FAST'26

IPADS
INSTITUTE OF PARALLEL AND DISTRIBUTED SYSTEMS

"**Range as a Key**" is the Key! Fast and Compact Cloud Block Store Index with **RASK**

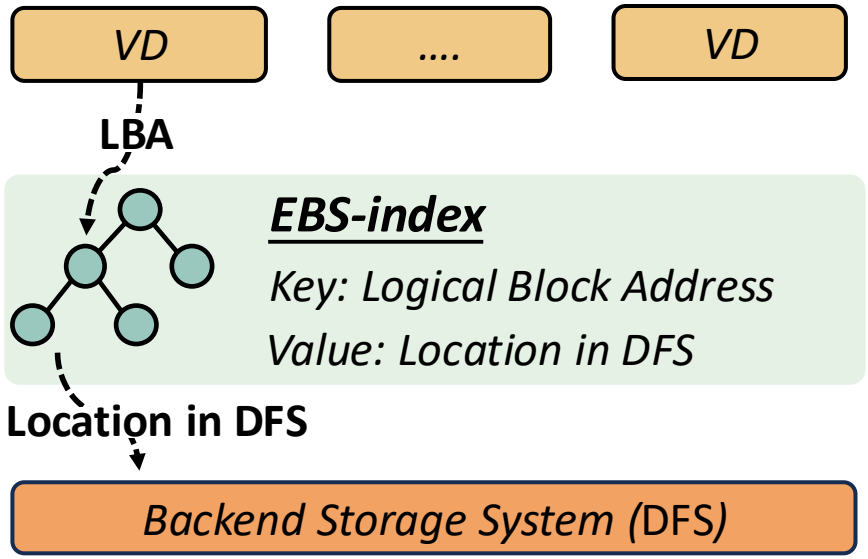
Haoru Zhao[1], Mingkai Dong[1], Erci Xu[1], Zhongyu Wang[2], Haibo Chen[1]

[1]Shanghai Jiao Tong University
[2]Alibaba Group

Cloud Block Store (EBS) & Index

- Provide virtual block (VD) device service

- Index (EBS-index) is responsible for mapping LBA in VDs to physical storage locations

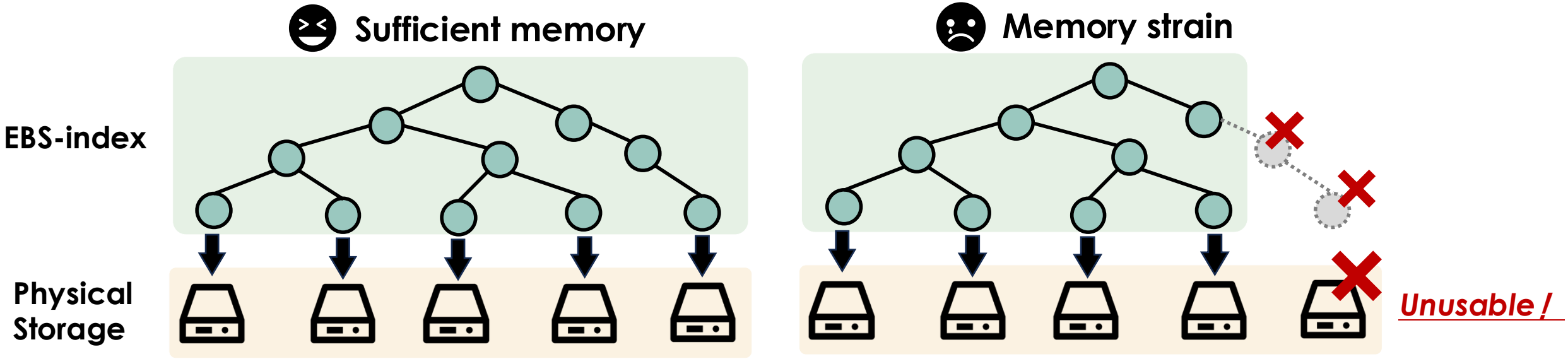


Cloud block store architecture

Index Strains Cloud Block Store Memory

Memory strains in Cloud Block Store

- The Consequences:
  - some physical storage cannot be indexed and thus unusable
  - **~10%** of clusters risk wasting **~35%** of storage



Index Strains Cloud Block Store Memory

Memory strains in Cloud Block Store

- The Consequences:
  - some physical storage cannot be indexed and thus unusable
  - **~10%** of clusters risk wasting **~35%** of storage

EBS-index strains the memory resources

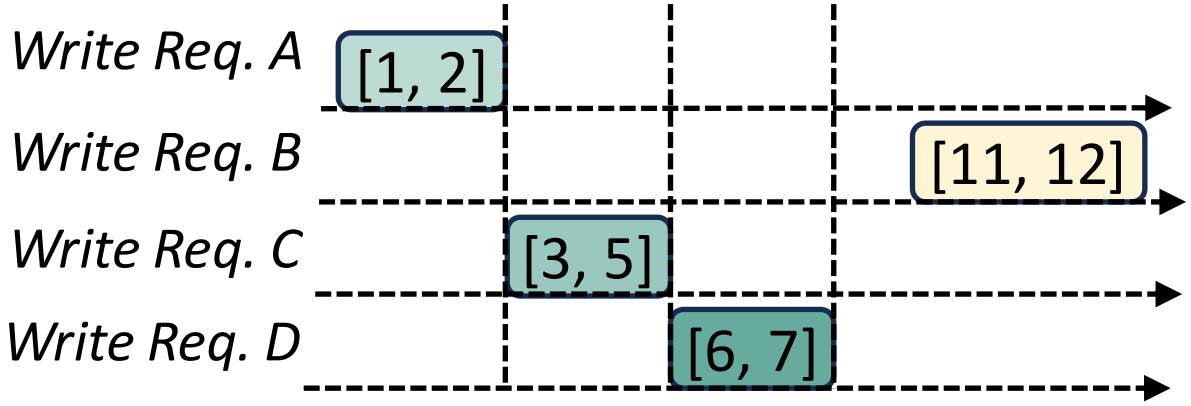
- Alibaba cloud's statistics: indexes consumes **57.1%** of total memory

A new memory-friendly index is needed for Cloud Block Store.

Observation: Consecutive Write Pattern

Key Observation: Prevalent Consecutive Writes (CW)

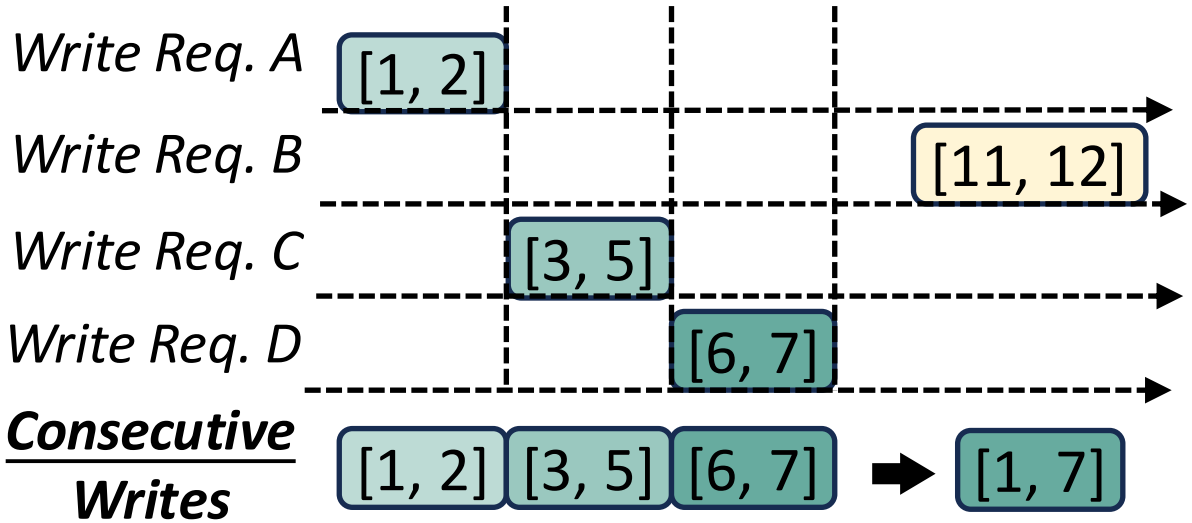
- Individual write requests that are **temporally close** can be **spatially consecutive**



Observation: Consecutive Write Pattern

Key Observation: Prevalent Consecutive Writes (CW)

- Individual write requests that are **temporally close** can be **spatially consecutive**

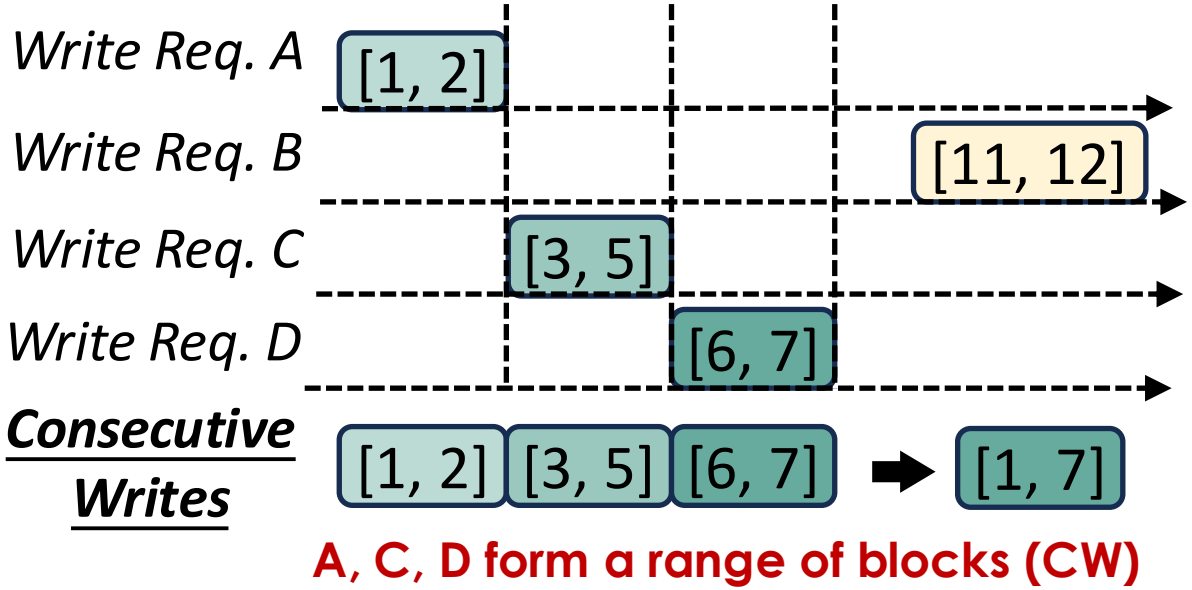


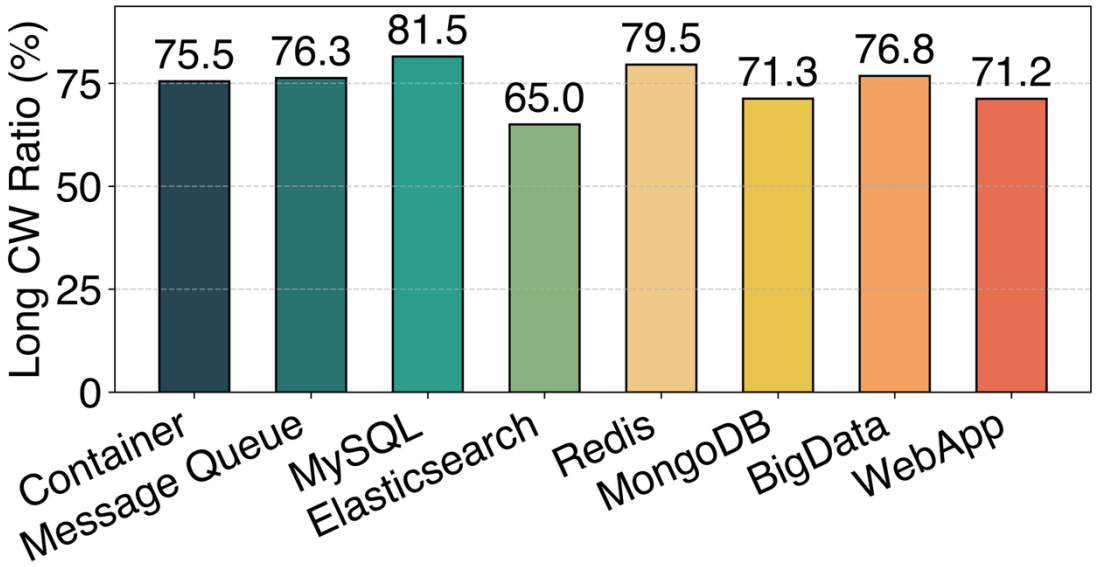
A, C, D form a range of blocks (CW)

Observation: Consecutive Write Pattern

Key Observation: Prevalent Consecutive Writes (CW)

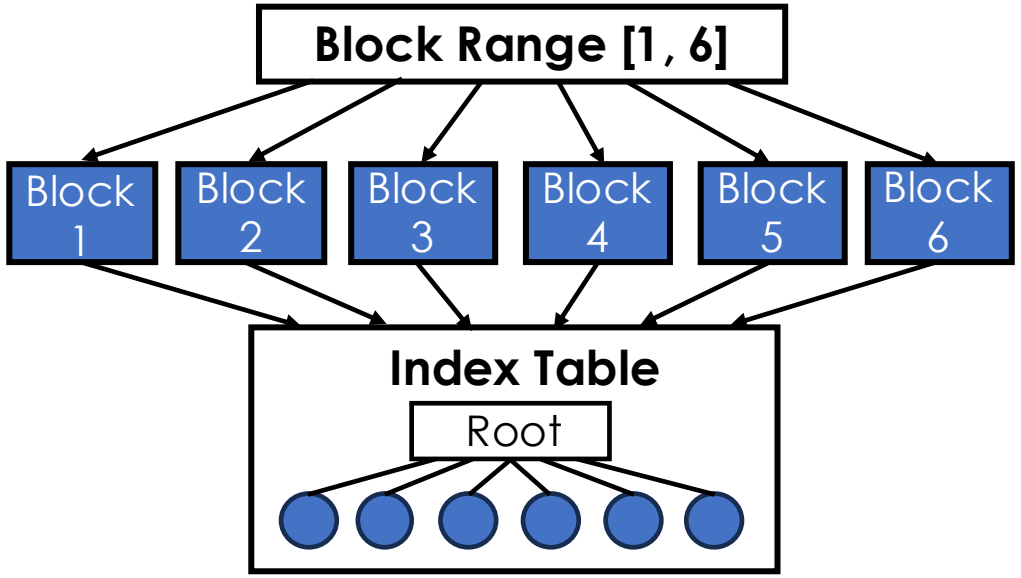
- Individual write requests that are **temporally close** can be **spatially consecutive**
- Widespread in Alibaba Cloud EBS traces: **65.0–81.5%** across 8 workloads
- Range writes stem from common root causes





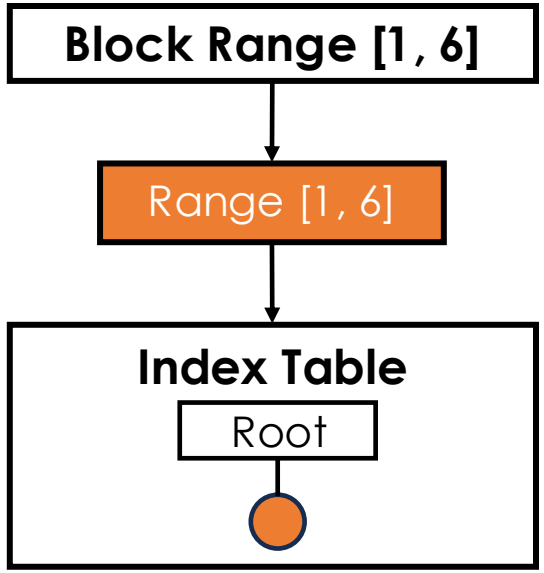
Call for "Range as a Key"

Previous (Block-based)



- High memory footprint
- Multiple index updates for range writes
- Slower queries due index scale

Range as a Key

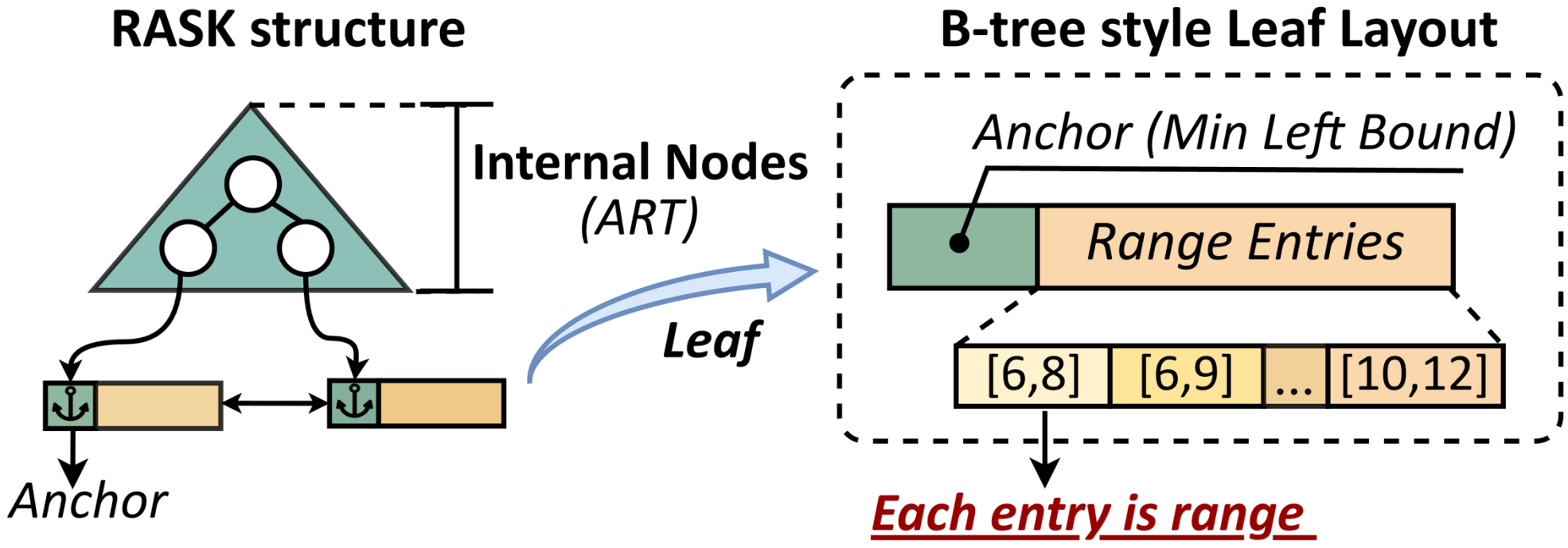


- Reduced memory footprint
  - (e.g., 58.4-91.1% for EBS)
- Single index update for range writes
- Faster queries due smaller index

Directly index ranges instead of individual blocks

Range-AS-a-Key Tree Index (RASK)

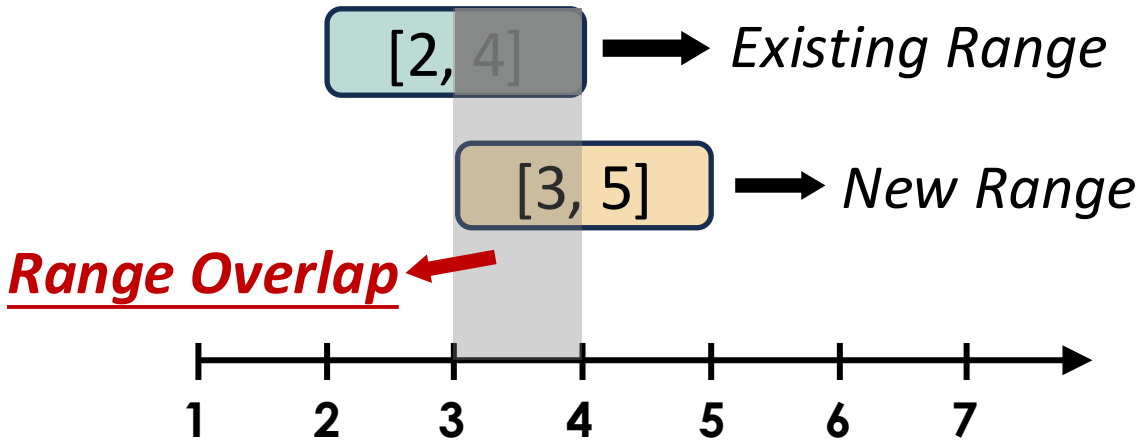
- Trie-format internal nodes (ART) and B-tree style leaves
  - Each entry represents ranges rather than individual objects



Challenge 1: Range Overlap

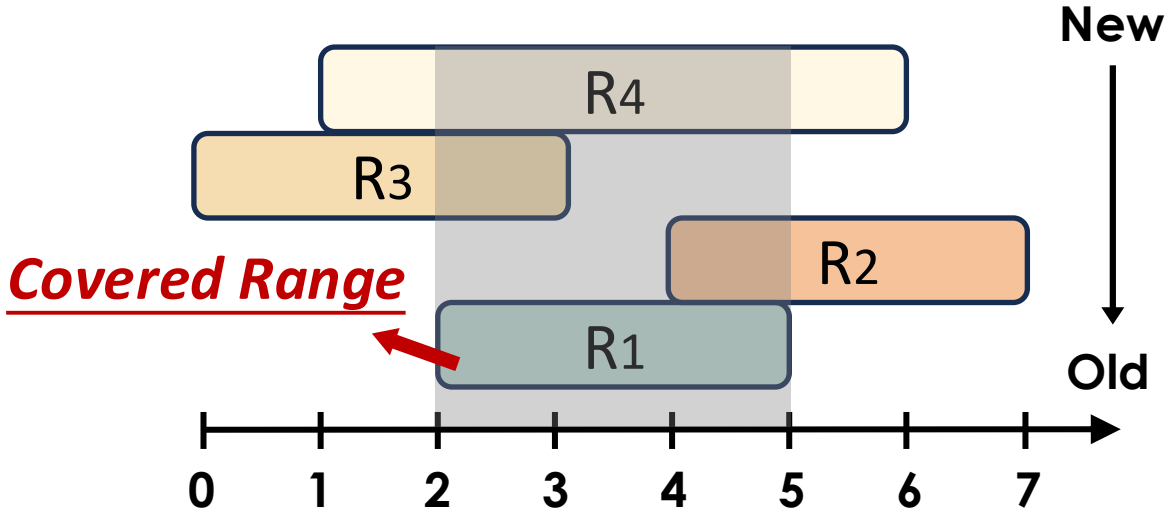
Problem

- New Range intersects existing ones
- E.g., [3, 5] overlaps [2, 4]



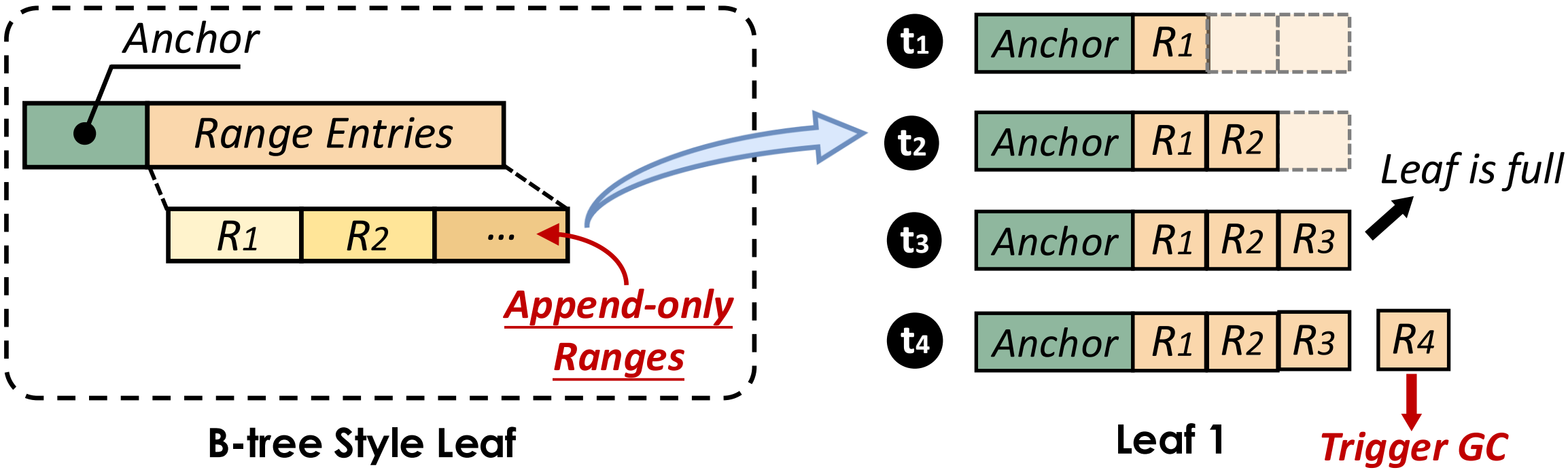
Impact

- Degrade read performance
- Memory waste
- Covered Range Removal Overhead



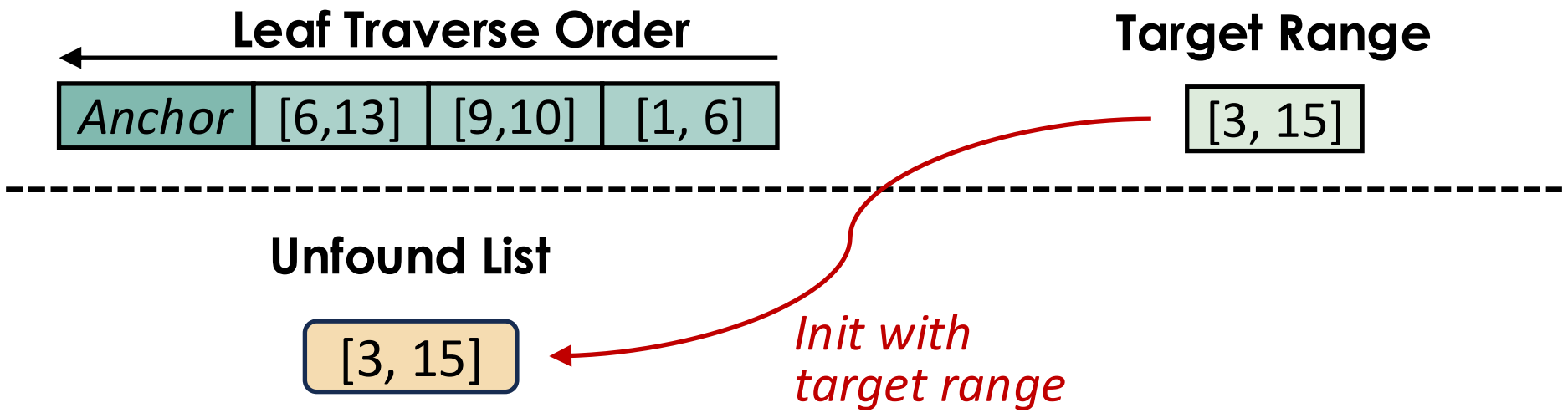
Log-structured Leaf

- Globally ordered Leaves, append-only updates within leaf
  - batch covered ranges removal by GC, reducing its impact on writes



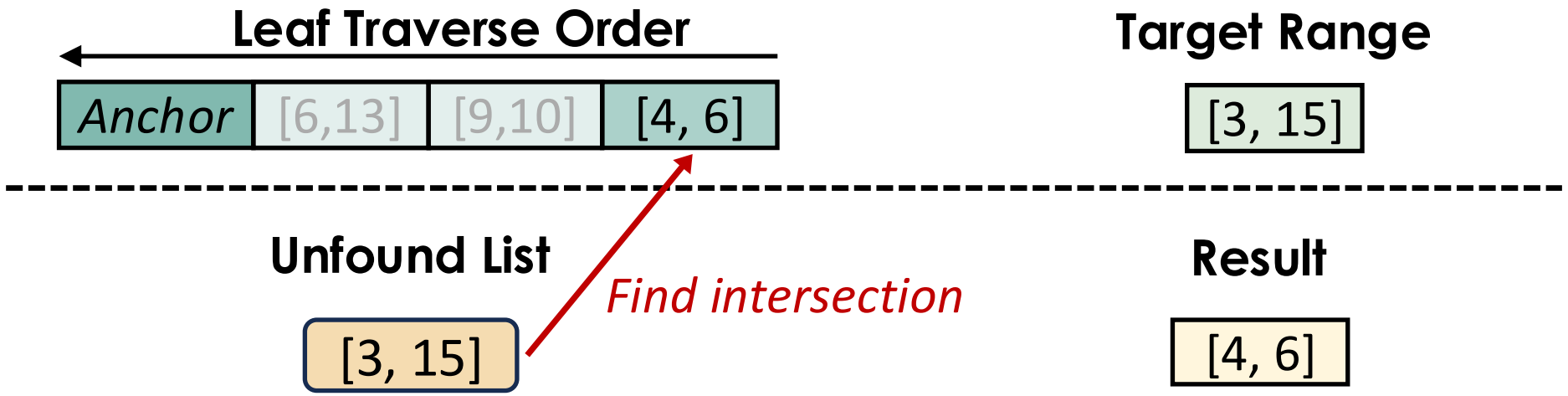
Ablation-based search

- Traverse the leaf in reverse order
- Collect (unfound parts of target range) ∩ (leaf's range)
- **Unfound List**: Track target range's unfound parts



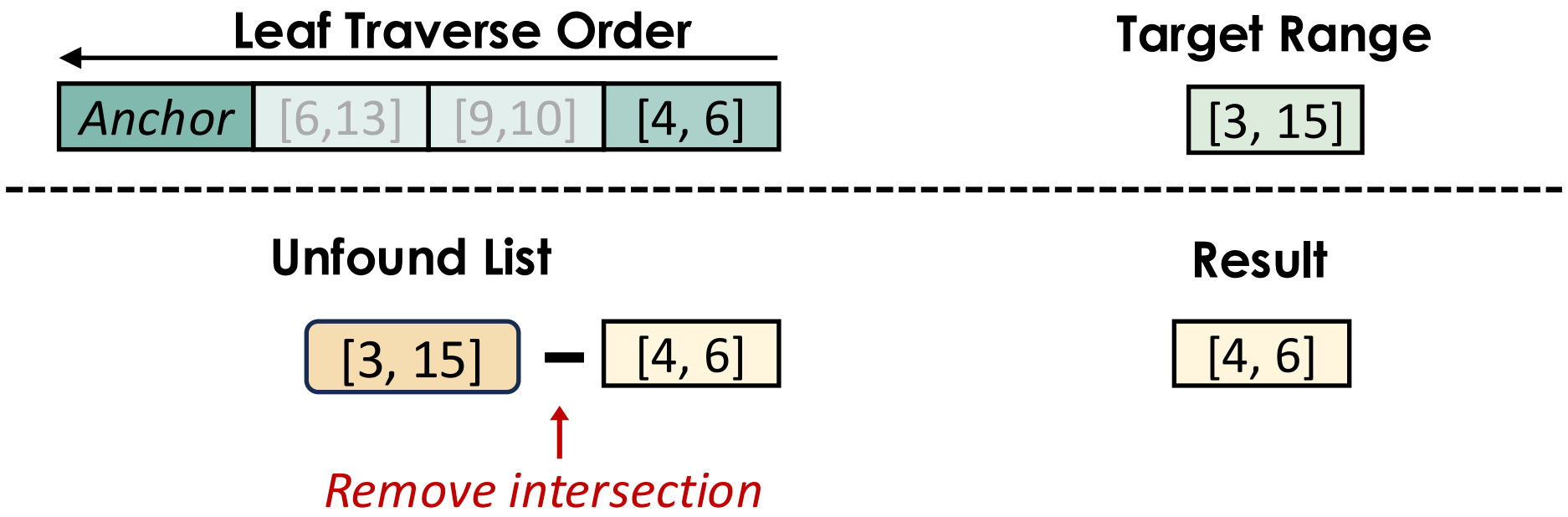
Ablation-based search

- Traverse the leaf in reverse order
- Collect (unfound parts of target range) ∩ (leaf's range)
- **Unfound List**: Track target range's unfound parts



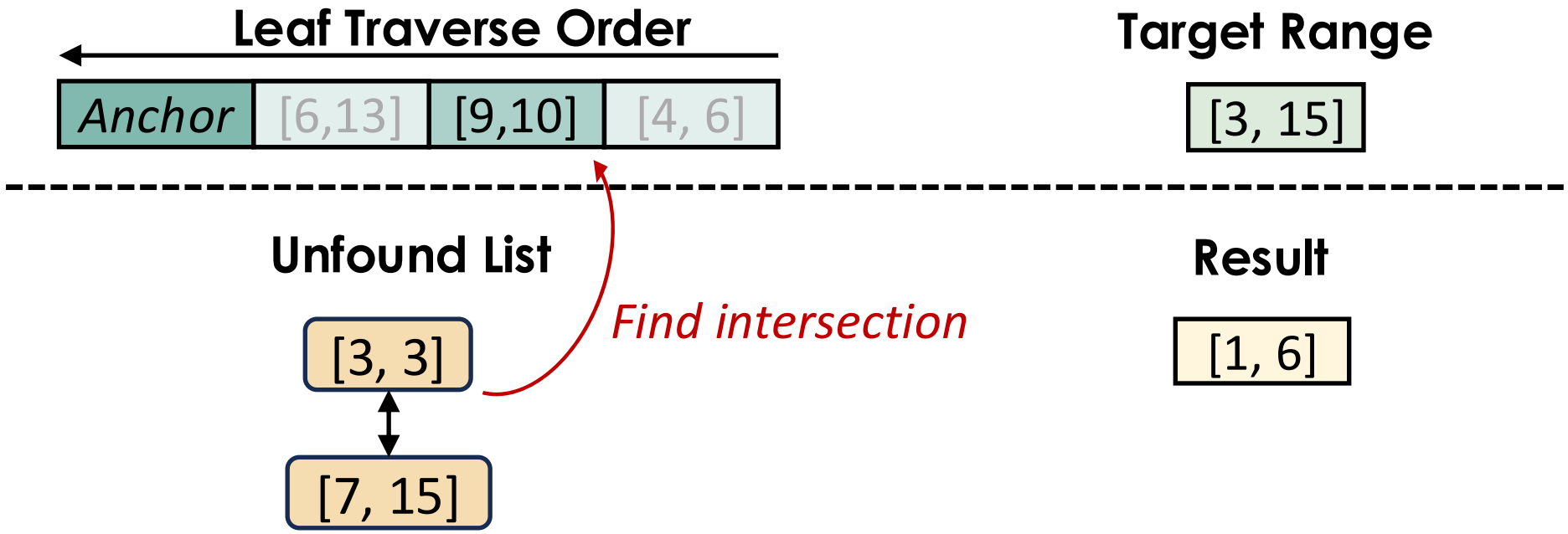
Ablation-based search

- Traverse the leaf in reverse order
- Collect (unfound parts of target range) ∩ (leaf's range)
- **Unfound List**: Track target range's unfound parts



Ablation-based search

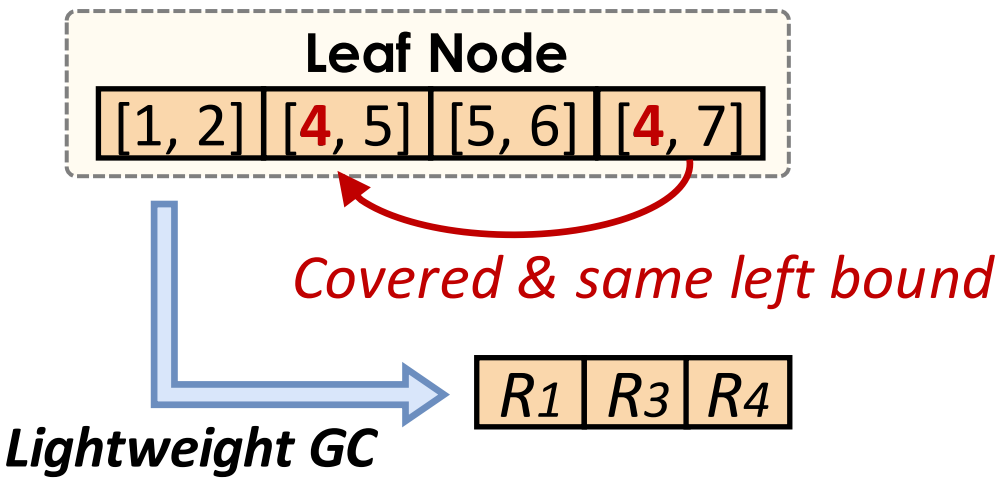
- Traverse the leaf in reverse order
- Collect (unfound parts of target range) ∩ (leaf's range)
- **Unfound List**: Track target range's unfound parts



Two-stage GC

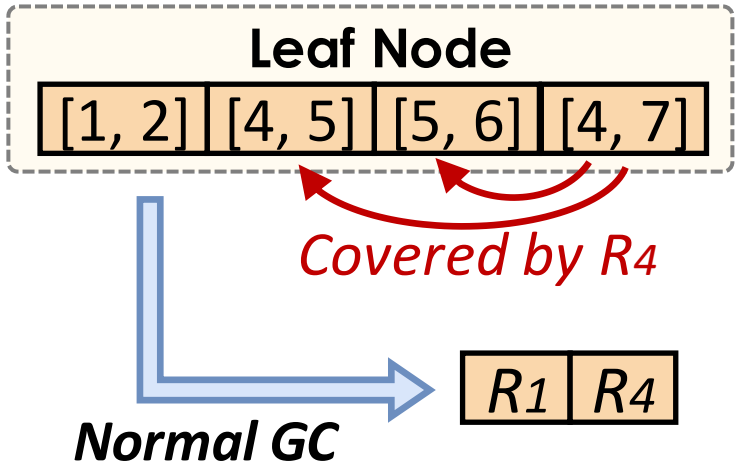
Lightweight GC

- quickly removes some common covered ranges
  - reduce write blocking time



Normal GC

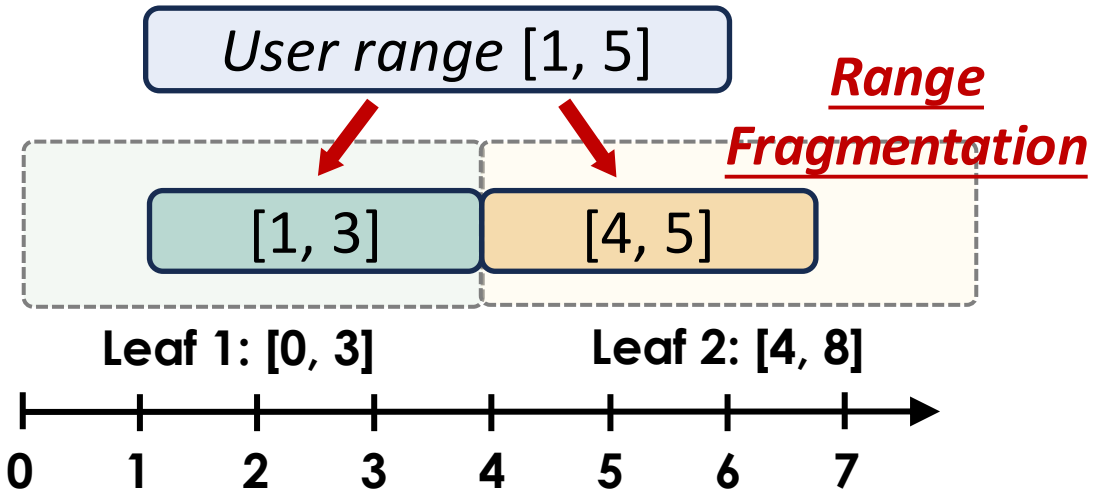
- thoroughly removes all covered ranges



Challenge 2: Range Fragmentation

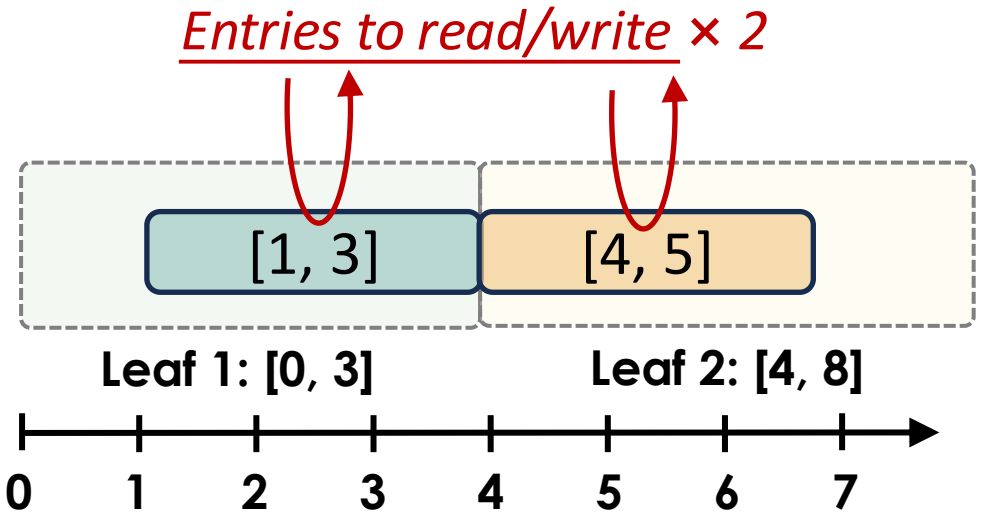
Problem

- User ranges spanning multiple leaves must be divided
- Range [1, 6] spans leaf 1 & 2



Impact

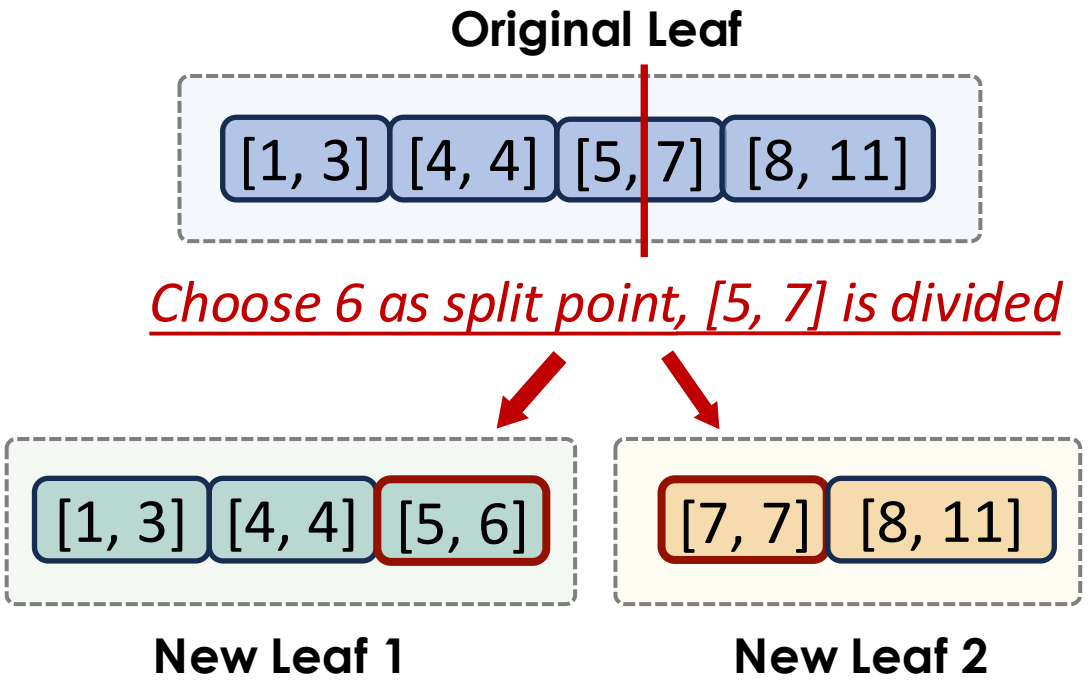
- Higher memory cost
- Increased read/write overhead



Range-conscious Split

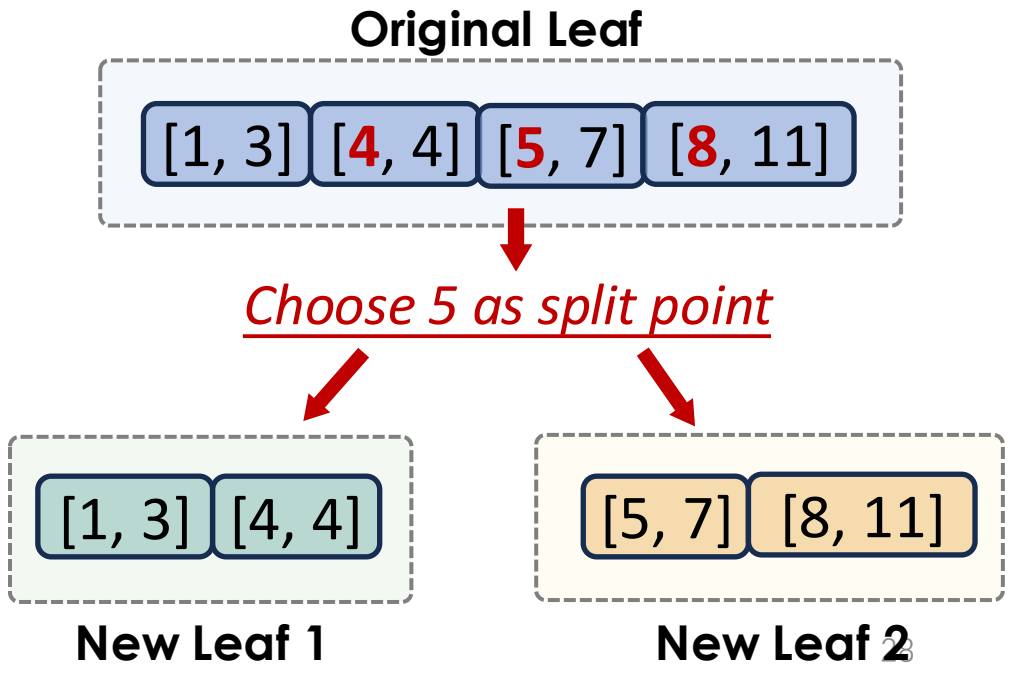
Point-based index Splitting (e.g., B+ tree)

- Split point is median key
  - agnostic to leaf's range



RASK

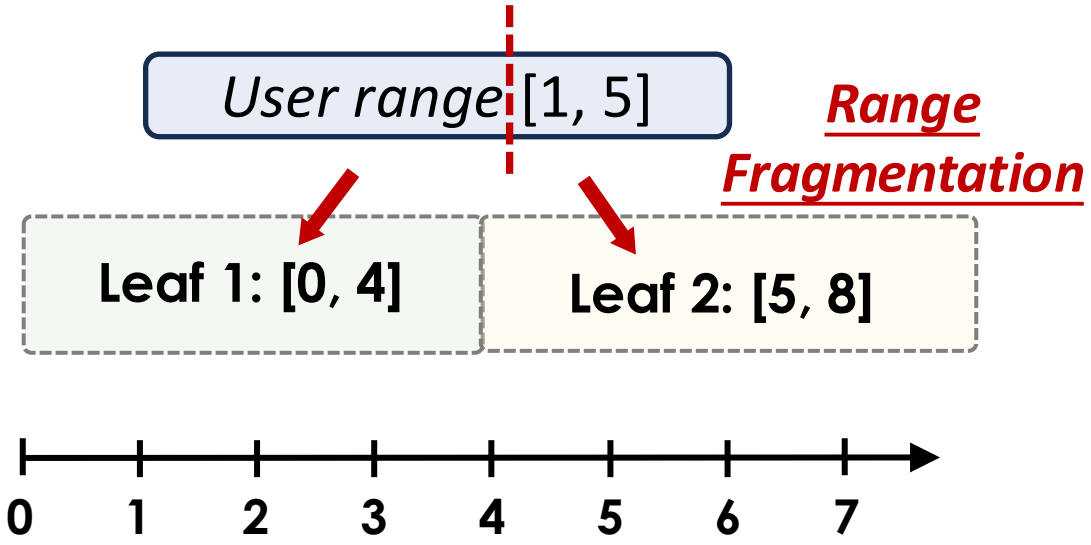
- Choose split point from ranges' boundaries
- Balance entry count in new leaves



Workload-aware Merge & Resplit

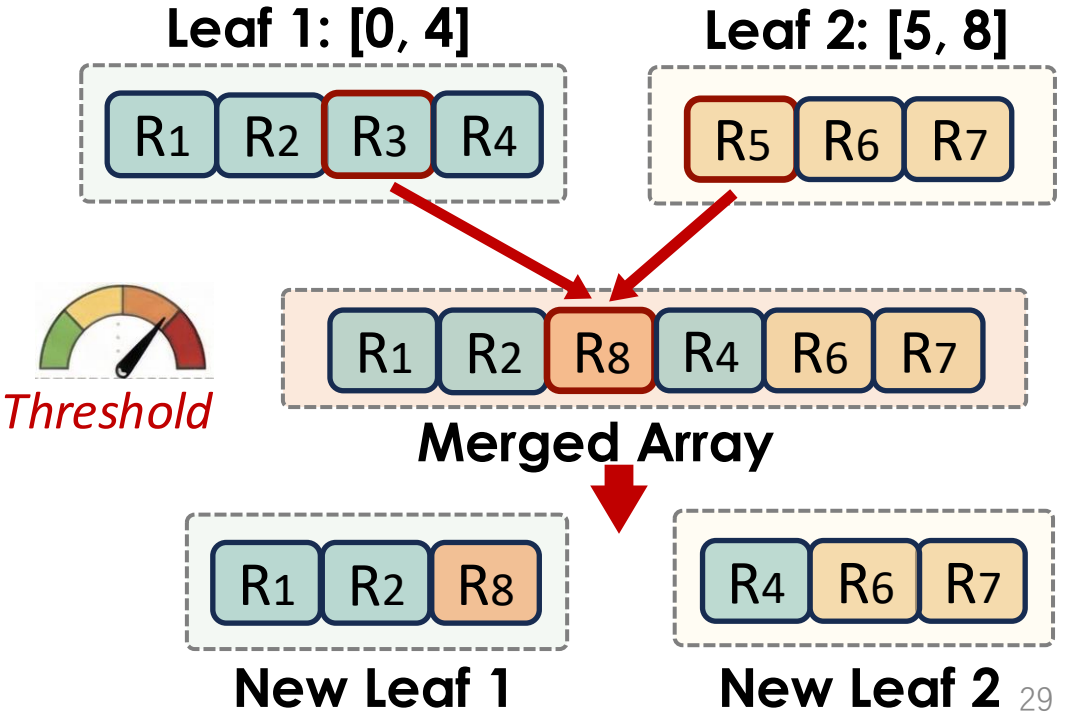
Problem

- New user-written ranges do not align with existing leaves



RASK

- Adjusts leaves' range space according to fragmentation severity



More Details

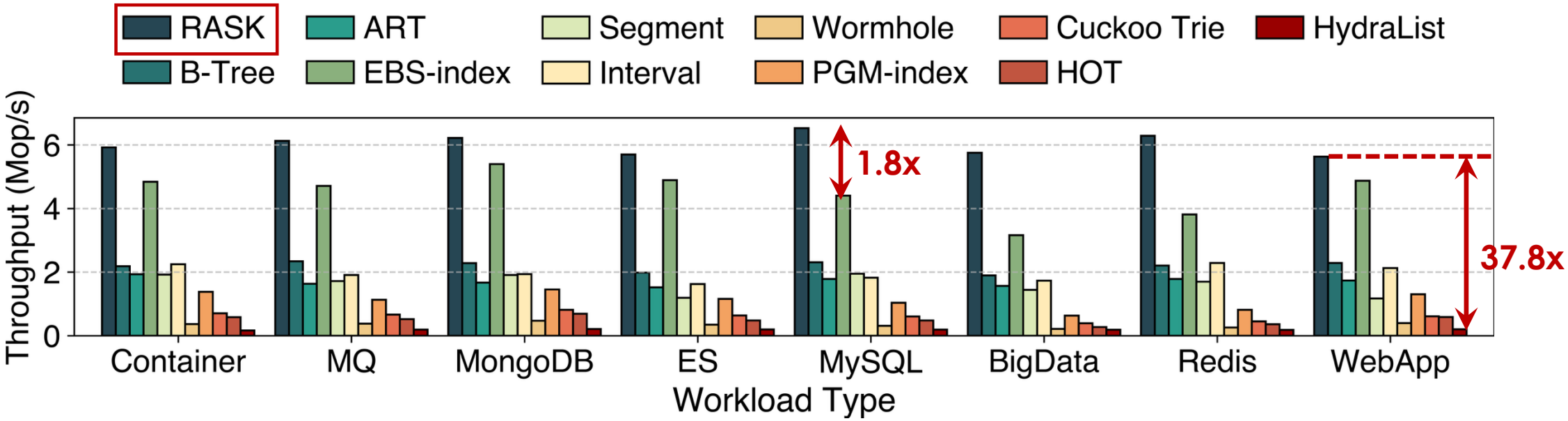
- Two-stage GC procedure
- Split point selection strategy
- Read/Write Interfaces
- Concurrency Support
- …

Evaluation

- Setup
  - a server with 1 Platinum 8369B CPU (2.70 GHz, 24 cores) and 96GB DRAM
- Benchmarks
  - Full Dataset: EBS traces of 1.8 k VDs at Alibaba Cloud
  - Sampled Dataset: 100 VDs from Full Dataset
  - Traces from Meta, Google, and Tencent
- 10 Baselines

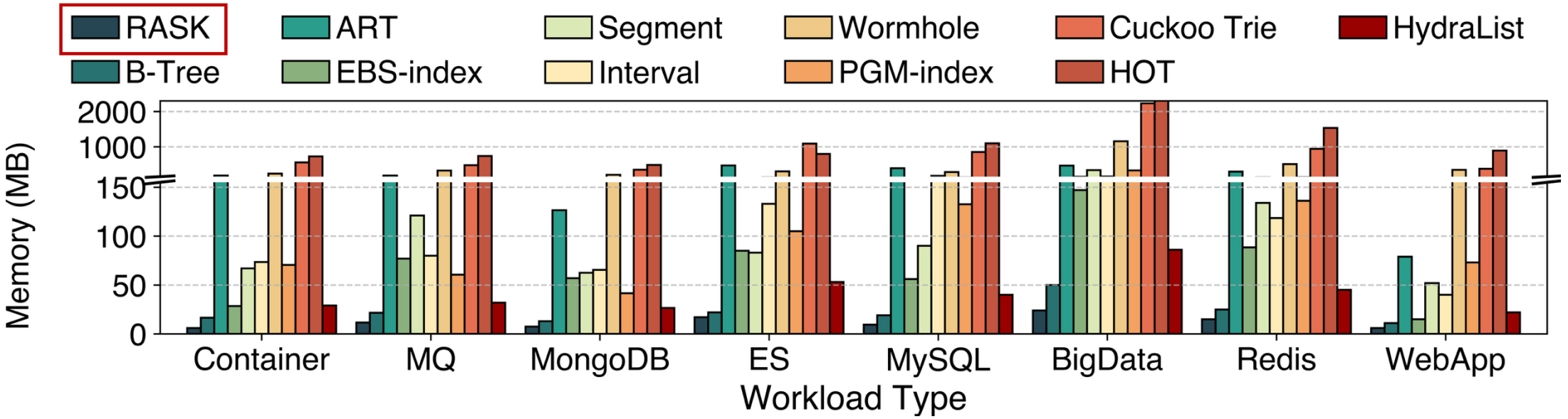
Cuckoo Trie (SOSP '21)	HydraList (VLDB '20)	Wormhole (EuroSys '19)	HOT (SIGMOD '18)
PGM-index (VLDB '20)	STX B-tree	ART (ICDE '13)	Segment tree
Interval Tree	Alibaba Cloud's EBS-index		

Throughput Comparison



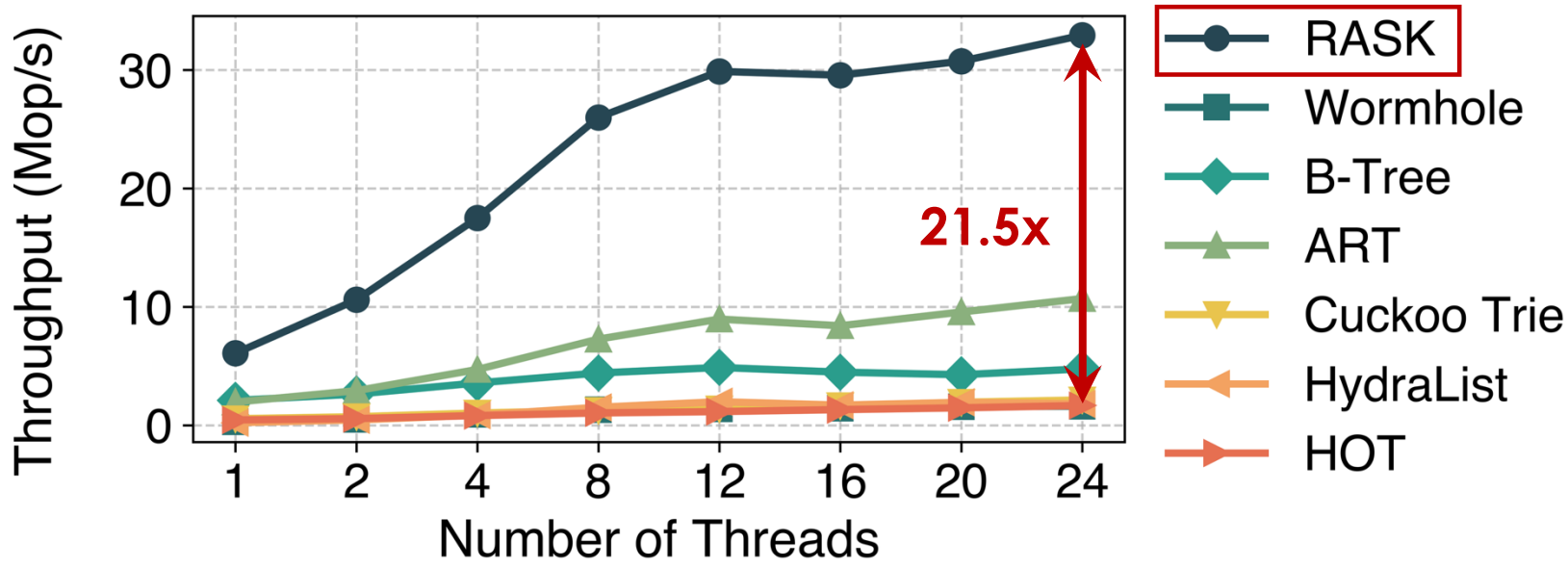
- Compared With EBS-index, RASK achieves up to **1.8x** improvement
- Compared With 9 ordered index, RASK achieves **2.5-37.8x** improvement

Memory Footprint Comparison



- Compared With EBS-index, RASK saves up to **80.1%** of memory
- Compared With 9 ordered index, RASK saves **45.3-98.9%** of memory

Multi-thread Performance



- RASK's throughput reaches **3.1-21.5x** that of baselines at 24 threads

Conclusion

- In cloud block store, writes typically target **a range of blocks**
- Key Insight: For cloud block store, the index should use **range as a key**
- We propose **RASK**, a high-performance and memory-efficient index
  - Natively support range-as-a-key
  - Address challenges of range overlap and range fragmentation
- RASK code: https://ipads.se.sjtu.edu.cn:1312/opensource/rask-index
- EBS traces: https://tianchi.aliyun.com/dataset/218875

Thanks & QA