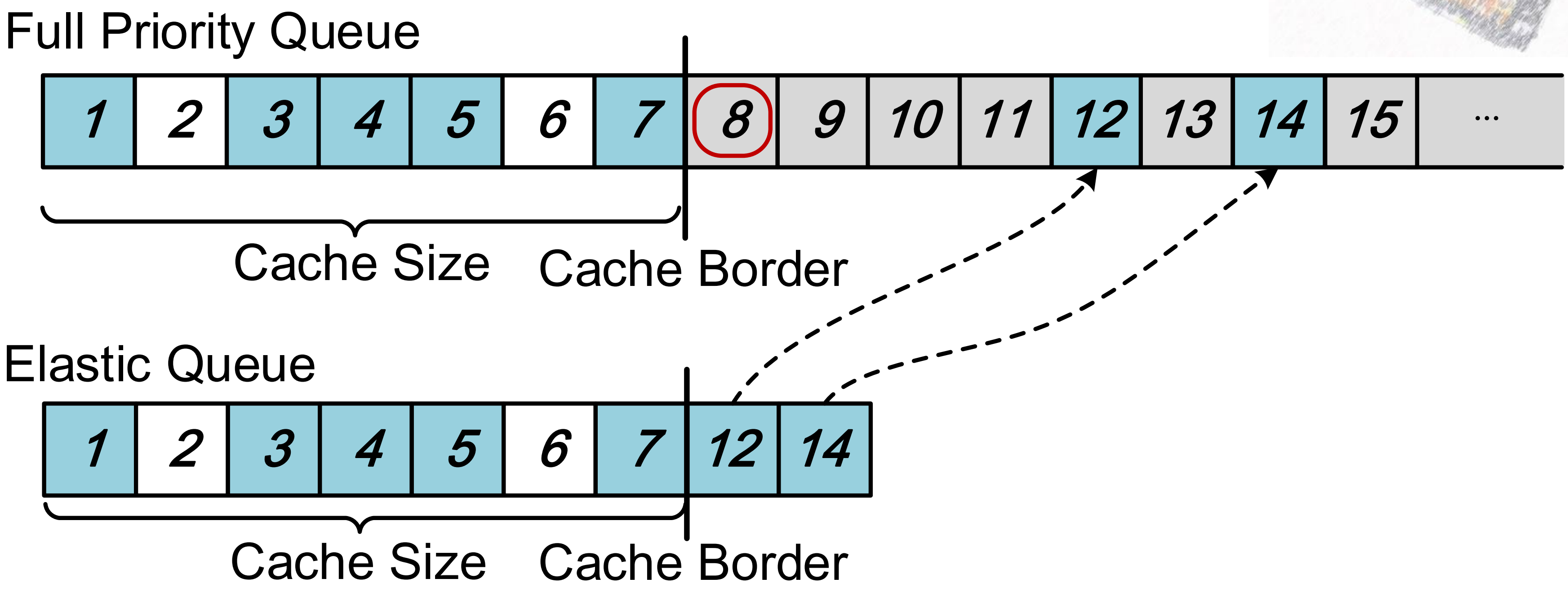
# Elastic Queue Design



□ **General blocks ahead of cache border**

- Only have metadata recorded in EQ
- e.g. block 6

# Elastic Queue Design



Full Priority Queue

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | … |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

Cache Size

Cache Border

Elastic Queue

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 12 | 14 |
|---|---|---|---|---|---|---|---|---|

Cache Size

Cache Border

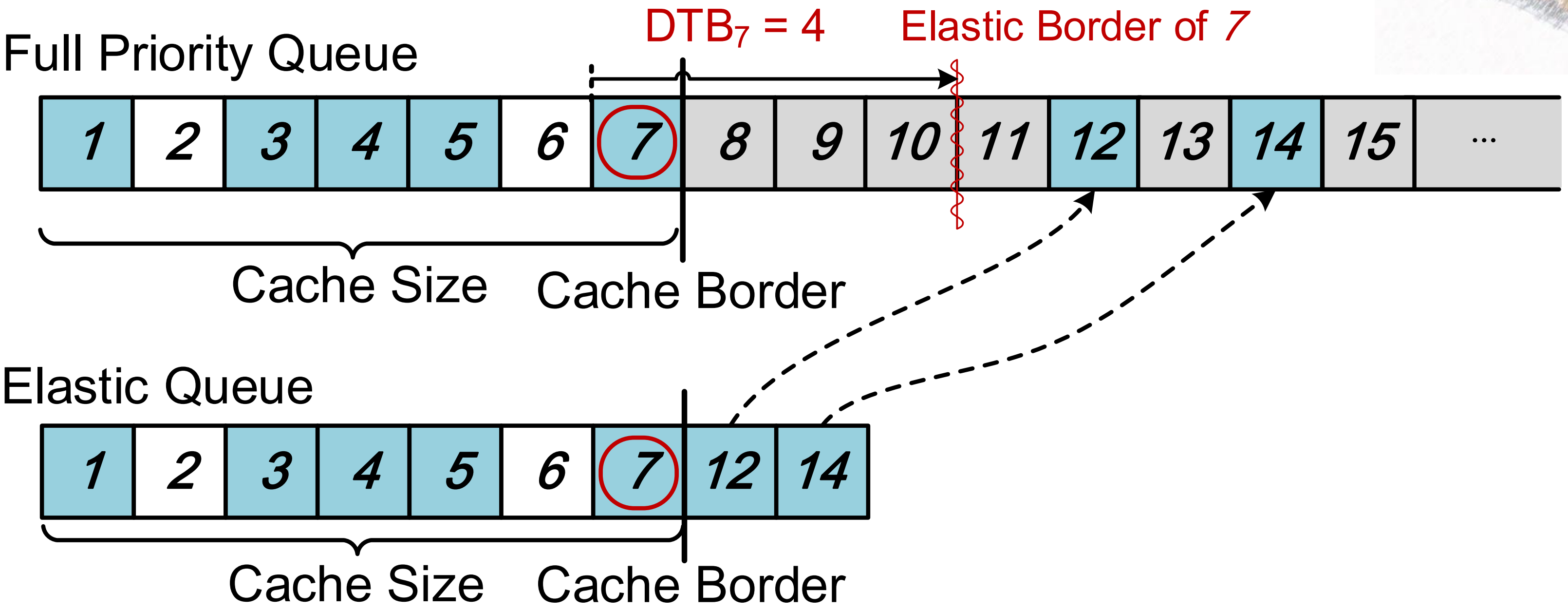
□ **General blocks ahead of cache border**

- Only have metadata recorded in EQ

▨ **General blocks behind cache border**

- Evicted, with no metadata in EQ
- e.g. block 8

# Elastic Queue Design



□ **General blocks ahead of cache border**

- Only have metadata recorded in EQ

▨ **General blocks behind cache border**

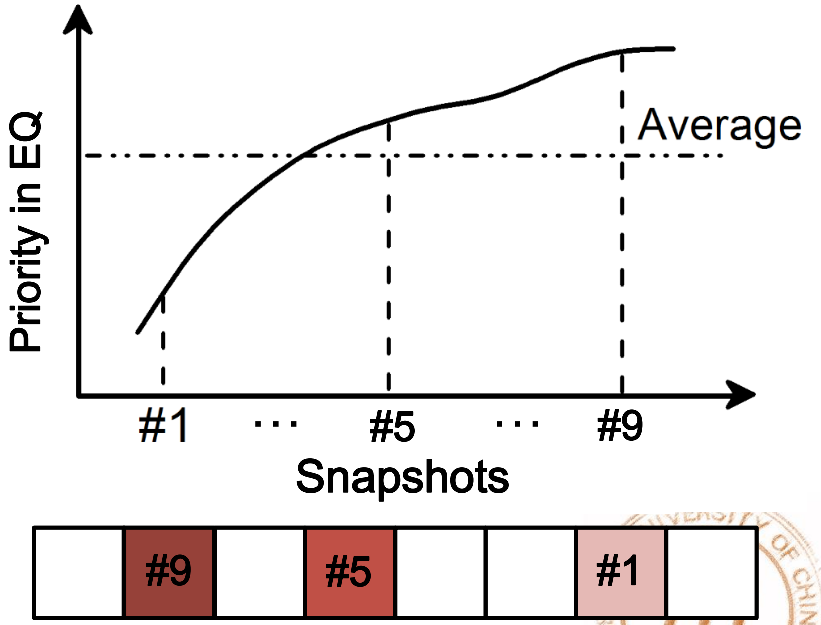
- Evicted, with no metadata in EQ

🟦 **Pinned blocks**

- Actually locate in SSD
- Assigned with elastic border
- e.g. border of block 7 is 4 steps further

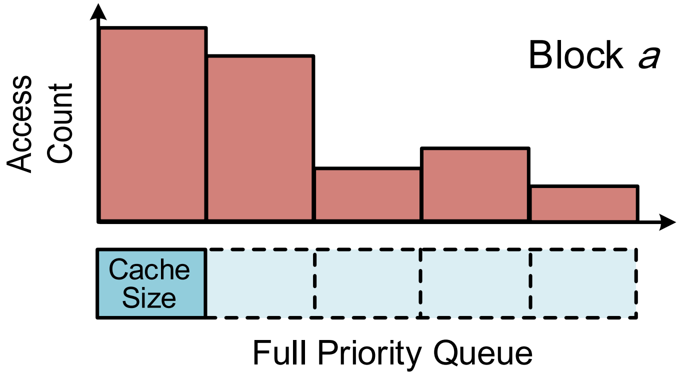
# Pinning Blocks

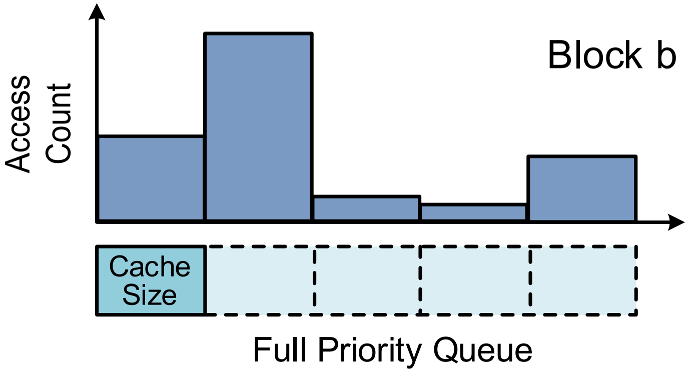
- Purpose
  - Loading the most popular blocks to SSD
- Timing
  - A free slot is available in SSD
- Selection criterion
  - Average priority
  - Changing tendency
- Mechanism
  - “Snapshot”
    - Short-term observation



# Unpinning Blocks

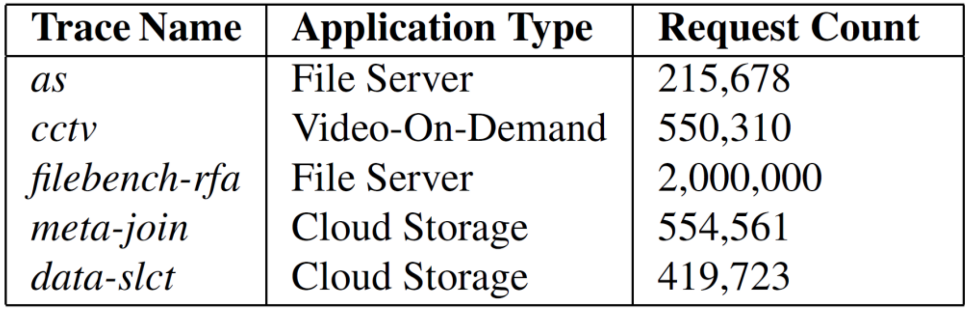
- Purpose
  - Determining where elastic borders should locate (DTB)
  - Evicting pinned blocks behind elastic borders
- DTB determination
  - Classifying data with access distributions
    - Long-term observation





# Evaluation

- Evaluation criterions
  - Cache hit ratio
  - Amounts of SSD written data
  - Write efficiency of SSD
- Traces



| Trace Name | Application Type | Request Count |
|---|---|---|
| *as* | File Server | 215,678 |
| *cctv* | Video-On-Demand | 550,310 |
| *filebench-rfa* | File Server | 2,000,000 |
| *meta-join* | Cloud Storage | 554,561 |
| *data-slct* | Cloud Storage | 419,723 |

- Coupled cache algorithms
  - LRU, LIRS, LARC

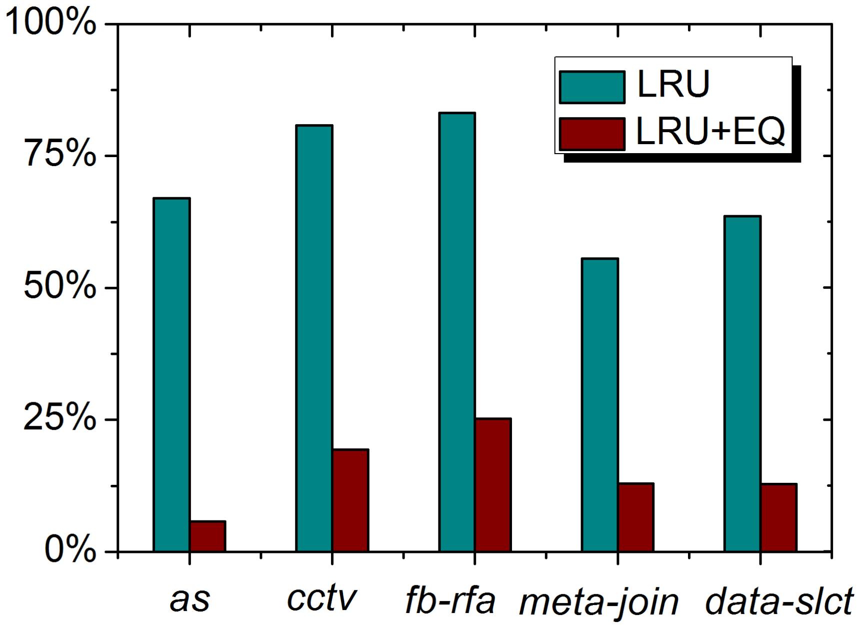
# Overall Results

* For LRU, LIRS, and LARC under all the five traces

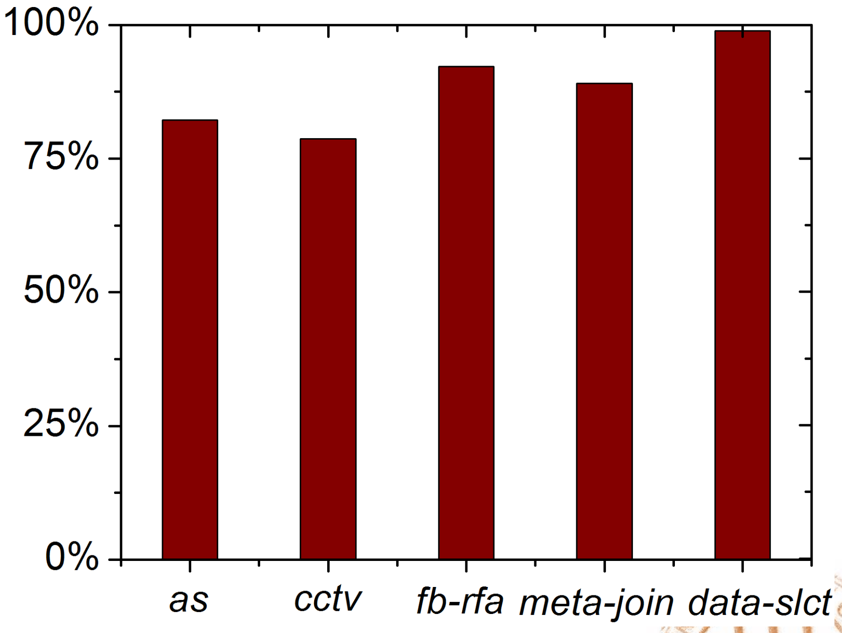
- Cache hit ratio
    - Higher in 66.67% of the cases
    - Average improvement – 17.30%
- Amounts of SSD written data
    - Reduce 39.03 times on average
- Write efficiency of SSD
    - 45.78 times enlarged on average

# Effectiveness of EQ

- Reduction of no-hit percentage

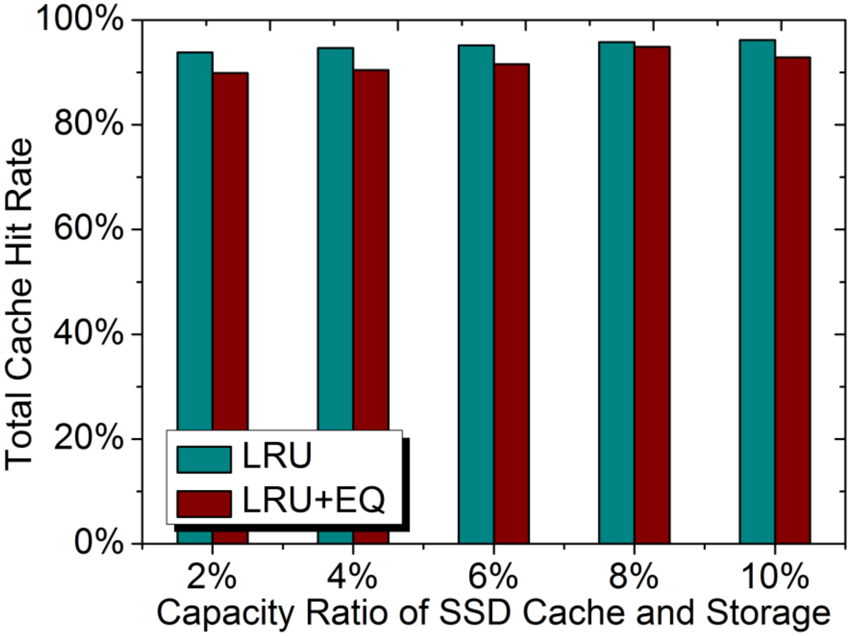


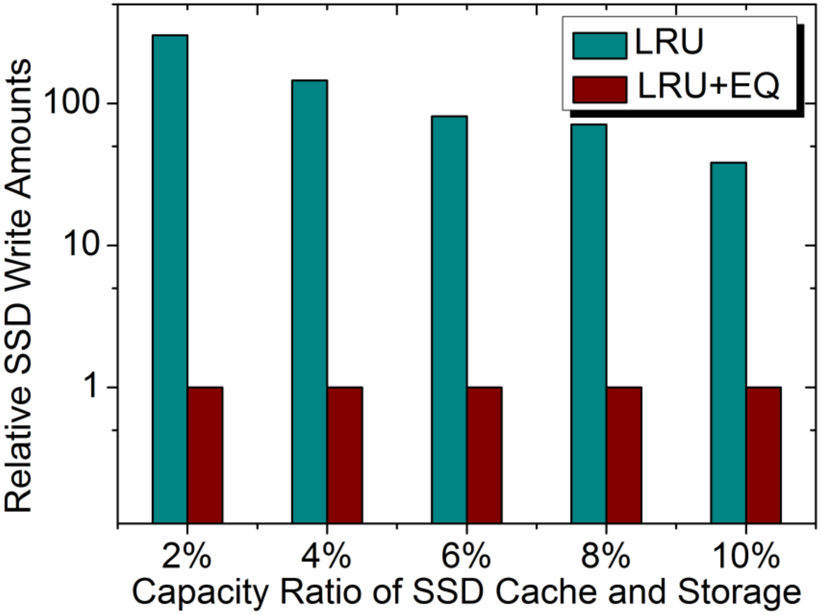
- Hotness of pinned blocks



# Parameter Settings

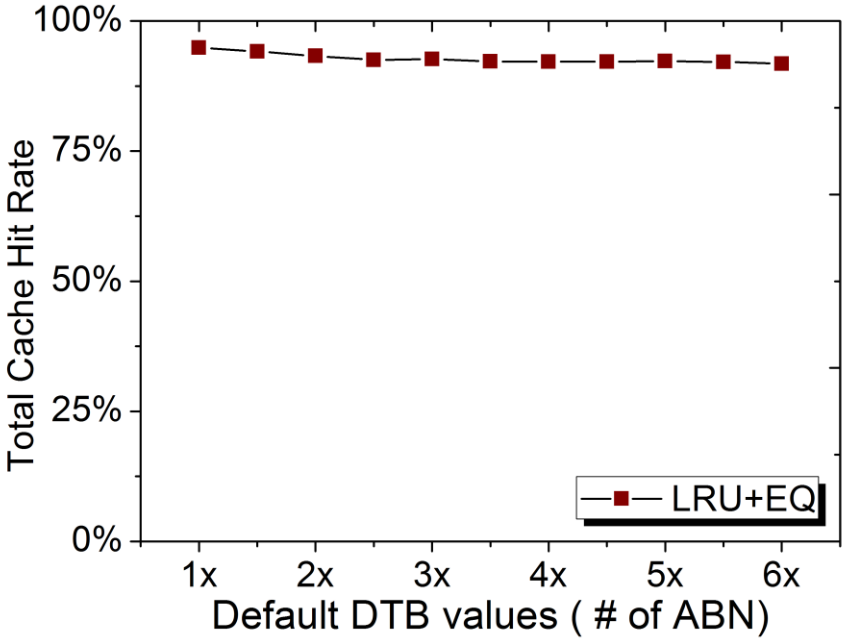
- Impact of SSD Size

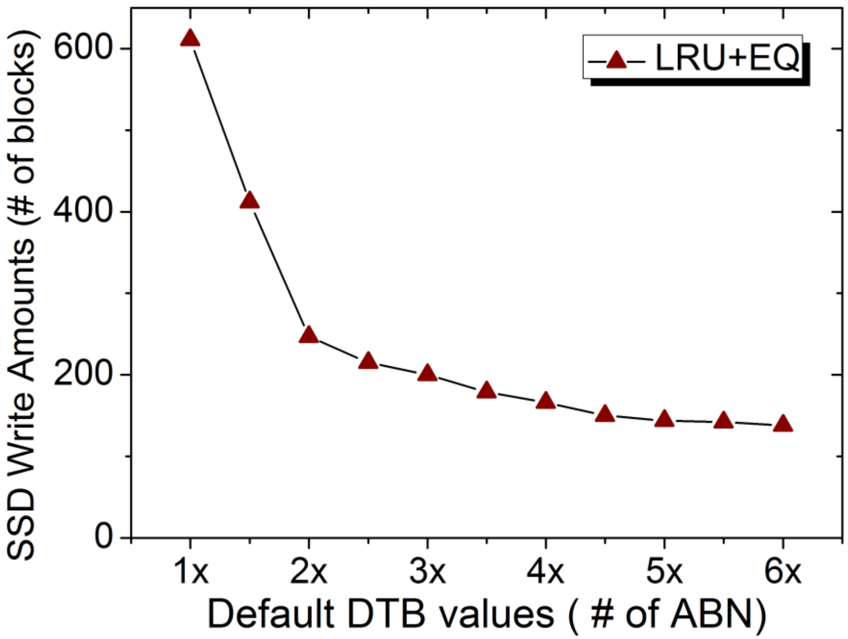




# Parameter Settings

- Impact of default distance-to-border





# Summary

- A universal SSD lifetime enhancement plug-in
  - Couple with any cache algorithm
  - Reduce SSD write amount
- A unified priority queue model for cache algorithms
- Make use of coupled cache policy
  - Priority Snapshot
  - Priority Distribution

# Thank you !

Q&A