Austere Flash Caching with Deduplication and Compression

Qiuping Wang*, Jinhong Li*, Wen Xia# Erik Kruus^, Biplob Debnath^, Patrick P. C. Lee*

*The Chinese University of Hong Kong (CUHK)

#Harbin Institute of Technology, Shenzhen ^NEC Labs

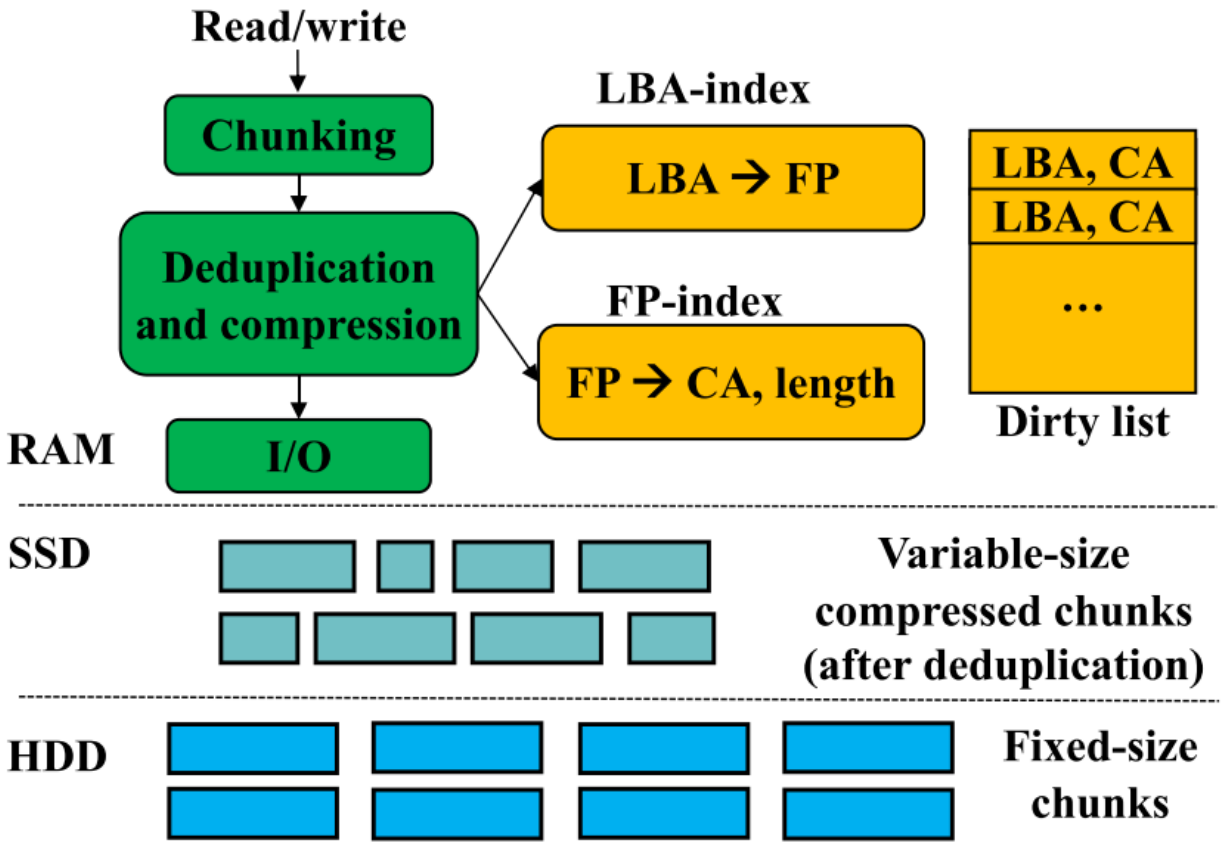
USENIX ATC'20

Background

- **Solid-state drives (SSDs) compared with hard disk drive (HDDs)**
  - Higher throughput; Lower latency; Better reliability
  - Performance, capacity and endurance are inversely proportional to price
- **Flash caching**
  - Accelerate HDD by caching frequently accessed blocks in flash
- **Deduplication & Compression**
  - Reduce storage & I/O overheads
  - Deduplication: In units of chunks (fixed-or-variable size)
  - Compression: In units of bytes

Background & Problems

- Deduplicated and compressed flash cache
  - LBA: chunk address in HDD
  - FP: chunk fingerprint
  - CA: chunk address in cache
- Memory amplification for index
  - 32KB chunk, 512GB cache with 4TB HDD
  - Conventional flash cache
    - LBA(8B)->CA(8B): 256MB
  - With deduplication and compression
    - LBA(8B)->FP(20B): 3.5GB
    - FP(20B)->CA(8B)+Length(4B):512MB
  - At least 16x memory amplification

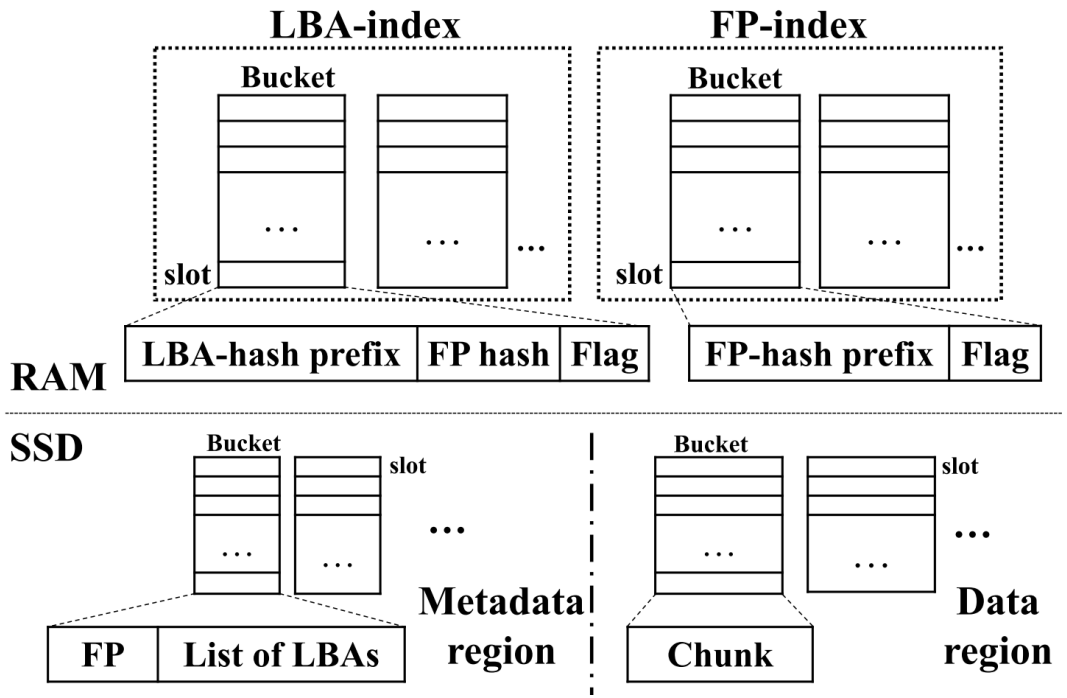


Contributions

- **AustereCache: A flash cache support deduplication and compression with austere memory-efficient management**
  - Cache data structure based on bucket grouping and prefix index
    - Hash chunks to storage locations, no overhead for address mapping
    - Prefix index in the memory and the complete index in flash
  - Fixed-size compressed data management
    - Compressing fixed-size chunks, slice and pad result to subchunks
    - No tracking for compressed length of chunks in memory
  - Bucket-based cache replacement
    - Cache replacement per bucket
    - Count-Min Sketch for low-memory reference counting

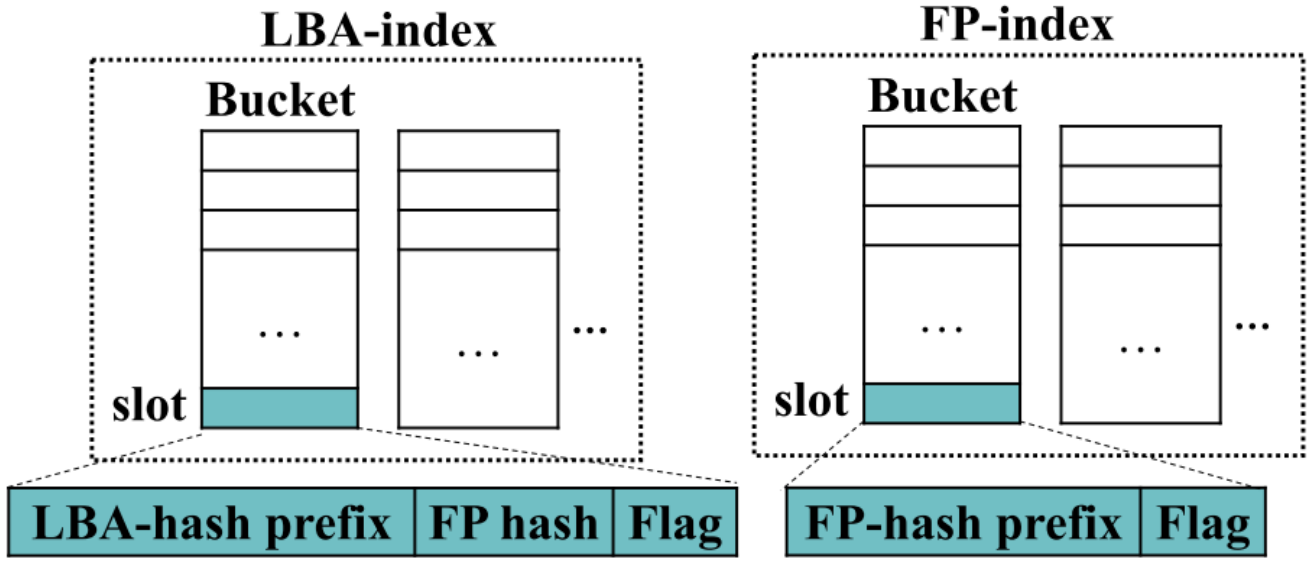
System Architecture

- Data structure
  - Use hashing to partition index and cache space
    - (RAM) LBA-index and FP-index
    - (SSD) metadata region and data region
- Layout
  - Hash entries into equal-sized buckets
  - Each bucket has fixed-number of slots



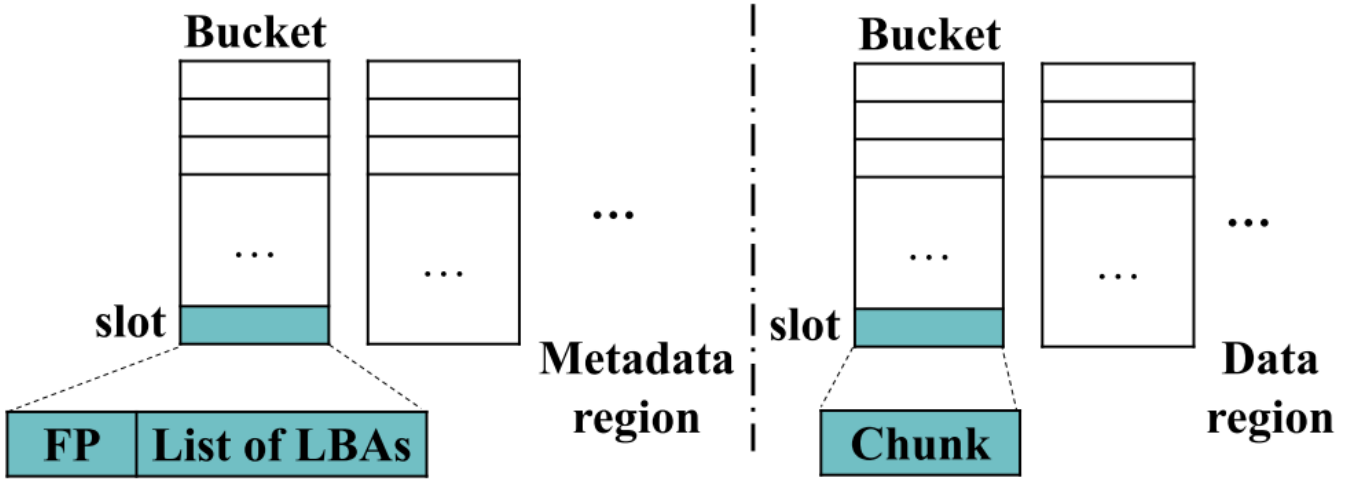
LBA-index and FP-index

- Locate
  - Buckets: hash suffixes
  - Slot: hash prefixes
- Each slot in FP-index corresponds to a storage location in flash



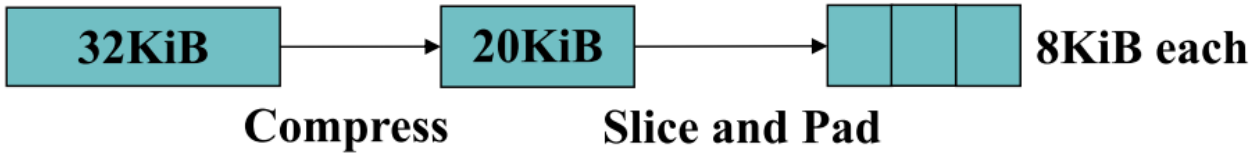
Metadata and Data Regions

- Metadata region
  - Store full FP and list of full LBAs (for validation against prefix collisions)
- Data region
  - Locate by FP-index slot
  - Store chunk content

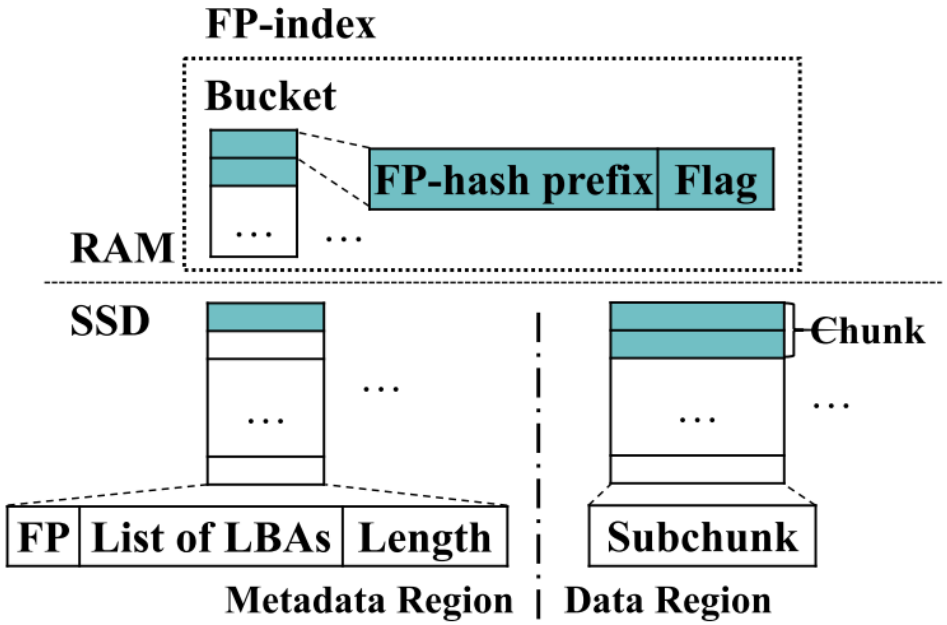


Fixed-size Compressed Data Management

- Method
  - Slicing and padding compressed chunk into fixed-size subchunks



- Layout
  - One chunk use multiple consecutive slots
  - Store chunk length in metadata region (no additional memory overhead)

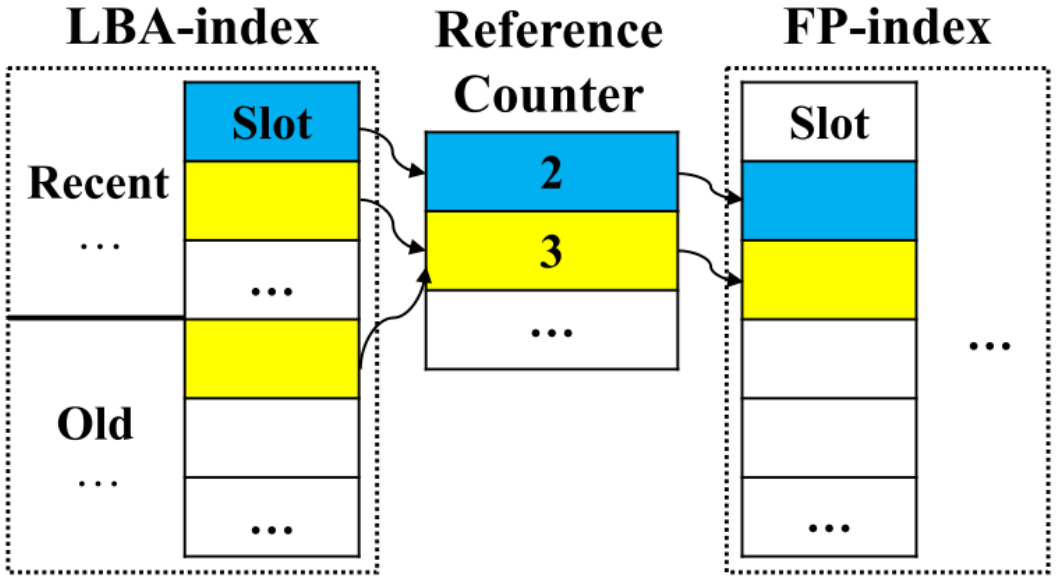


Fixed-size Compressed Data Management

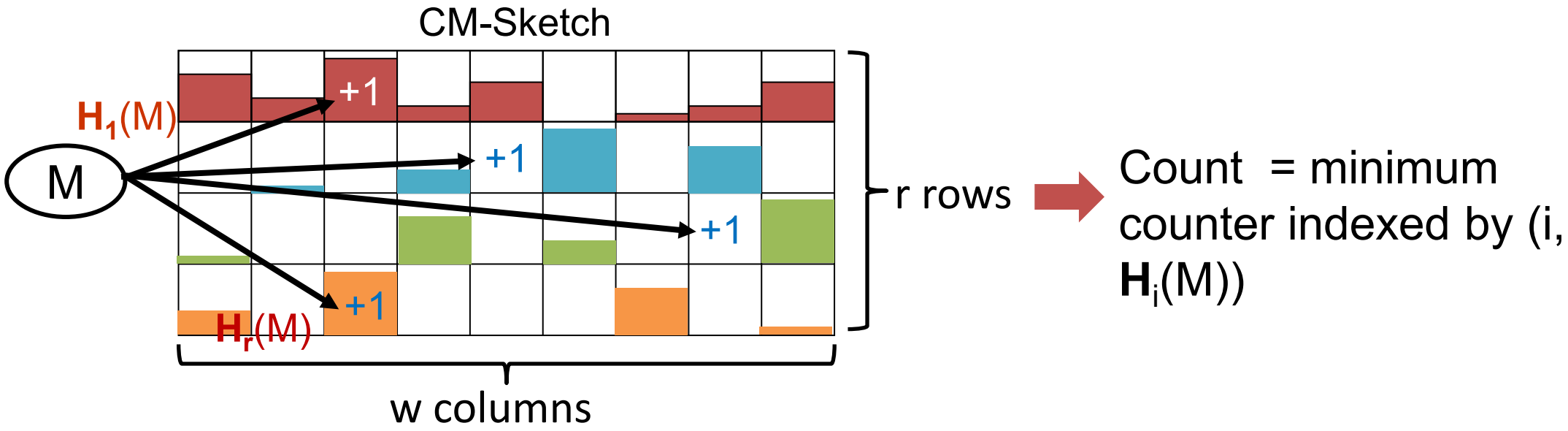
- Advantages
  - Compatible with bucketization (store each subchunk in one slot)
  - Allow pre-chunk management for cache replacement
- Disadvantages
  - The padding subchunk wastes storage space

Bucket-based Cache Replacement

- Cache replacement in each bucket independently
  - LBA-index divided into recent/old
- Combine recency and deduplication
  - LBA-index: least-recently-used policy
  - FP-index: least-referenced policy
  - Weighted reference counting based on recency in LBAs for FP-index

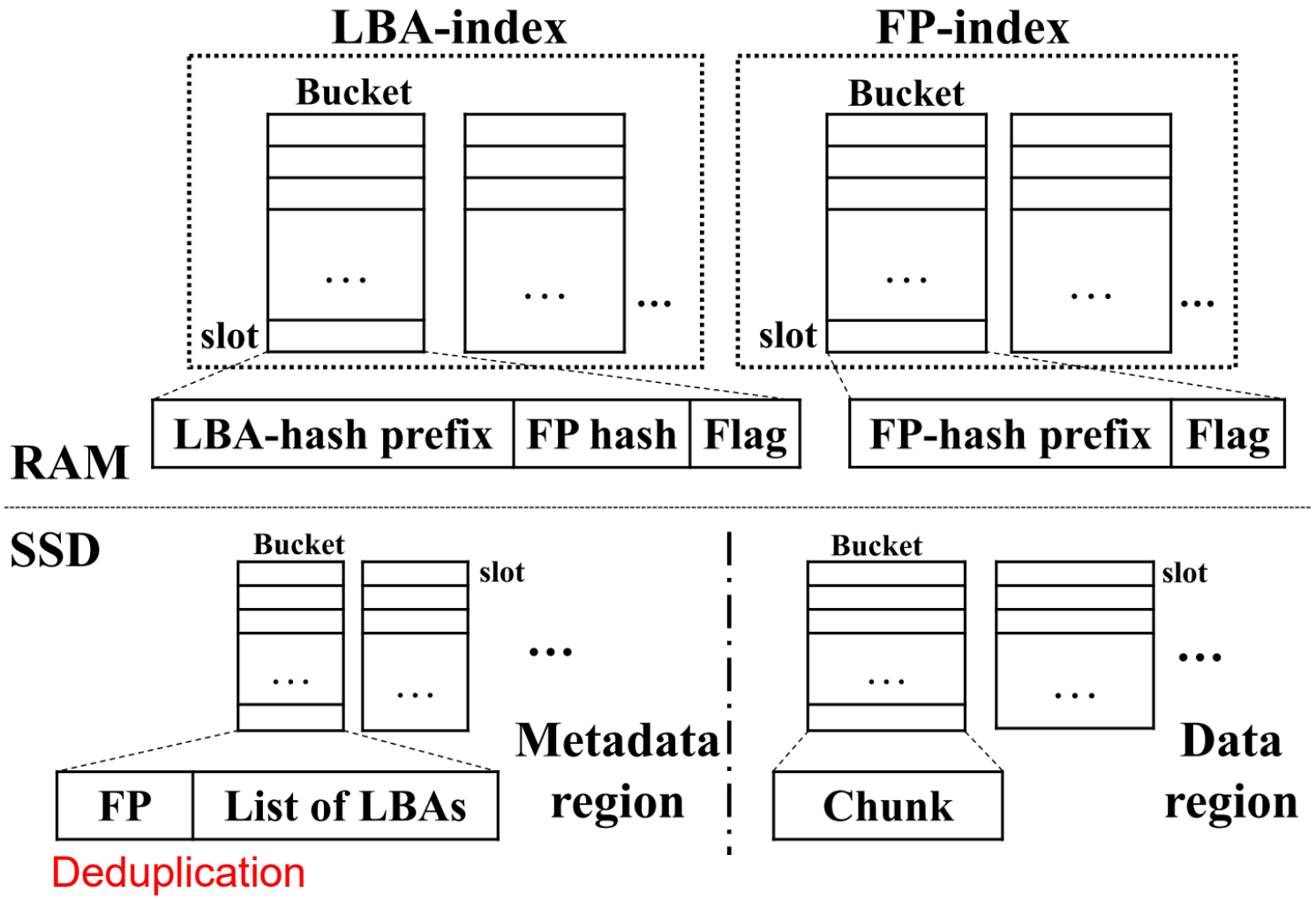


Sketch-based Reference Counting



- High memory overhead for complete reference counting
  - One counter for every FP-hash
- Count-Min Sketch[Cormode 2005]
  - Fixed memory usage with provable error bounds

Read, Write, and Deduplication Path



Evaluation

➢ Traces

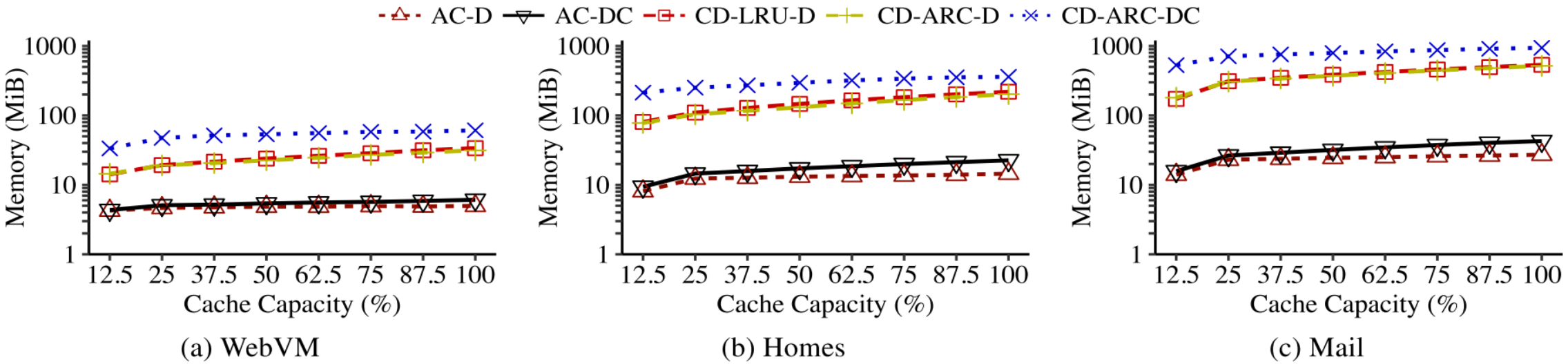
- FIU traces: WebVM, Homes, Mail
- Synthetic traces: varying I/O deduplication ratio and write-read ratio
  - I/O deduplication ratio: fraction of duplicate written chunks in all written chunks

➢ Schemes

- AustereCache:
  - AC-D Only deduplication
  - AC-DC deduplication & compression
- CacheDedup:
  - CD-LRU-D: LRU-based, only deduplication
  - CD-ARC-D: ARC-based, only deduplication
  - CD-ARC-DC: ARC-based, deduplication and compression

Evaluation

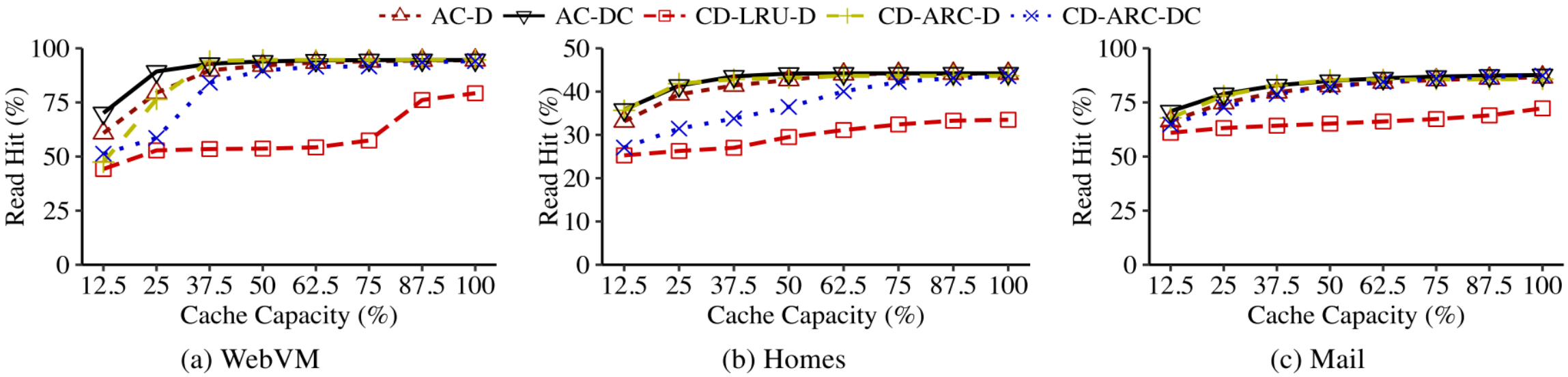
- **Memory overhead**



- AC-DC incurs 87%~97% less memory usage than CD-ARC-DC
- AC-D incurs 70%~94% less memory usage than CD-LRU-D and CD-LRU-DC

Evaluation

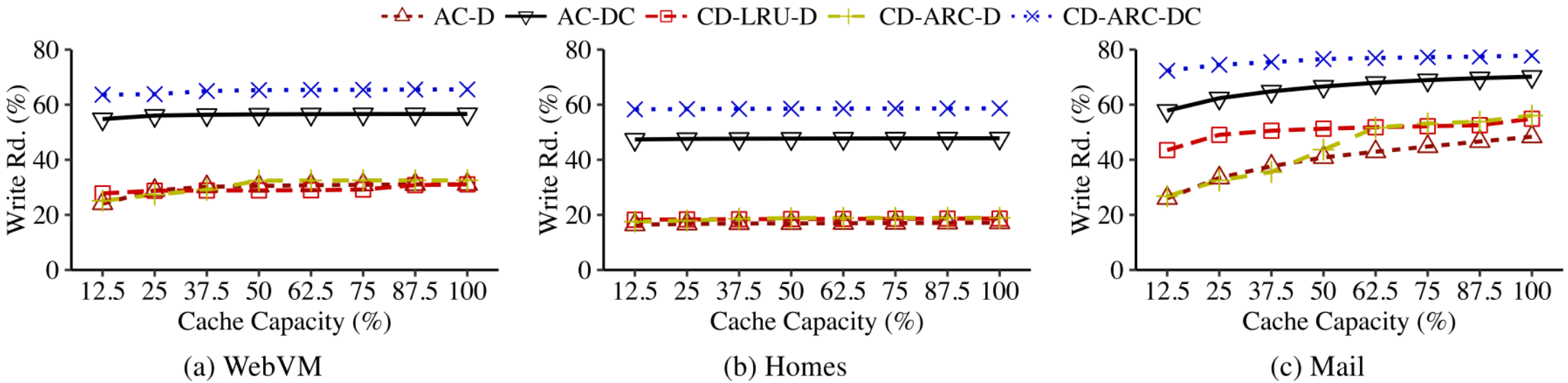
- **Read hit ratios**



- AC-D has up to 39.2% higher read hit ratio than CD-LRU-D, and similar read hit ratio as CD-ARC-D
- AC-DC has up to 30.7% higher read hit ratio than CD-ARC-DC

Evaluation

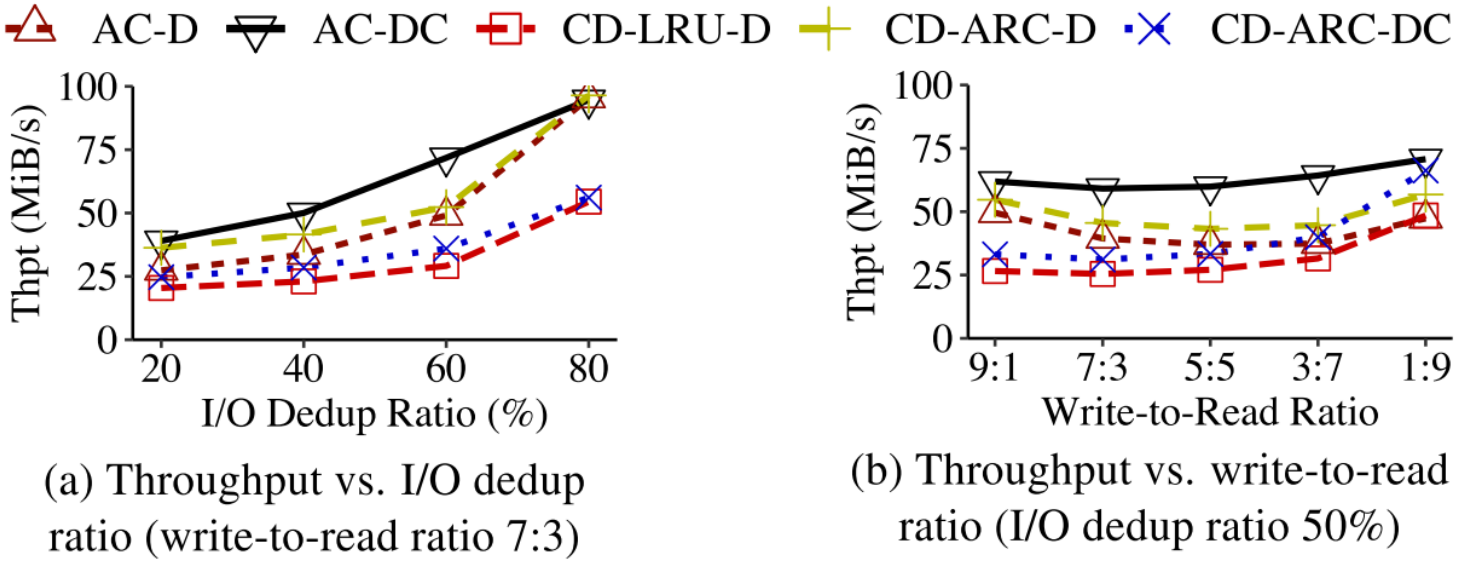
- **Write reduction ratios**



- AC-D is comparable as CD-LRU-D and CD-ARC-D
- AC-DC is slightly lower (by 7.7-14.5%) than CD-ARC-DC
  - Padding in compressed data management

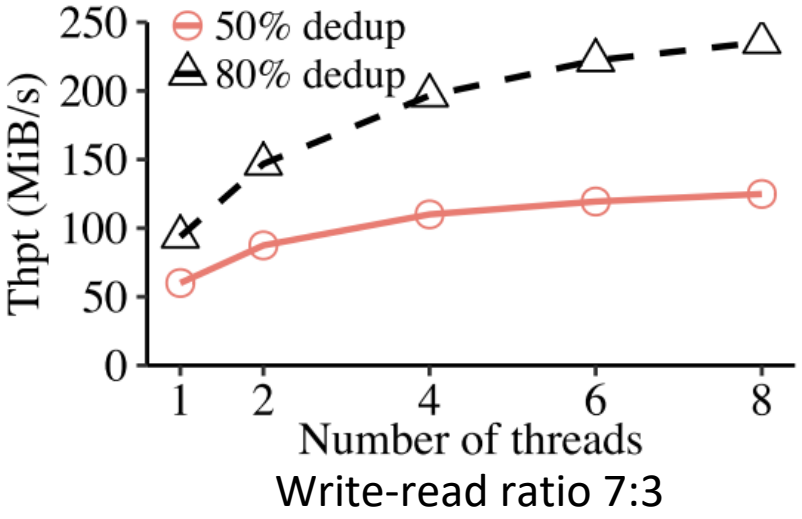
Evaluation

➢ Throughput



(a) Throughput vs. I/O dedup ratio (write-to-read ratio 7:3)

(b) Throughput vs. write-to-read ratio (I/O dedup ratio 50%)



Write-read ratio 7:3

➢ AC-DC has highest throughput

➢ AC-D has slightly lower throughput than CD-ARC-D

- AC-D needs to access the metadata region during indexing

- 50% I/O dedup ratio: 2.08X
- 80% I/O dedup ratio: 2.51X

Conclusion

- **AustereCache:** memory efficiency in deduplicated and compressed flash caching via:
  - Bucketization
  - Fixed-size compressed data management
  - Bucket-based cache replacement

Drawbacks

- The throughput is lower than HDD itself after adding the flash cache for single thread read or write.
- The size of the index structure is directly related to the hard disk's total size for caching.
- Use the slice and pad method to divide the compressed variable-length chunk into n fixed-length subchunks, which wastes part of the flash space.

Q&A