CDAC: Content-Driven Deduplication-Aware Storage Cache

Yujuan Tan, Jing Xie, Congcong Xu, Zhichao Yan,
Hong Jiang, Yajun Zhao, Min Fu, Xianzhang Chen,
Duo Liu, Wen Xia

Outline

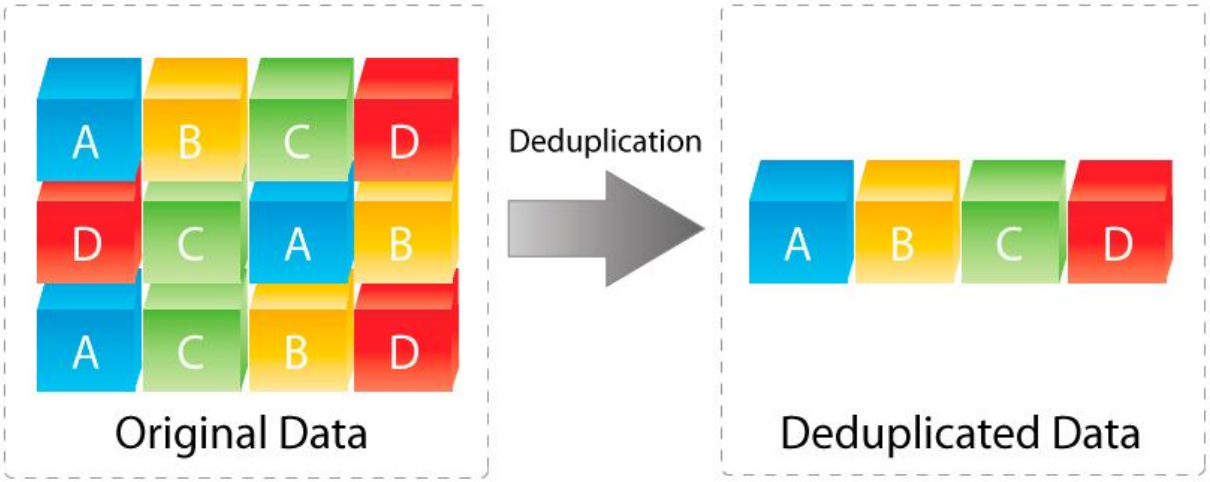
- Introduction and Motivation
- Design of CDAC
- Performance Evaluation
- Conclusion

Outline

- Introduction and Motivation
- Design of CDAC
- Performance Evaluation
- Conclusion

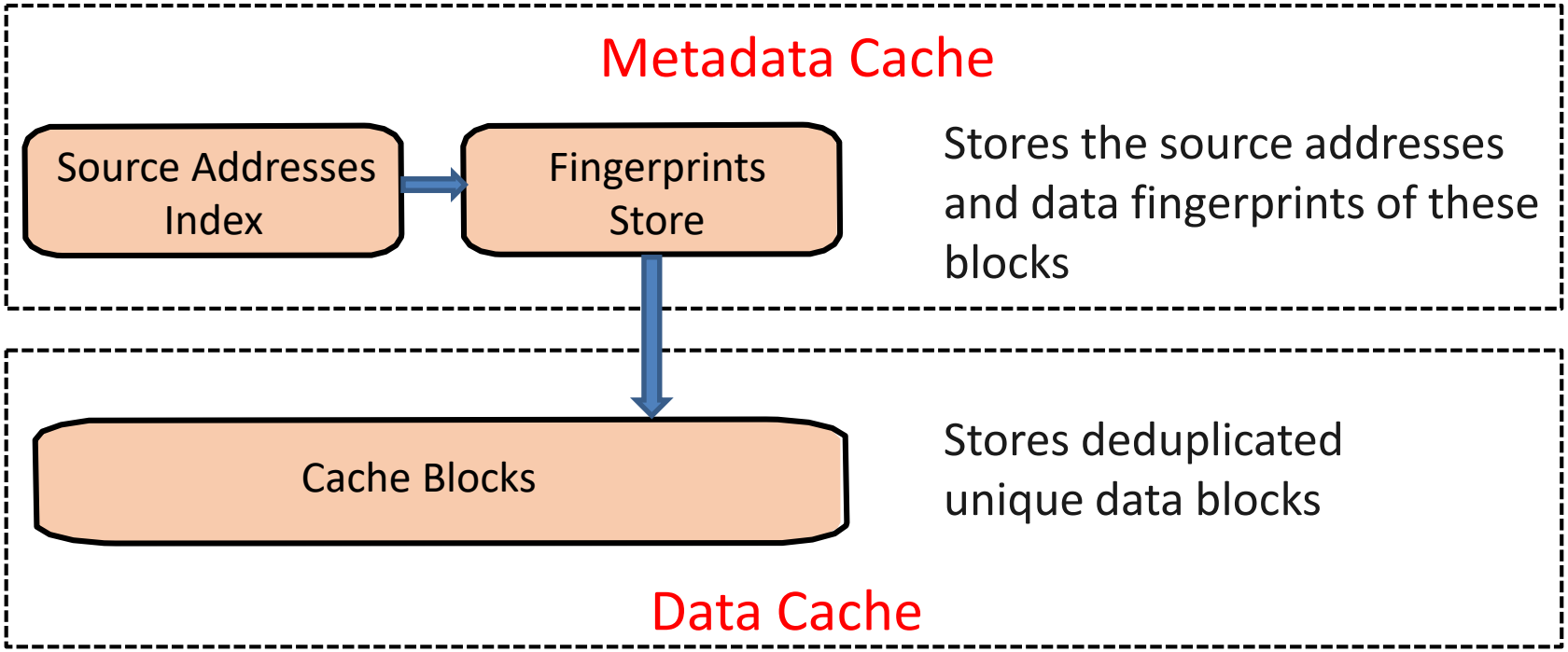
Cache Deduplication

- The deduplication overhead, would degrade the performance of the overall storage system
- Be carefully designed and managed to reap the benefits of increased logical capacity and cache hit ratios



Identifying and removing redundant data to reduce data footprint

CacheDedup

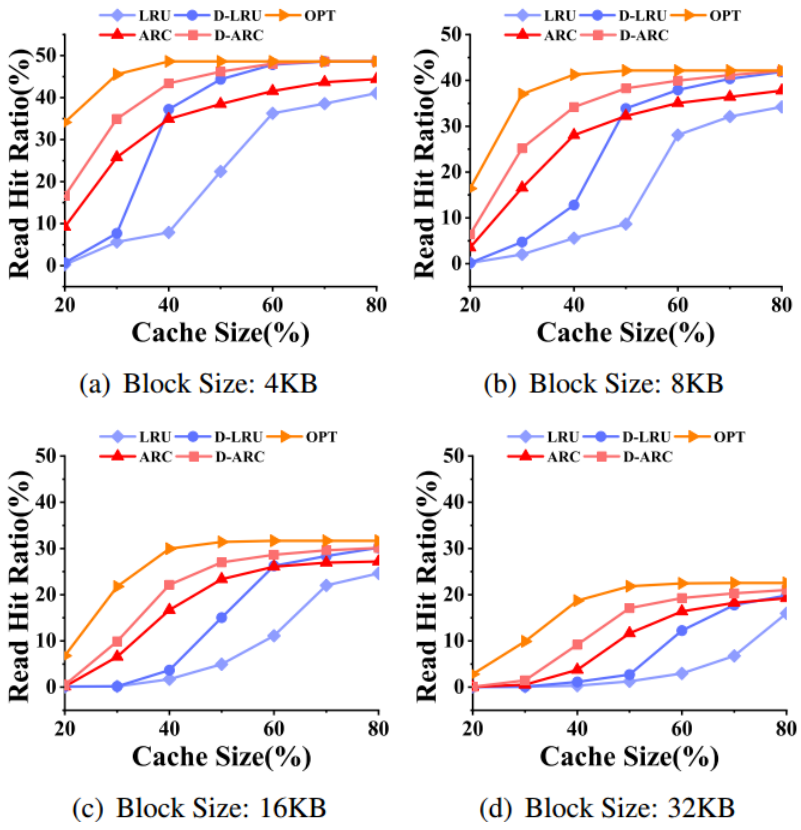


- **D-LRU, D-ARC** are designed based on this architecture[1]
- Data Cache and Metadata Cache are separately managed and accessed

[1] LI, W., JEAN-BAPTISE, G., RIVEROS, J., NARASIMHAN, G.,ZHANG, T., AND ZHAO, M. CacheDedup: In-line Deduplication for Flash Caching. In USENIX FAST'16 (Feb. 2016).

Analysis of D-ARC and D-LRU

- Read hit ratio from WebVm



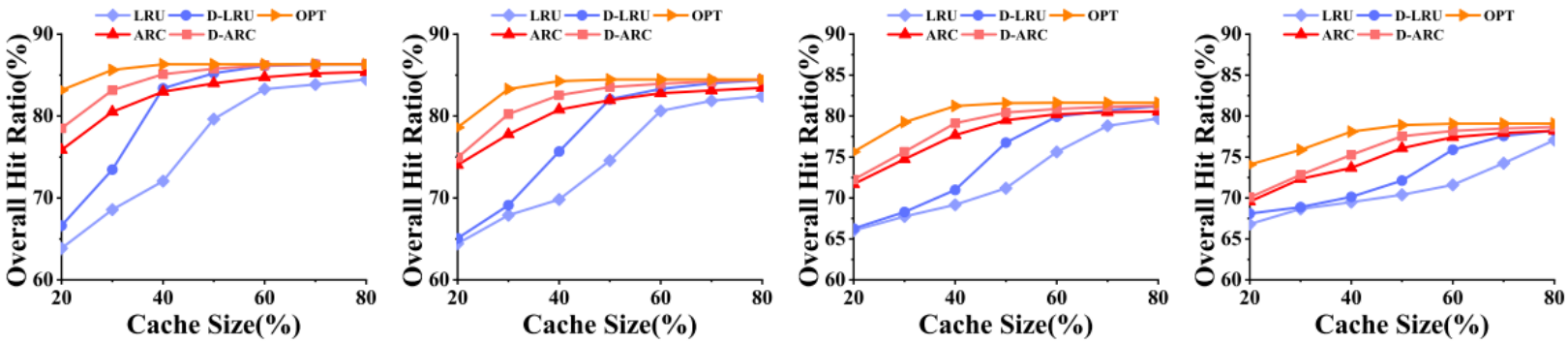
- For 4KB, D-ARC and D-LRU is 6.91% and 11.85% higher than ARC and LRU on average
- For 8KB, 16KB and 32KB, D-ARC are 5.38%, 3.00% and 2.65% higher than ARC on average, and D-LRU are 8.70%, 5.58% and 3.77% higher than LRU on average

- For 4KB, the read hit ratios of D-LRU and D-ARC are 76.5% and 89.2% of that of OPT
- For 8KB, 16KB and 32KB, the read hit ratios of D-LRU are only 31%, 12.3% and 6% of that of OPT, and D-ARC's read hit ratios are only 82.9%, 73.9% and 49% of that of OPT

(Cache size:40%)

Analysis of D-ARC and D-LRU

- Overall hit ratio from WebVM



(a) Block Size: 4KB (b) Block Size: 8KB (c) Block Size: 16KB (d) Block Size: 32KB

As the block size increases, the benefits of deduplication become limited and their hit ratios decreases significantly

Existing algorithm analysis

Analysis

- Do Not fully utilize the characteristics of deduplication

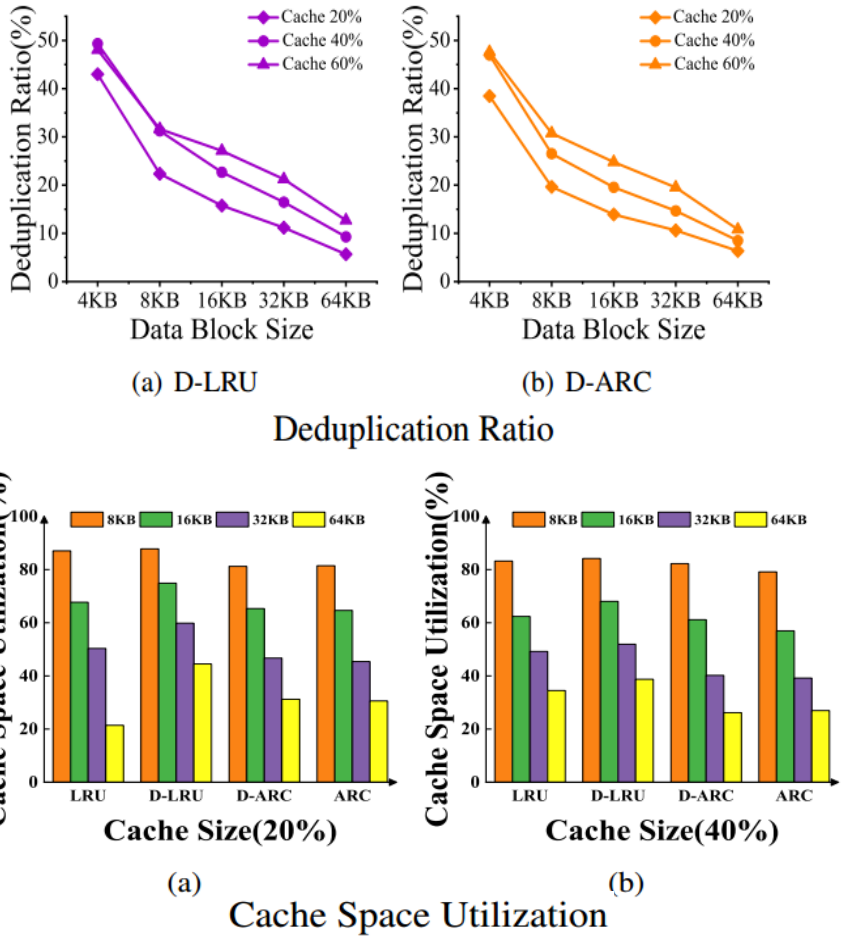
  ——Missed the opportunity to effectively leverage the intensity of content redundancy and sharing.

  ——Based on this discovery, we proposed RCE technology

- Cache space utilization is also low

  ——Read/write alignment causes a large amount of invalid data in the cache.

  ——Based on this discovery, we proposed BHI technology



Outline

- Introduction and Motivation
- Design of CDAC
- Performance Evaluation
- Conclusion

CDAC Design

- Architecture
  - Based on CacheDedup architecture
    ——Data Cache stores the data blocks, Metadata Cache stores the source addresses and data fingerprints of these blocks
    ——The source address and its corresponding data block do not need to be fetched in or evicted out synchronously

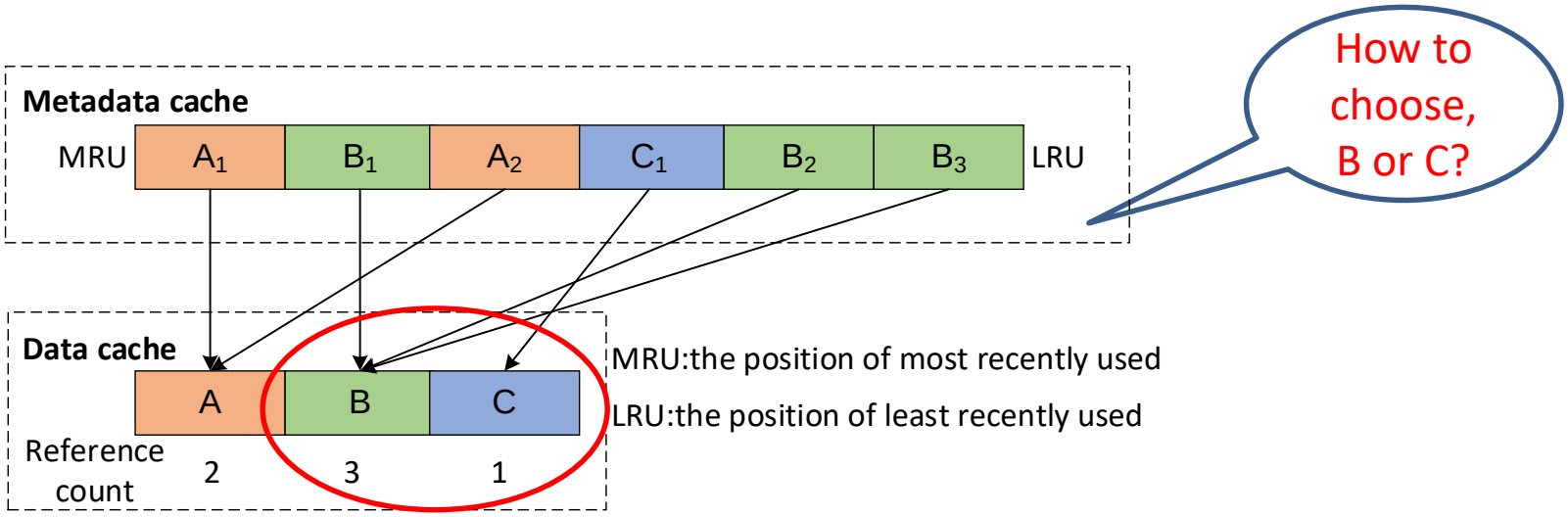
CDAC focus on how to select the source addresses to be deleted from Metadata Cache to generate the free blocks to improve the cache hit ratios

    ——More accurately identify the content hotness of the block, especially for large blocks, minimizing false-positive hot block identifications

- Terminology
  - Free block
    —— The block in Data Cache that there is no source address in Metadata cache pointing to it
  - Reference count
    ——The total number of the source addresses pointing to that block

Referenced-Count based Eviction（RCE）

- The higher the reference count, the more requests will be associated with this data block, and so the hotter the data block wil be
- However, using reference counting as the only hint to find the block to be replaced is not sufficiently effective



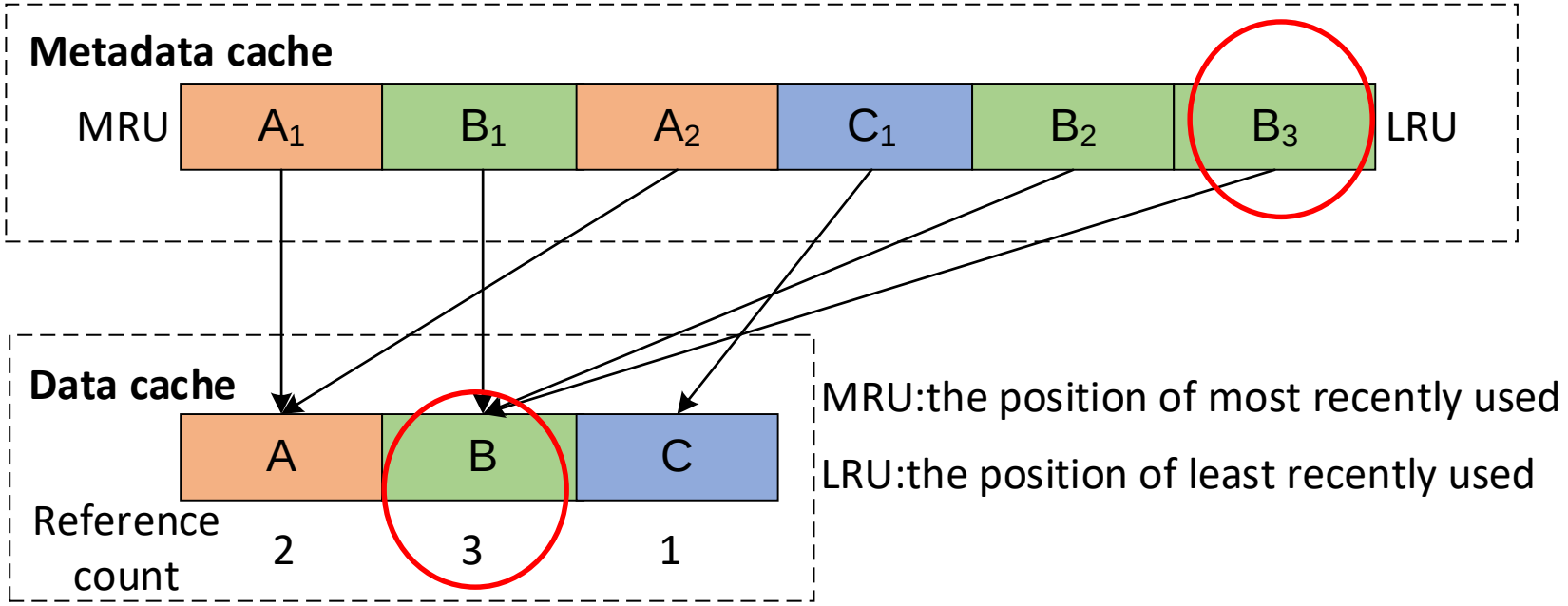
RCE takes both reference counts and access locality into consideration

Referenced-Count based Eviction（RCE）

- RCE only focuses on the source addresses in the LRU position
- RCE divides the data blocks into two categories
  - Category 1: the data block that is referenced only once
    - ——No other source address is associated with it, RCE will delete it
  - Category 2: the block that is referenced multiple times
    - ——Move the source address to the MRU position to keep it
    - ——Further observe how the reference count changes in the next cycle
    - ——If the rate of decline exceeds the threshold in the next cycle, the source address is deleted

- A cycle: The time required for the source address to go from the MRU position to the LRU position

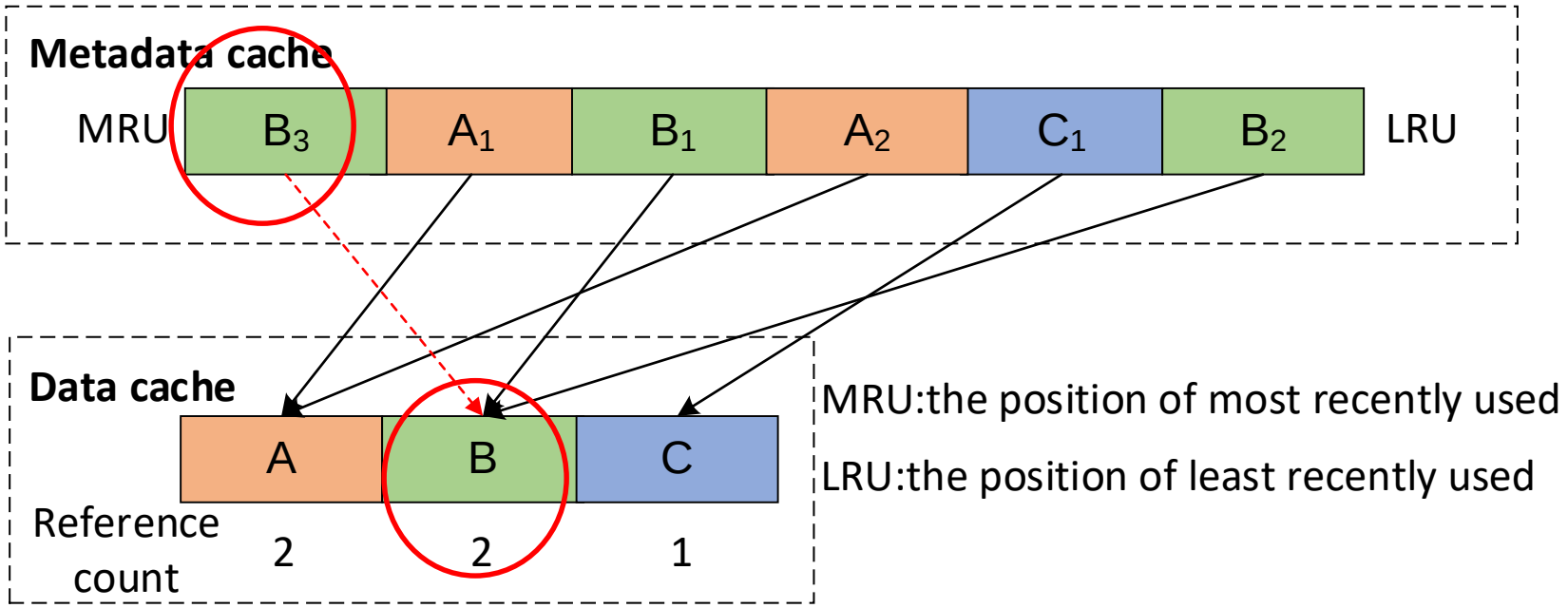
Referenced-Count based Eviction（RCE）



- Access Order:

——B3, B2, C1, A2, B1, A1, D1, B3, E1, F1……

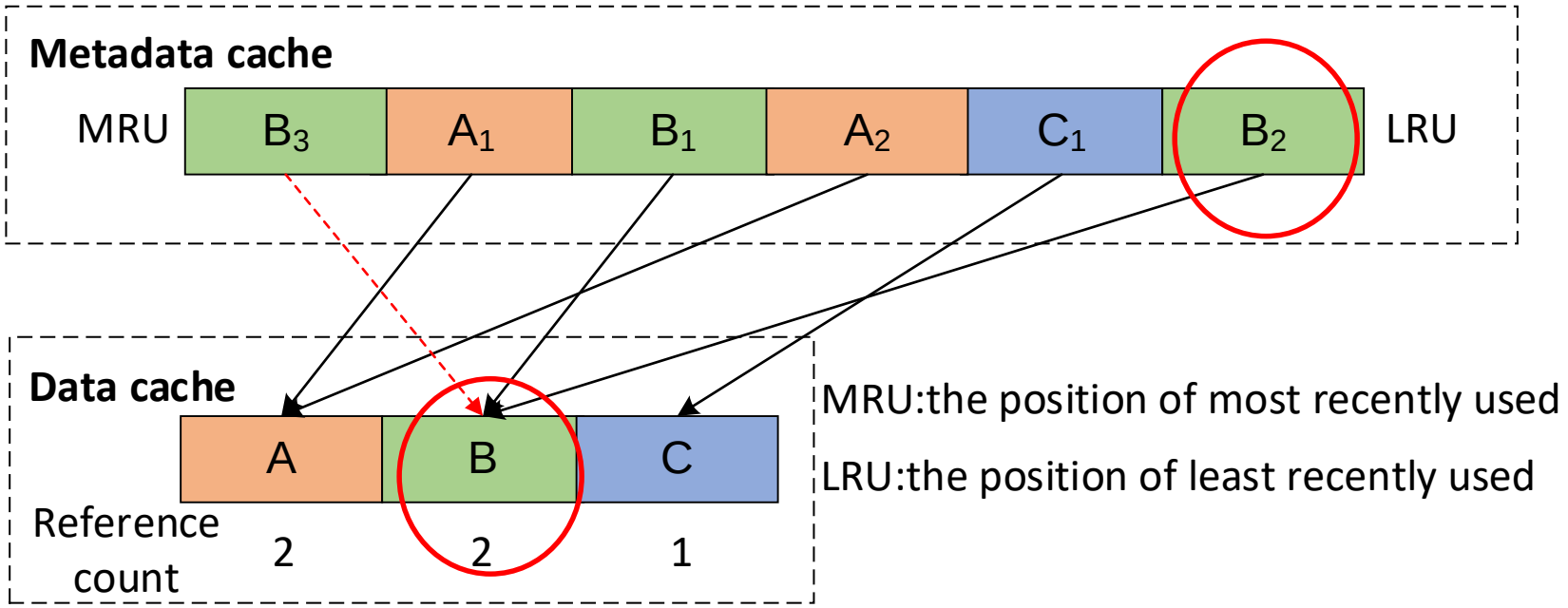
Referenced-Count based Eviction（RCE）



- Access Order:

——B3, B2, C1, A2, B1, A1, D1, B3, E1, F1……

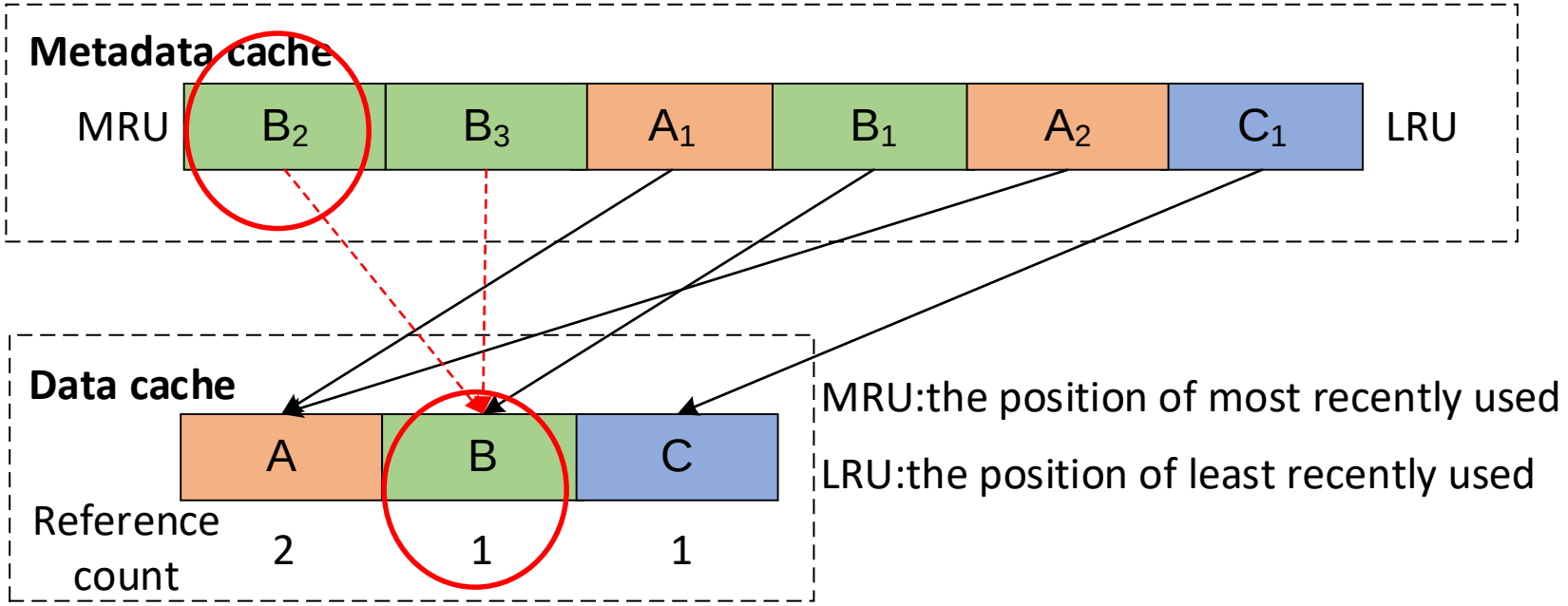
Referenced-Count based Eviction（RCE）



- Access Order:

——B3, B2, C1, A2, B1, A1, D1, B3, E1, F1……

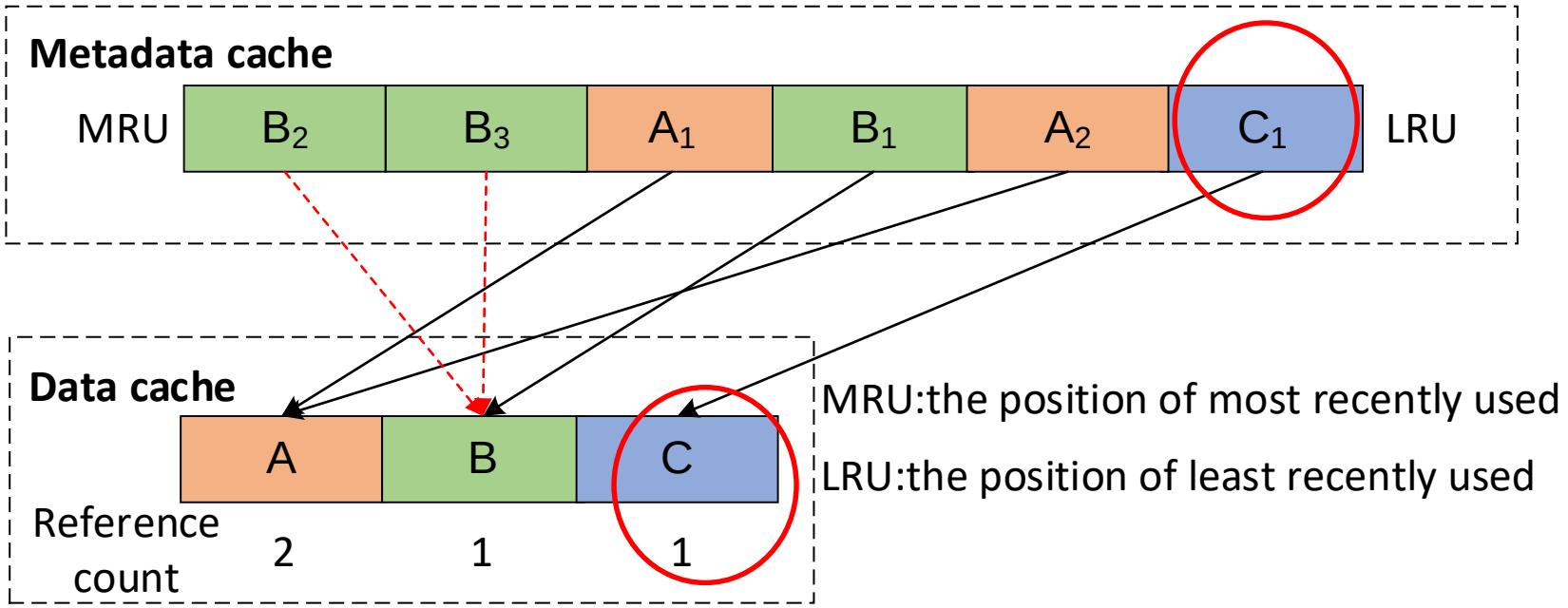
Referenced-Count based Eviction（RCE）



- Access Order:

——B3, B2, C1, A2, B1, A1, D1, B3, E1, F1……

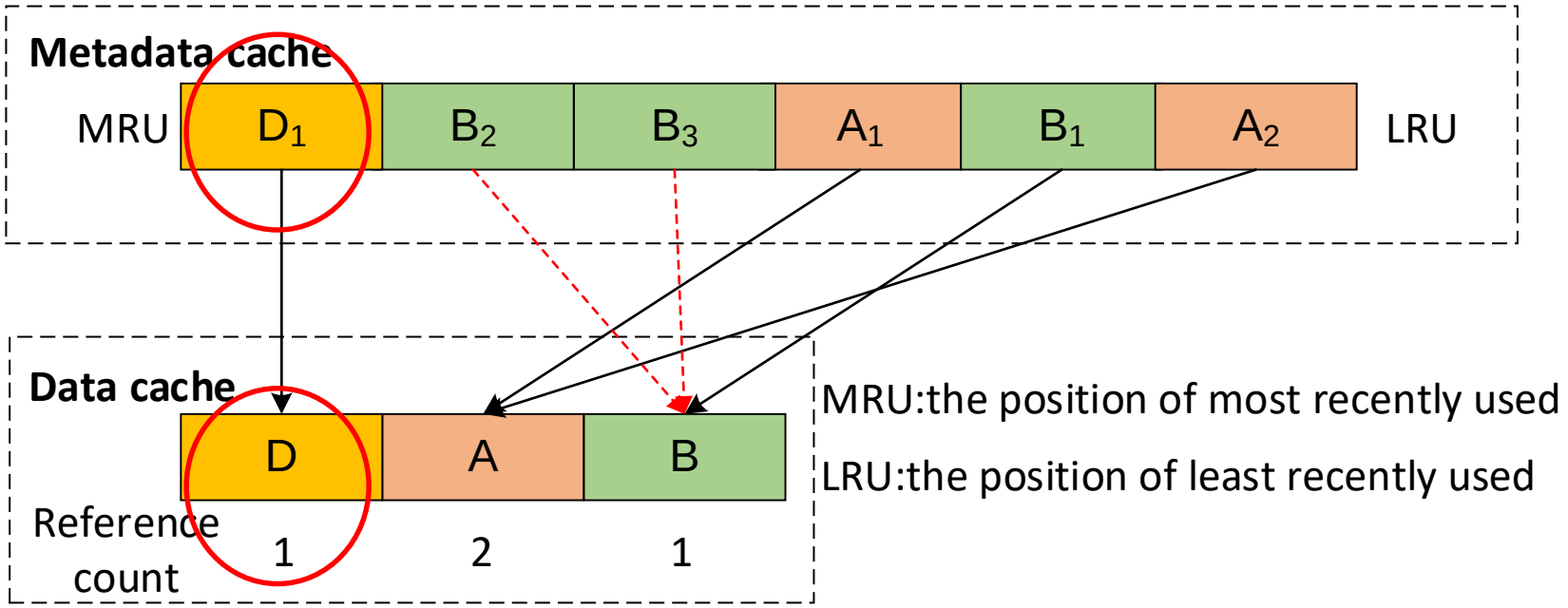
Referenced-Count based Eviction（RCE）



- Access Order:

——B3, B2, C1, A2, B1, A1, D1, B3, E1, F1……

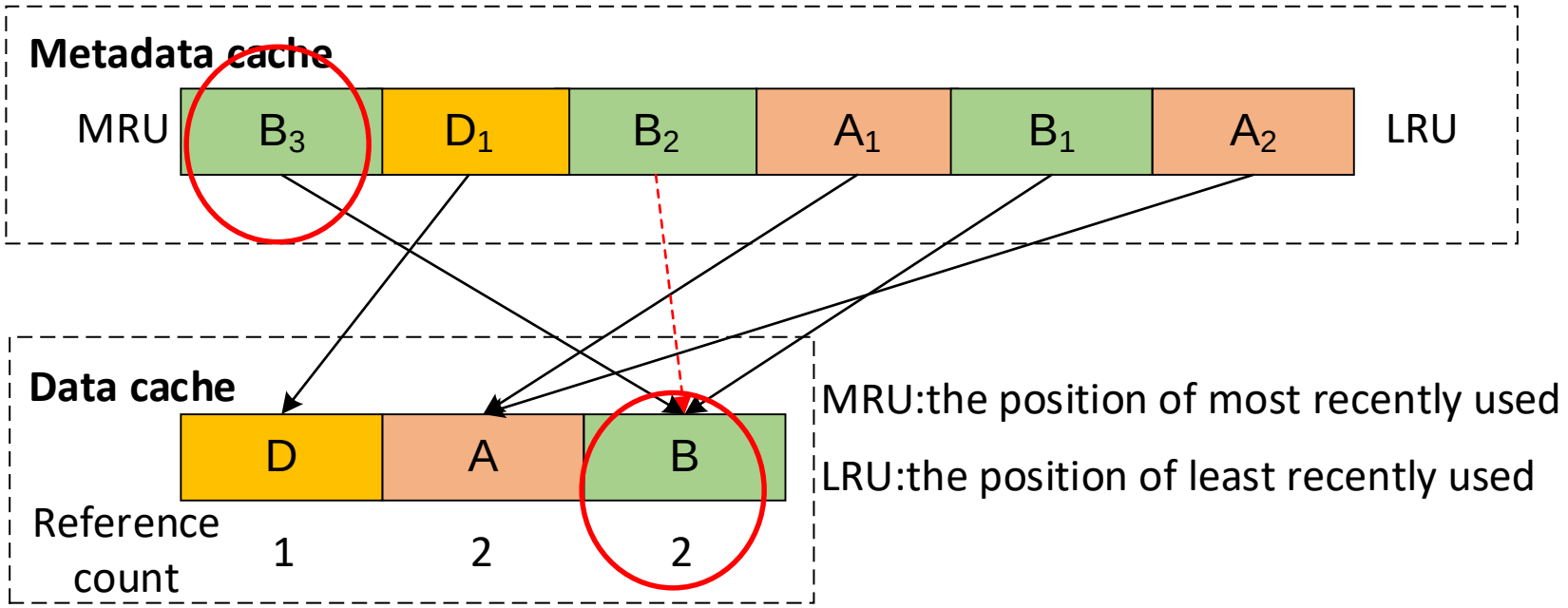
Referenced-Count based Eviction（RCE）



- Access Order:

——B3, B2, C1, A2, B1, A1, D1, B3, E1, F1……

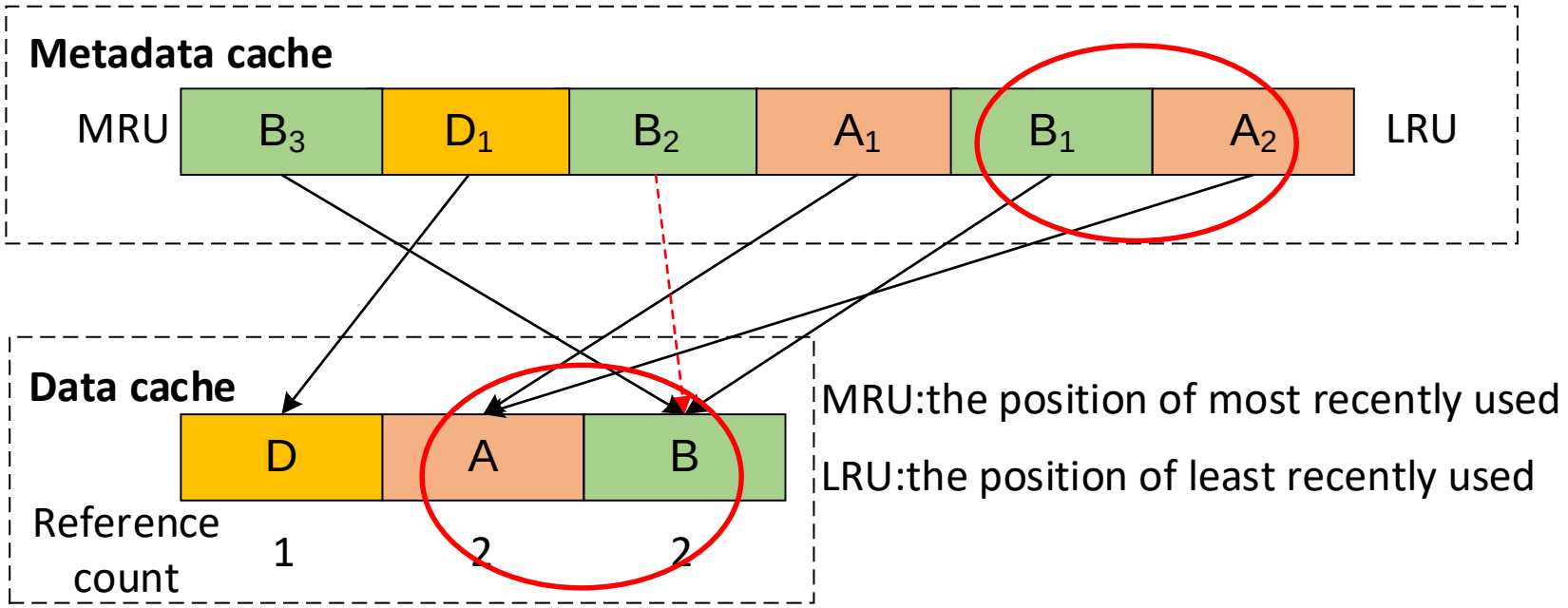
Referenced-Count based Eviction（RCE）



- Access Order:

——B3, B2, C1, A2, B1, A1, D1, B3, E1, F1……

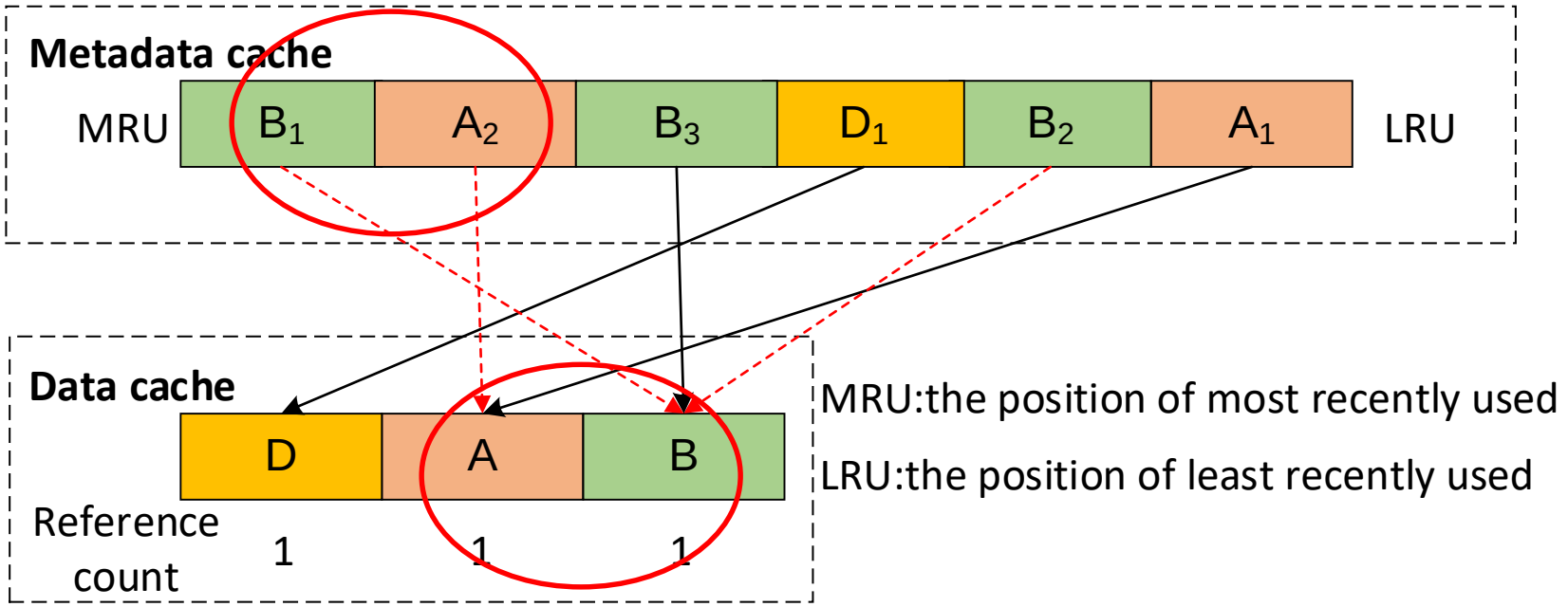
Referenced-Count based Eviction（RCE）



- Access Order:

——B3, B2, C1, A2, B1, A1, D1, B3, E1, F1……

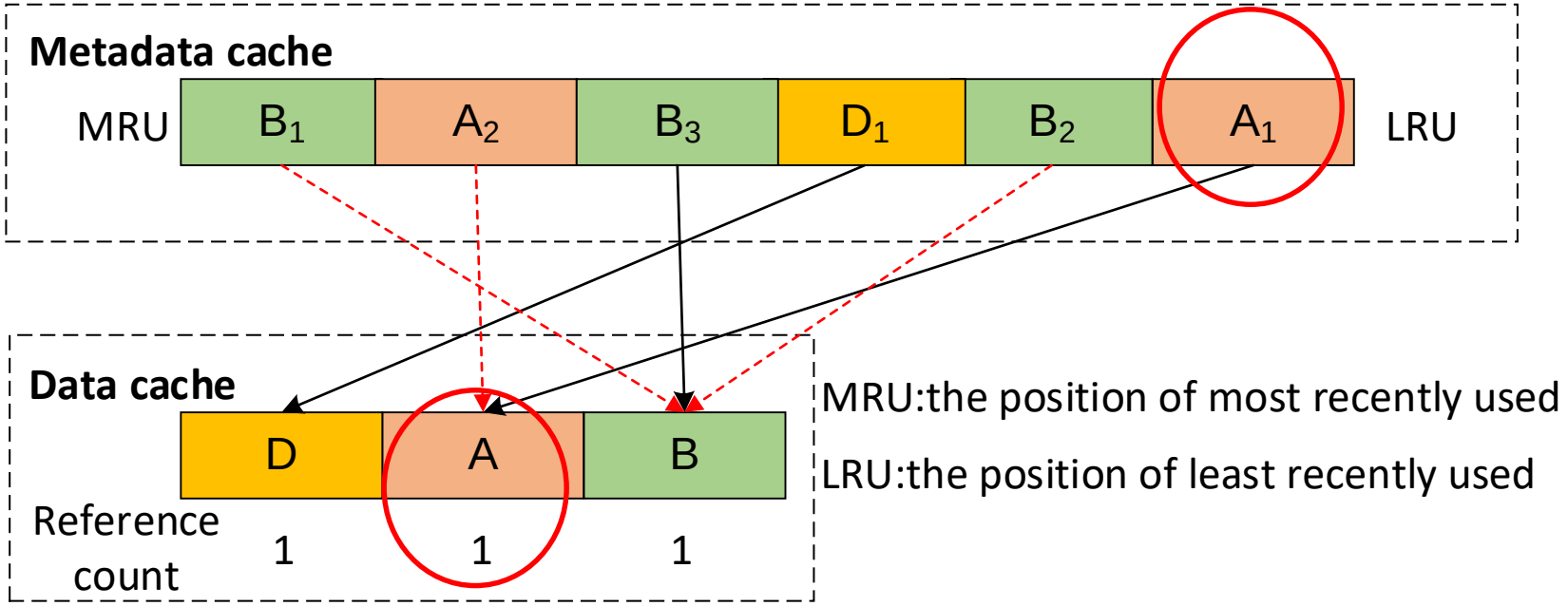
Referenced-Count based Eviction（RCE）



- Access Order:

——B3, B2, C1, A2, B1, A1, D1, B3, E1, F1……

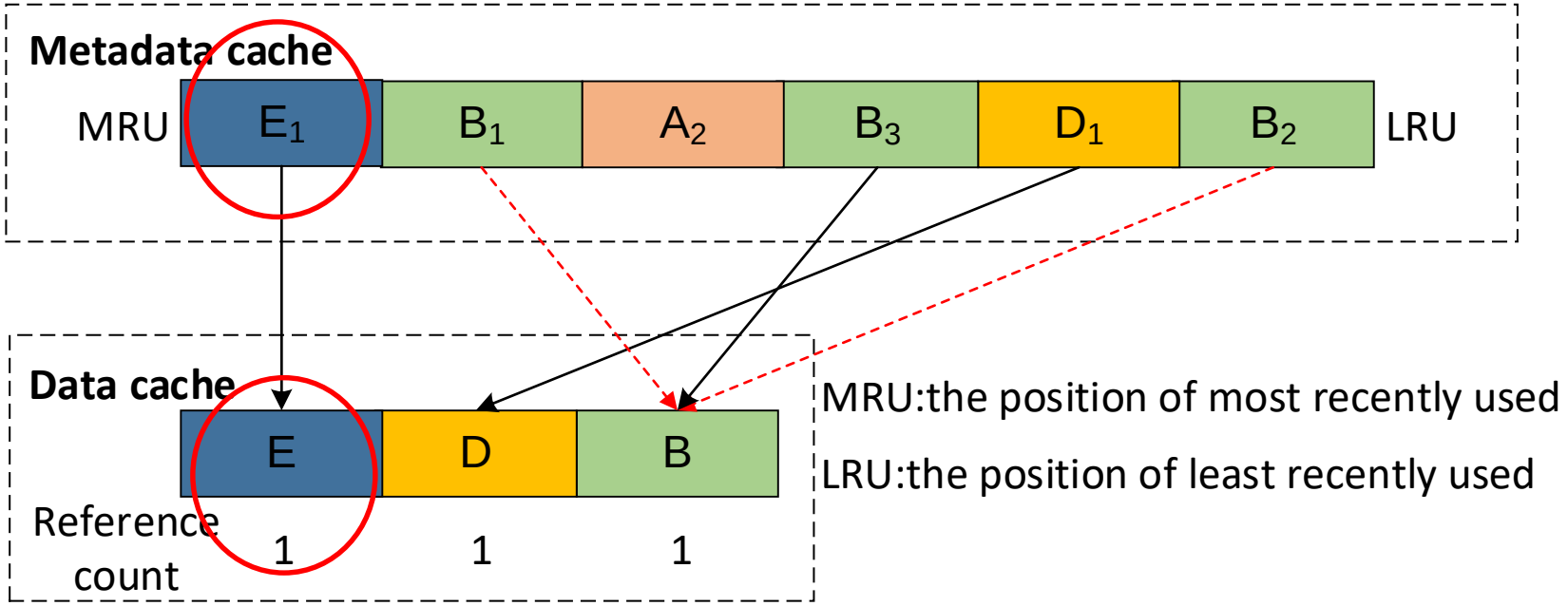
Referenced-Count based Eviction（RCE）



- Access Order:

——B3, B2, C1, A2, B1, A1, D1, B3, E1, F1……

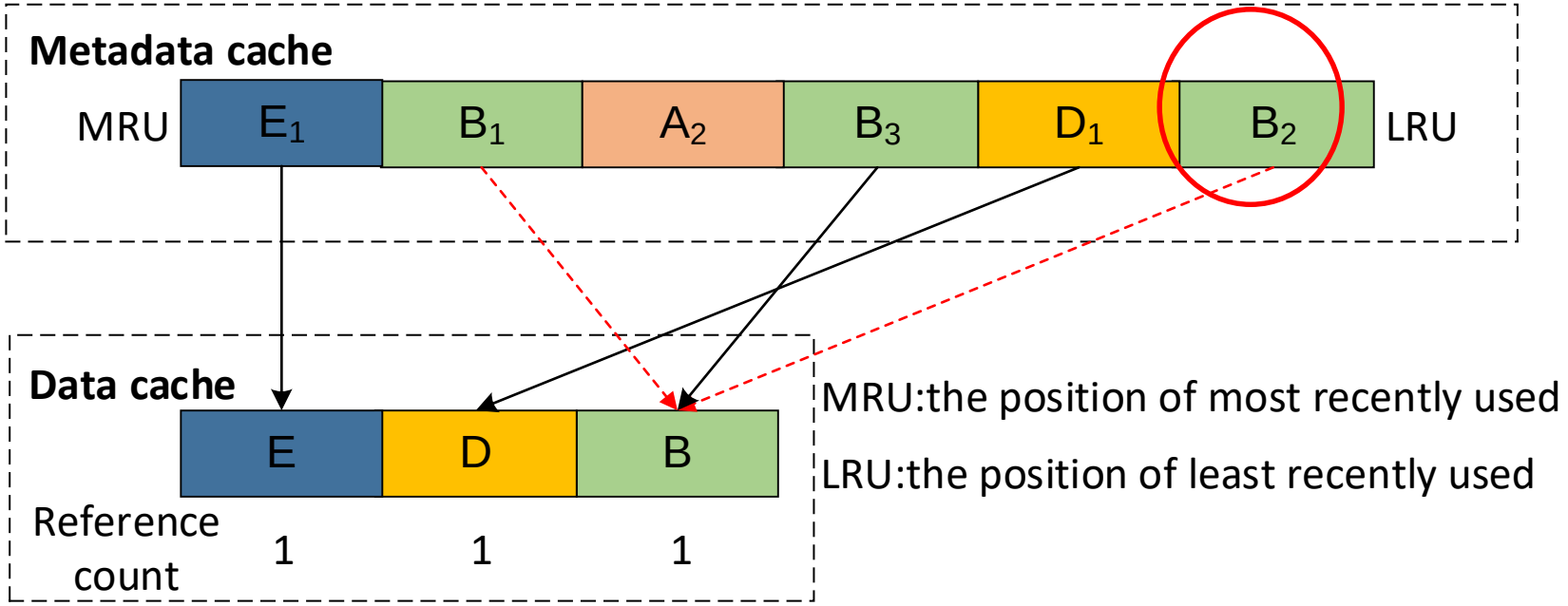
Referenced-Count based Eviction（RCE）



- Access Order:

——B3, B2, C1, A2, B1, A1, D1, B3, E1, F1……

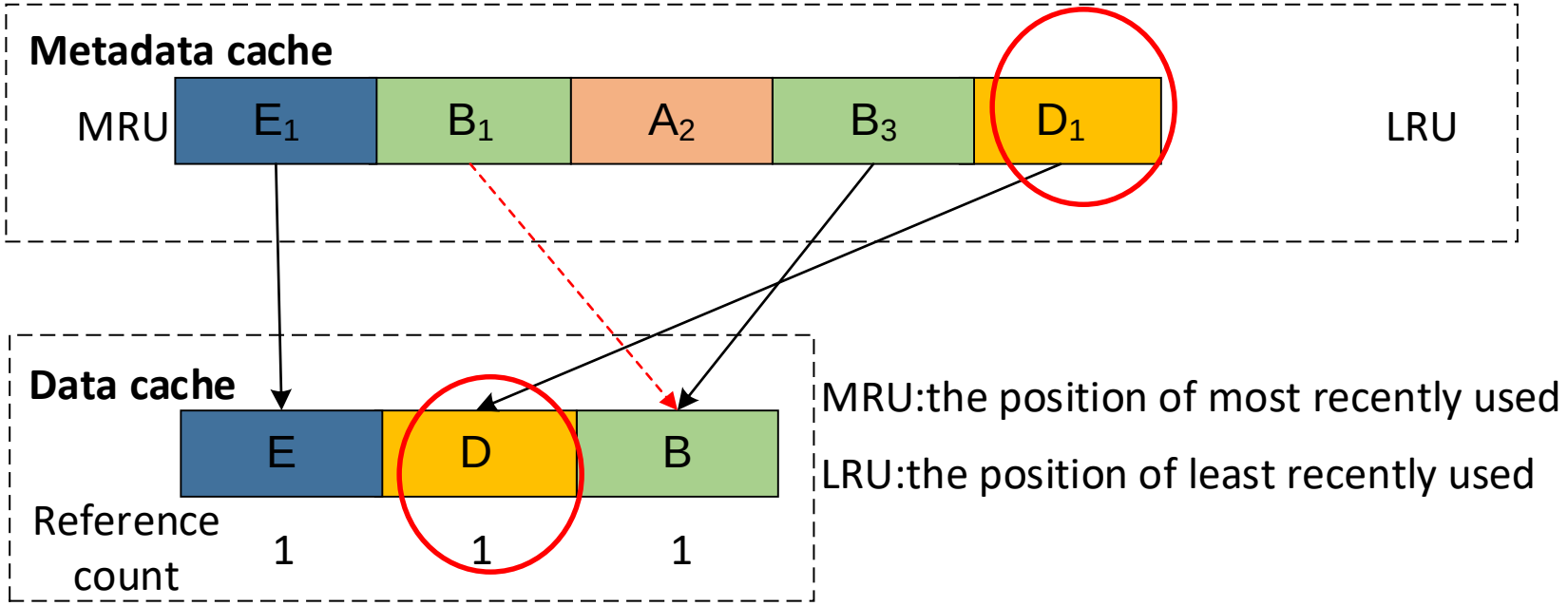
Referenced-Count based Eviction（RCE）



- Access Order:

——B3, B2, C1, A2, B1, A1, D1, B3, E1, F1……

Referenced-Count based Eviction（RCE）



- Access Order:

——B3, B2, C1, A2, B1, A1, D1, B3, E1, F1……

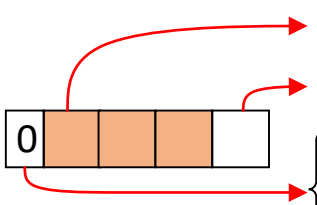
Bitmap based Hotness Identification（BHI）

- Read and write alignment causes the waste of much cache space

——The hotness/coldness recognition of traditional algorithms has ignored the valid content of each cached block for each access.

- BHI identifies hot/cold blocks based on finer-grained access

——breaks a block into multiple small parts and then uses bitmaps to record the access status of each part



represents for the accessed part

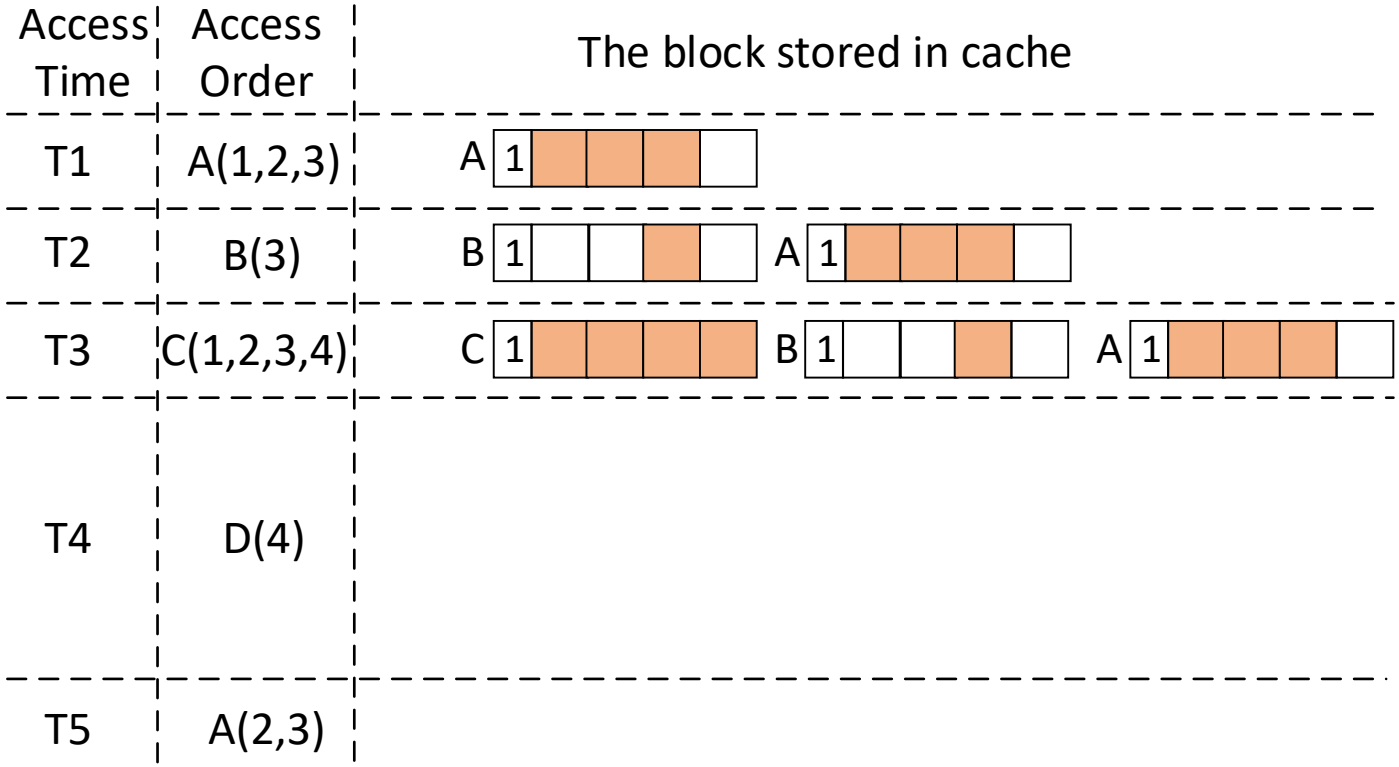
represents for the non-accessed part

1： the block has been accessed in the current cycle

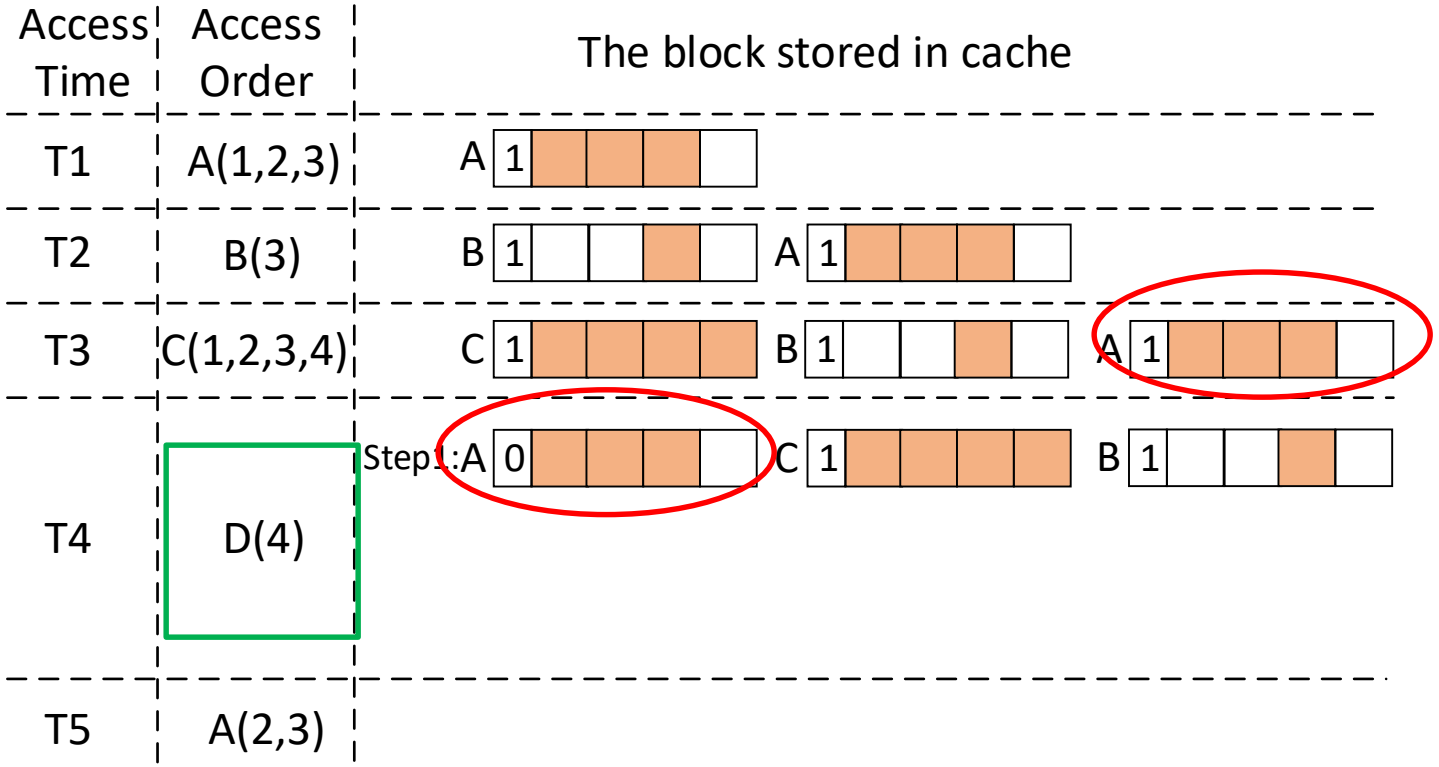
0： the block has not been accessed in the current cycle

- Flag: used to identify the source addresses that have not been accessed for a long time, even if their access parts greatly exceed the threshold
- The current cycle: The time required for the source address to go from the MRU position to the LRU position

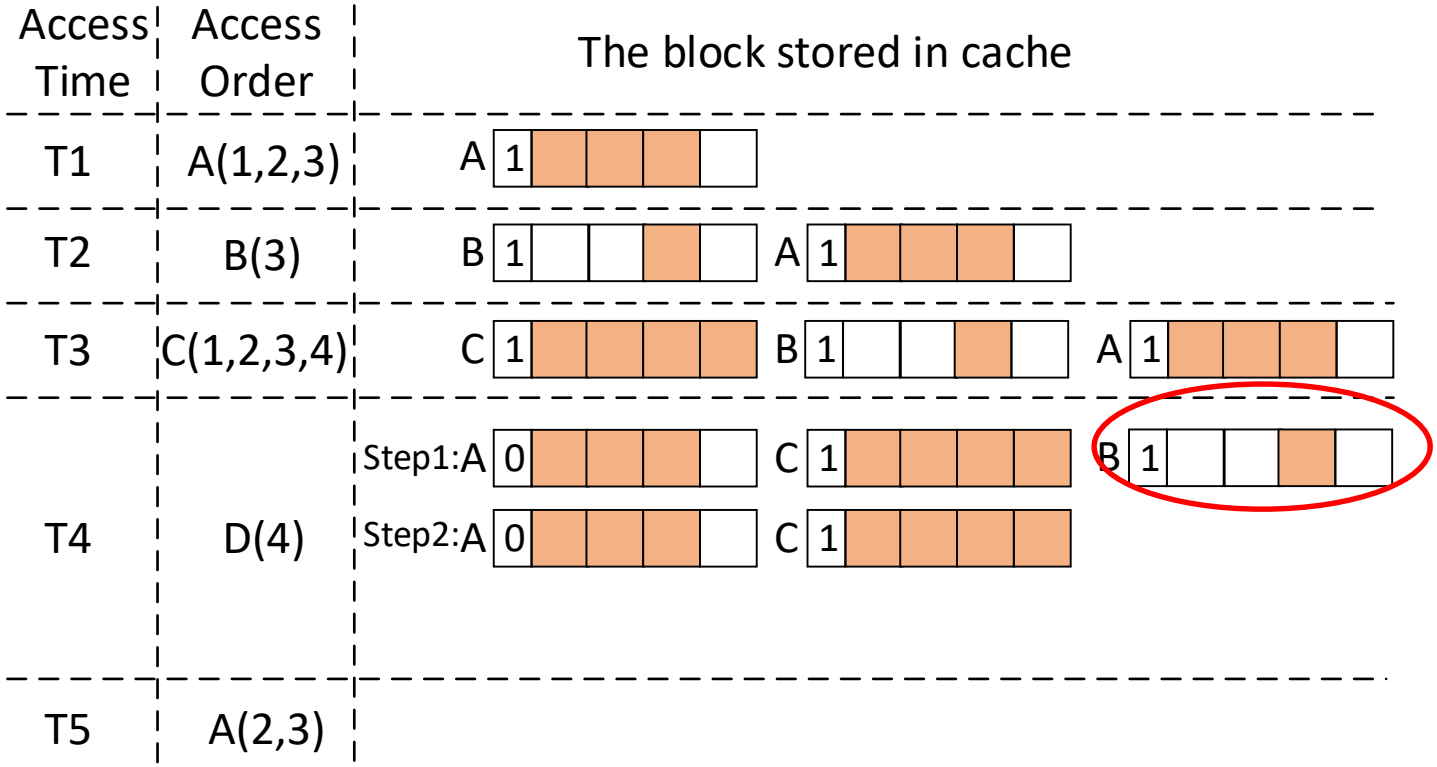
Bitmap based Hotness Identification（BHI）



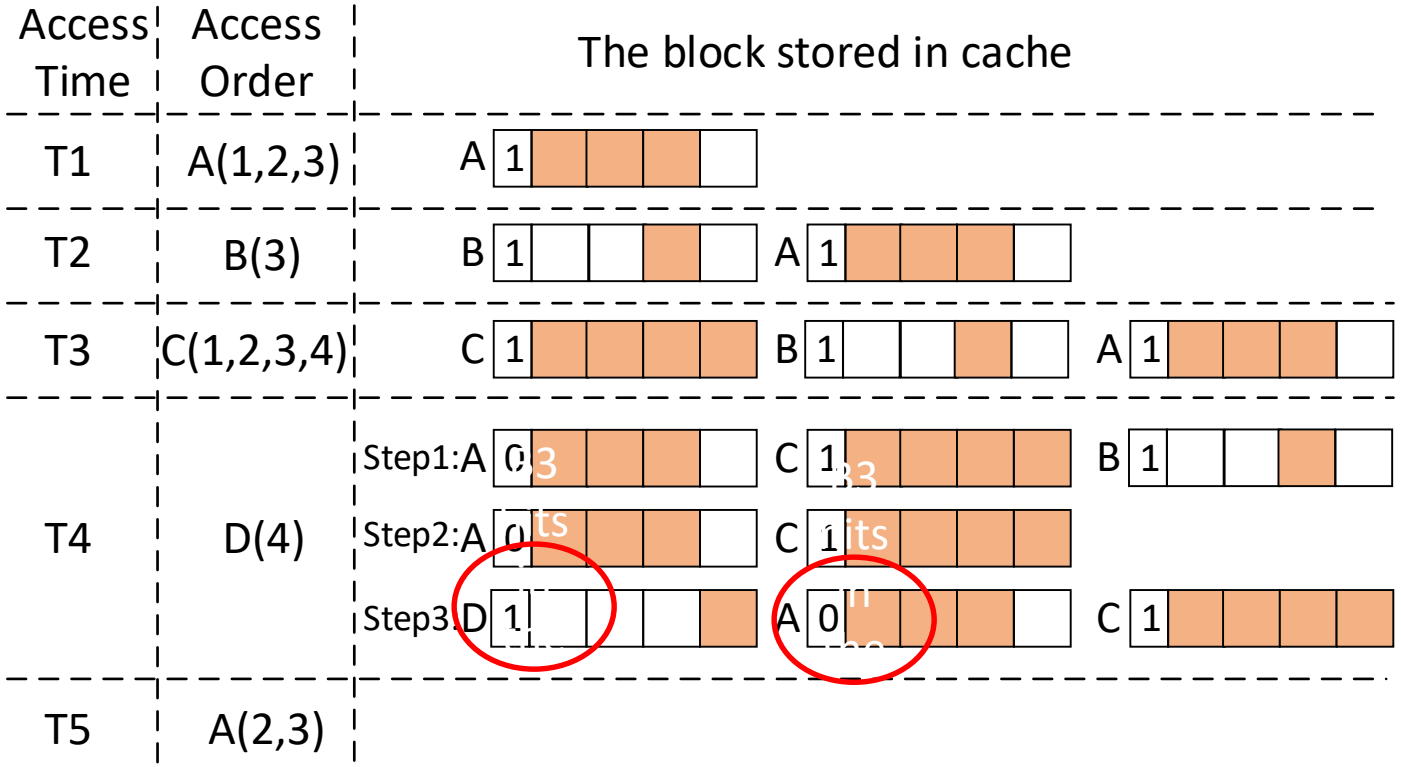
Bitmap based Hotness Identification（BHI）



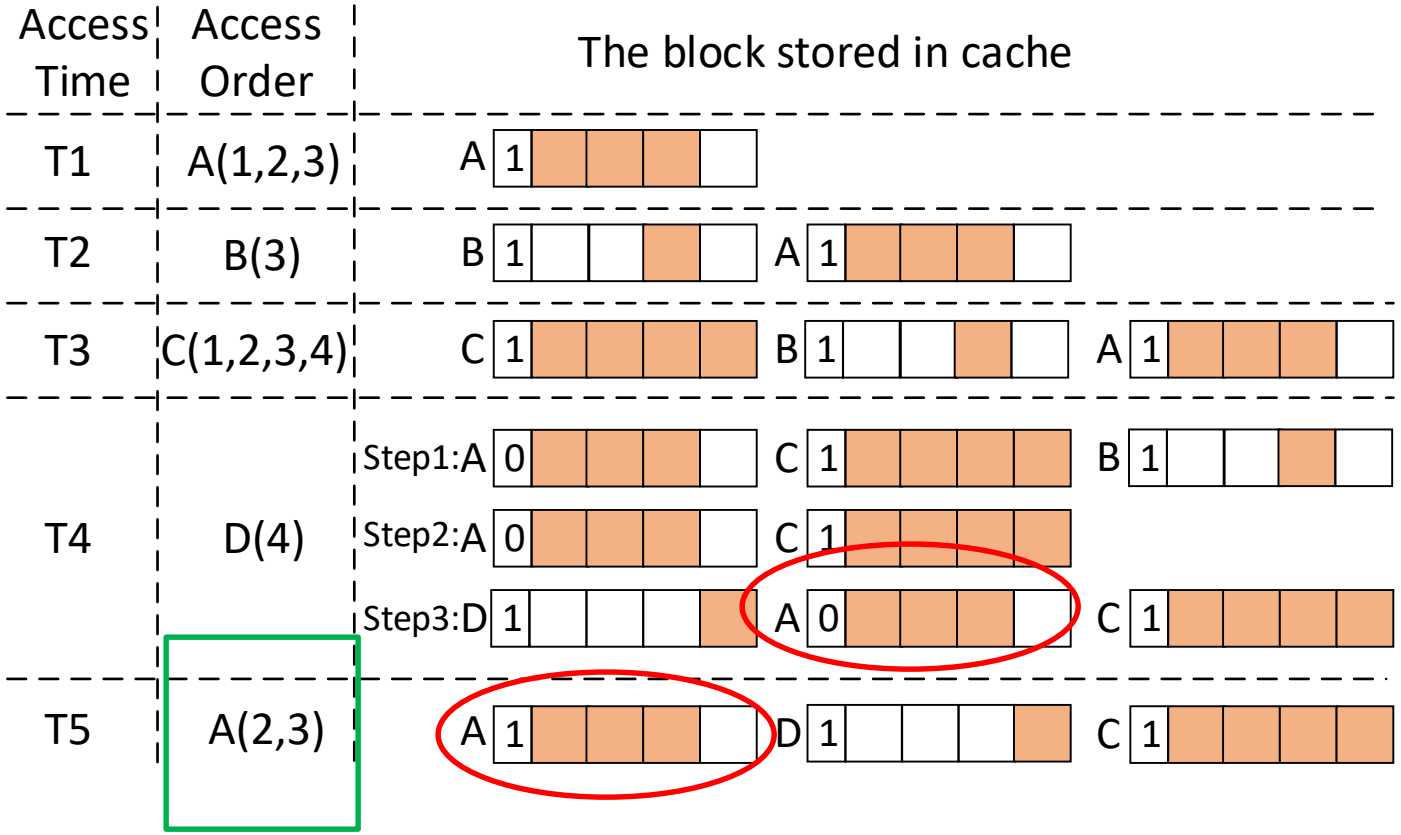
Bitmap based Hotness Identification（BHI）



Bitmap based Hotness Identification（BHI）



Bitmap based Hotness Identification（BHI）



CDAC Design

- Combining RCE and BHI together
  - CDAC first uses BHI to check if the source address in the LRU position is recognized as a cold source address
  - If it is, CDAC uses RCE to identify if it needs to be deleted
  - The source address in the LRU position needs to be constantly checked and deleted until a free block is found

The combination of BHI and RCE enables CDAC to more accurately identify the cold blocks and associated addresses to improve the cache hit ratios

Outline

- Introduction and Motivation
- Design of CDAC
- Performance Evaluation
- Conclusion

Performance Evaluation

- Baseline approaches
  - CDAC implementations: CDAC-ARC, CDAC-LRU
  - Baselines: D-ARC, D-LRU, ARC, LRU
- Parameter settings
  - Cache size: From 20% to 80% of the working set size
  - Block size: From 4KB to 64KB
  - Decline rate threshold for reference count in RCE: 50%
  - The threshold in BHI : 50%
  - The size of the smallest part of each block in BHI: 4KB
- Evaluation metric
  - Cache hit ratio

Performance Evaluation

- Trace statistics

Name	Total I/Os (GB)	Working Set(GB)	Write to read ratio	Unique Data(GB)
WebVM	54.5	2.1	3.6	23.4
Homes	67.3	5.9	31.5	44.4
Mail	1741	57.1	8.1	171.3

These traces were collected from a VM hosting the departmental websites for webmail and online course management (WebVM), a file server used by a research group (Homes), and a departmental mail server (Mail) [1].

[1] KOLLER, R., AND RANGASWAMI, R. I/o deduplication: Utilizing content similarity to improve i/o performance. In Usenix Conference on File & Storage Technologies (2010).

Performance Evaluation

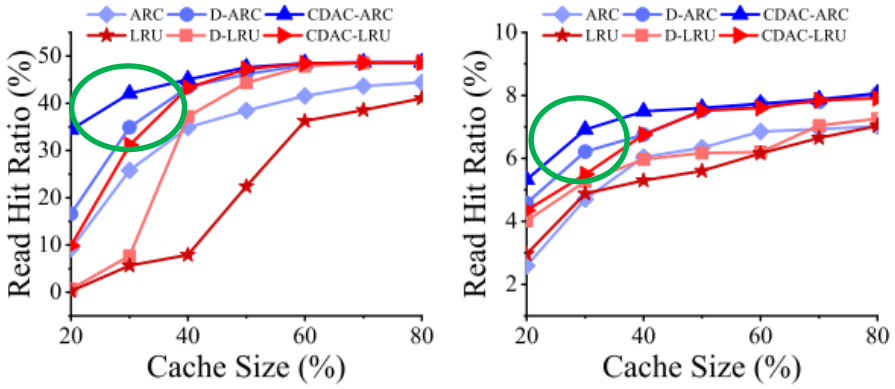
- Hit Ratios (4KB sized block)

  ——Contain only the performance results of RCE

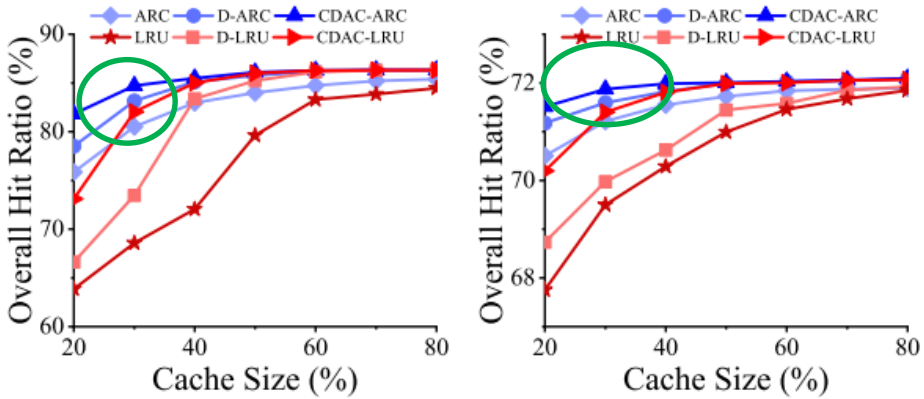
  —— the threshold for Decline rate threshold for reference count in RCE： 50%

- Analysis

  Using the reference counts that represent the intensity of the content sharing among source addresses helps preserve a lot of hot data blocks and associated source addresses， thereby increasing the cache hit rate



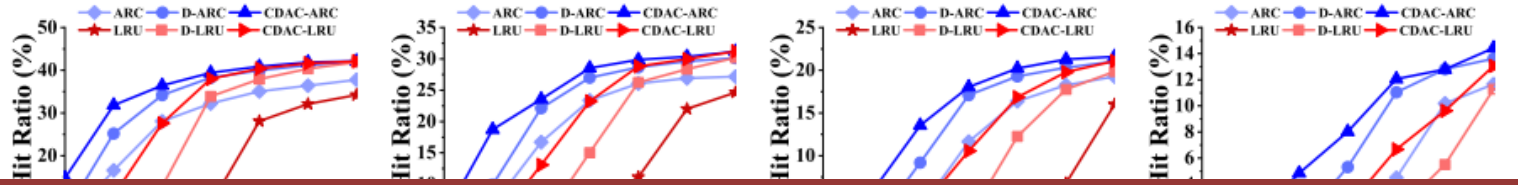
Read hit ratio with 4KB block size



Overall hit ratio with 4KB block size

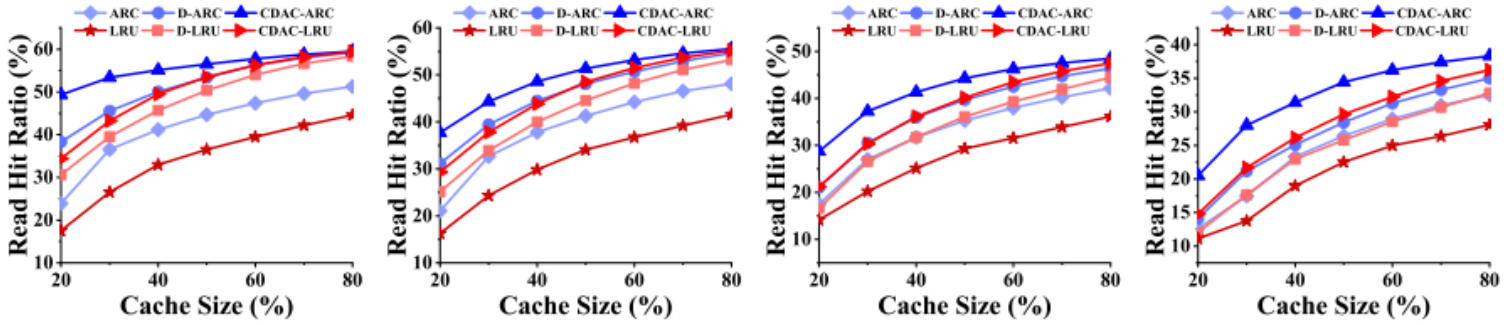
Performance Evaluation

- Read hit ratio



CDAC-LRU and CDAC-ARC outperform D-LRU and D-ARC by 1.95X on average

(A)WebVM

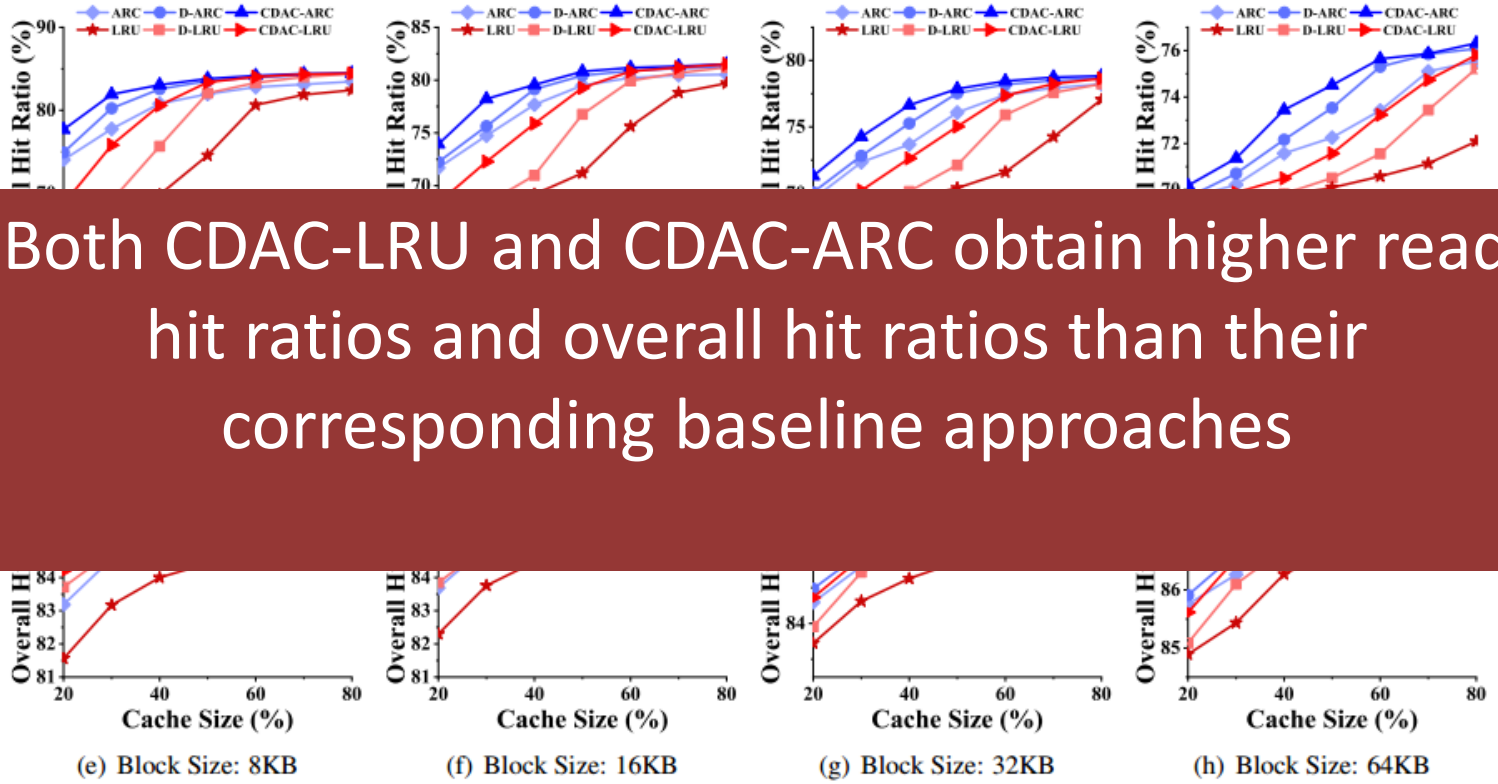


(e) Block Size: 8KB (f) Block Size: 16KB (g) Block Size: 32KB (h) Block Size: 64KB

(B)Mail

Performance Evaluation

- Overall hit ratios



Both CDAC-LRU and CDAC-ARC obtain higher read hit ratios and overall hit ratios than their corresponding baseline approaches

Performance Evaluation

- Analysis of the experimental results
  - CDAC's performance improvement in read hit ratios is greater than overall hit ratios
  - As the block size increases, the amount of cache space required for maximum improvement in CDAC increases
  - CDAC's overall performance advantages over the baselines become more pronounced as the block size increases

Outline

- Introduction and Motivation
- Design of CDAC
- Performance Evaluation
- Conclusion

Conclusion and Acknowledgement

- Analyzing the existing deduplication-aware caching algorithms
- Proposing CDAC based on CacheDedup architecture, which focuses on exploiting the blocks' content redundancy and their intensity of content sharing among source addresses
- Our extensive experimental results show that CDAC-LRU and CDAC-ARC outperform D-LRU and D-ARC by 1.95X on average under real-world traces
- Acknowledgement: We are very grateful to Professor Ming Zhao from Arizona State University for providing us with CacheDedup Prototype and many instructive comments.

Thanks!

Q&A