DiSC lab

BOSTON UNIVERSITY

Benchmarking, Analyzing, and Optimizing Write Amplification of Partial Compaction in Rocksdb

Ran Wei* Zichen Zhu* Andrew Kryczka

Jay Zhuang Manos Athanassoulis

Log-Structured Merge-tree (LSM-tree)

Widely adopted because it offers fast ingestion rate and competitive read latency

NoSQL

levelDB RocksDB amazon DynamoDB

WT APACHE HBASE cassandra

Relational

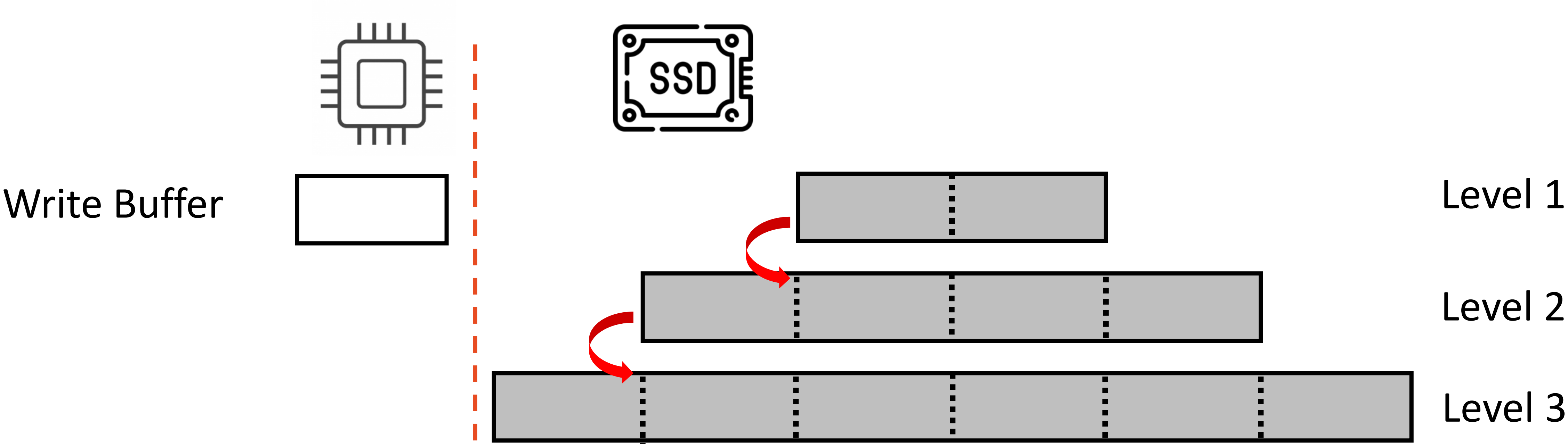
SQLite

Time-series

influxdb

APACHE IoTDB

Log-Structured Merge-Trees (LSM-Trees)



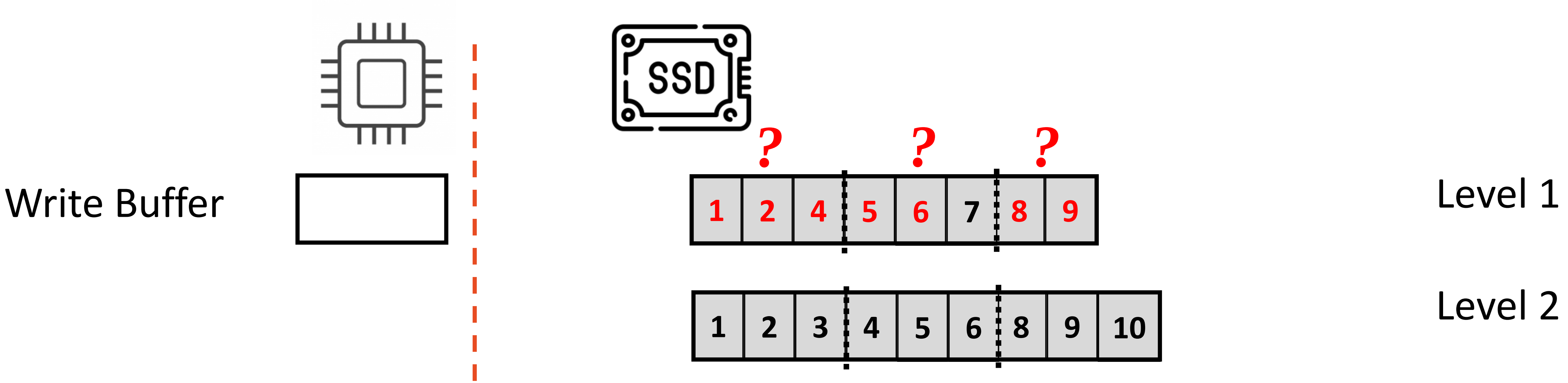
Level capacity grows by size ratio T

Leveling (Classic): one sorted run per level

Compaction introduces ***Write Amplification***, short as

Write Amp, defined as $\frac{\#bytes\ written\ to\ storage}{\#bytes\ inserted\ by\ user}$

LSM Basics - Compaction



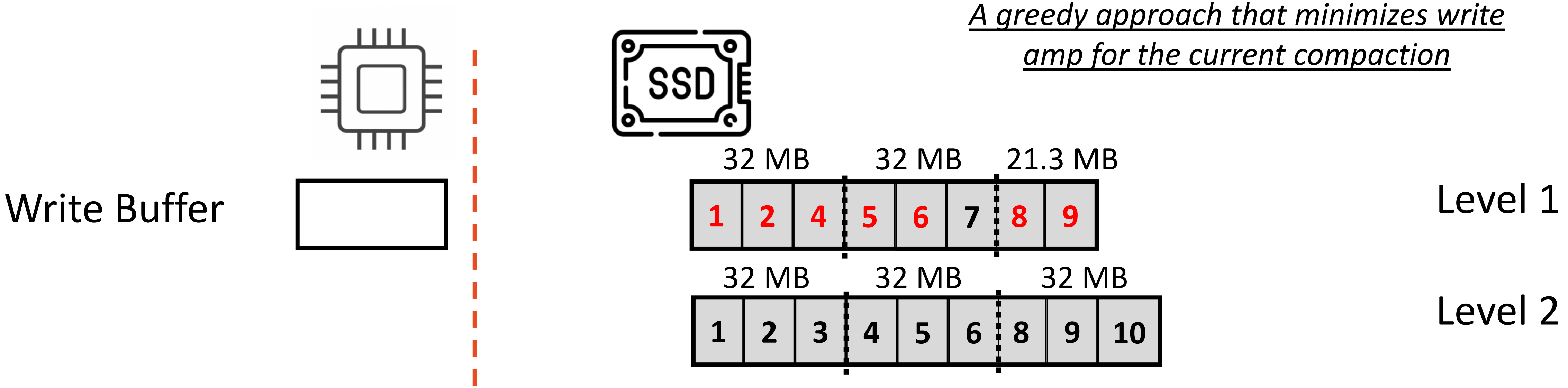
Full compaction -> *High compaction latency*

Partial compaction -> Selecting one file to compact

Which file in level 1 should we choose?

LSM Basics - MinOverlappingRatio (MOR)

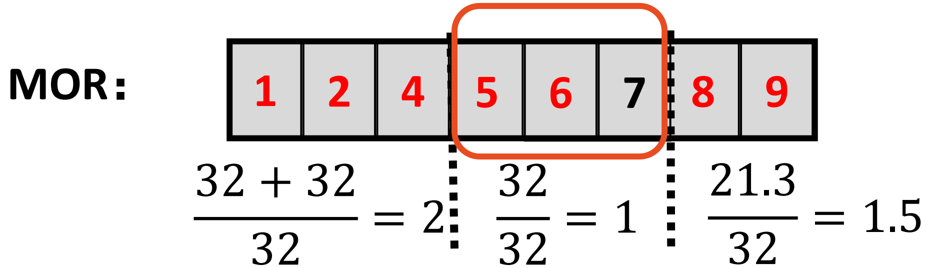
A greedy approach that minimizes write amp for the current compaction



Overlapping Ratio for file i in level 1: $\frac{\sum_{j \in \Theta_i} fs_j}{fs_i}$

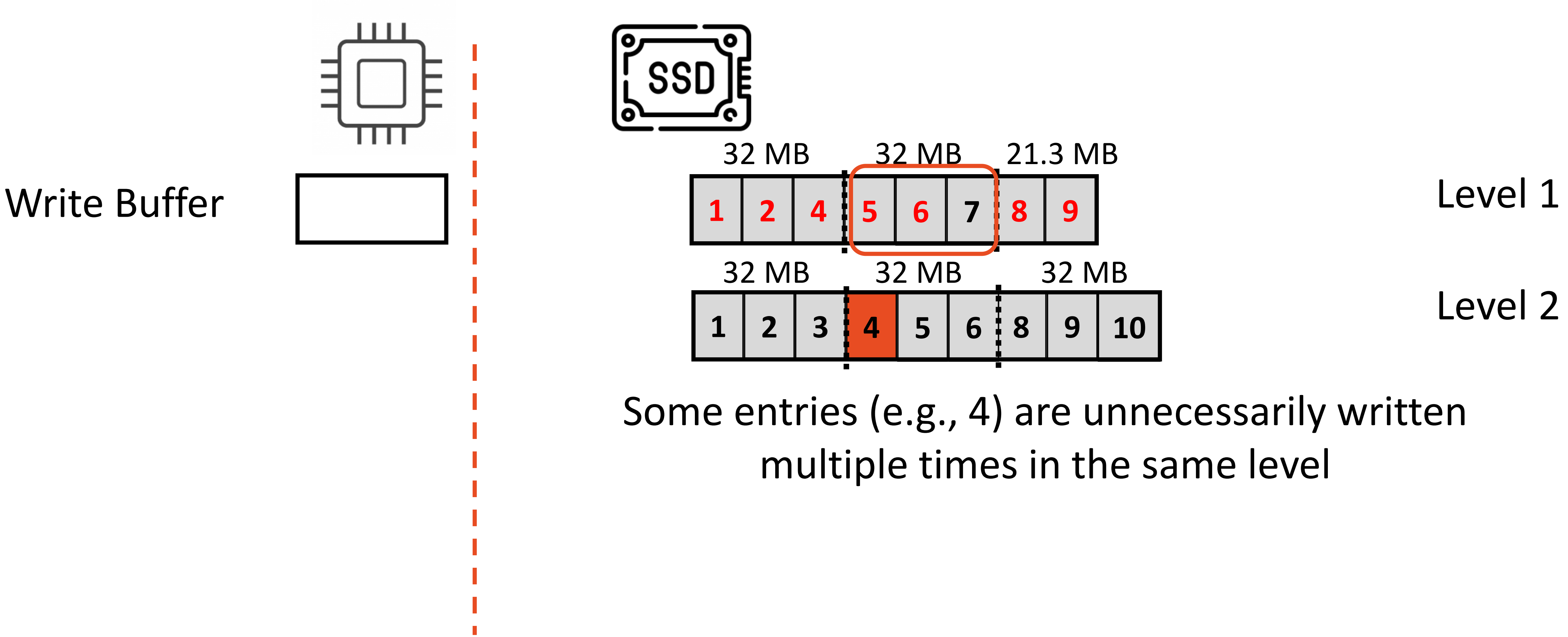
Θ_i ≔ the set of files in level 2 that overlap with file i

fs_i ≔ the file size of file i



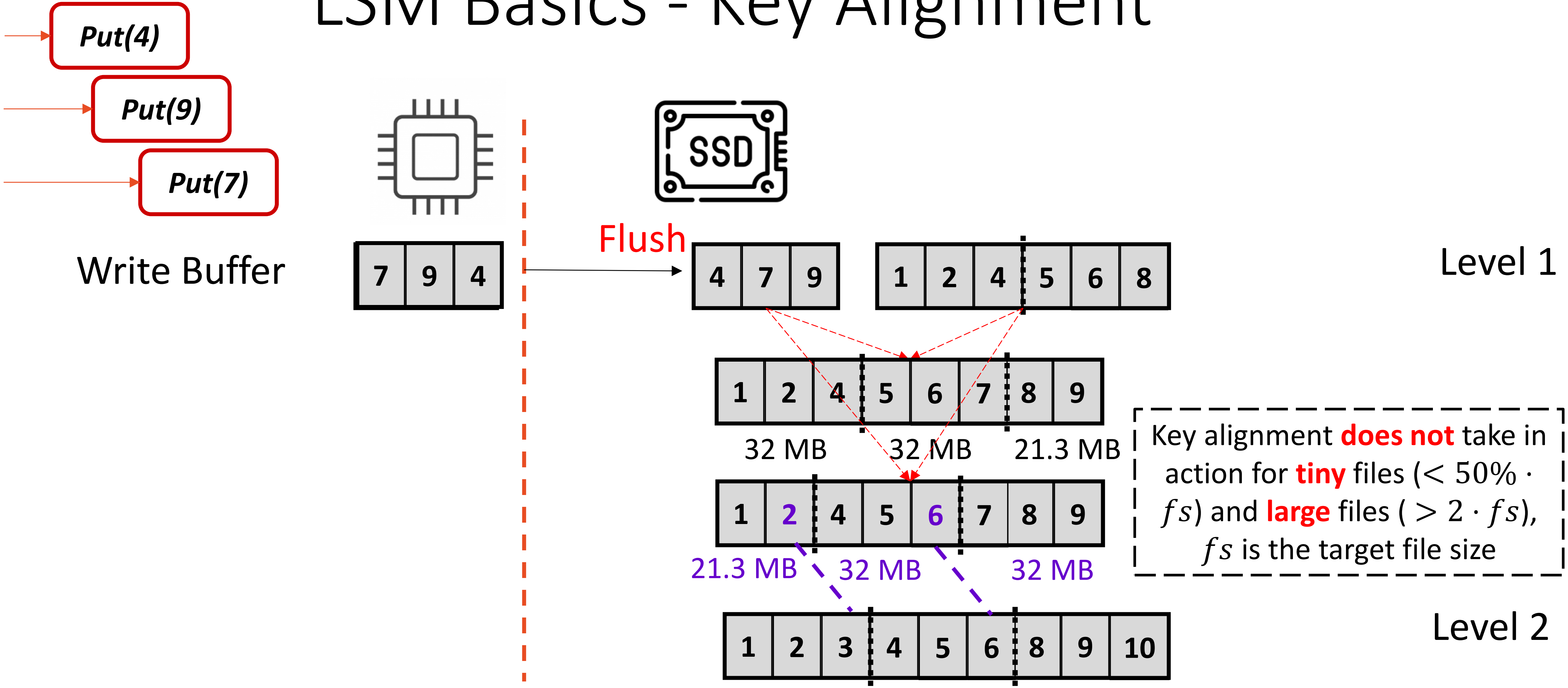
$\frac{32+32}{32} = 2$ $\frac{32}{32} = 1$ $\frac{21.3}{32} = 1.5$

LSM Basics - Key Alignment



Some entries (e.g., 4) are unnecessarily written multiple times in the same level

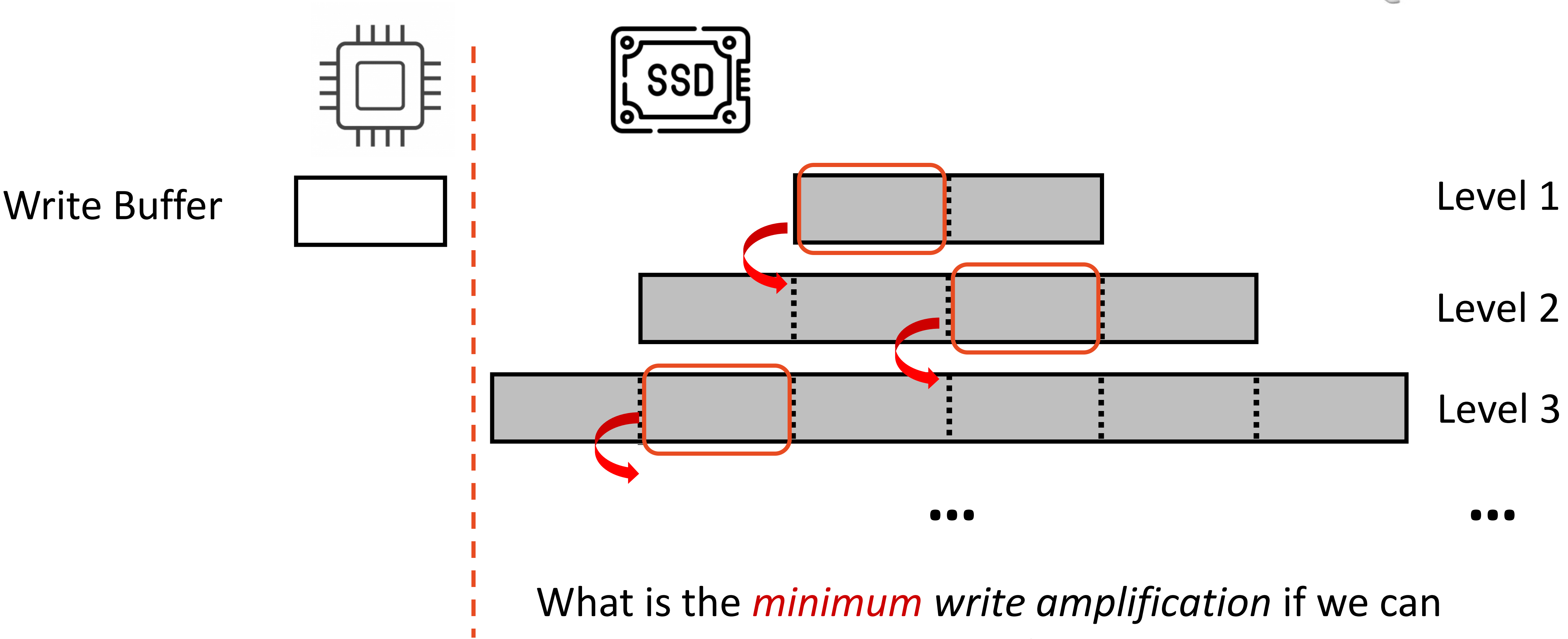
LSM Basics - Key Alignment



Key alignment **does not** take in action for **tiny** files ($< 50\% \cdot fs$) and **large** files ($> 2 \cdot fs$), fs is the target file size

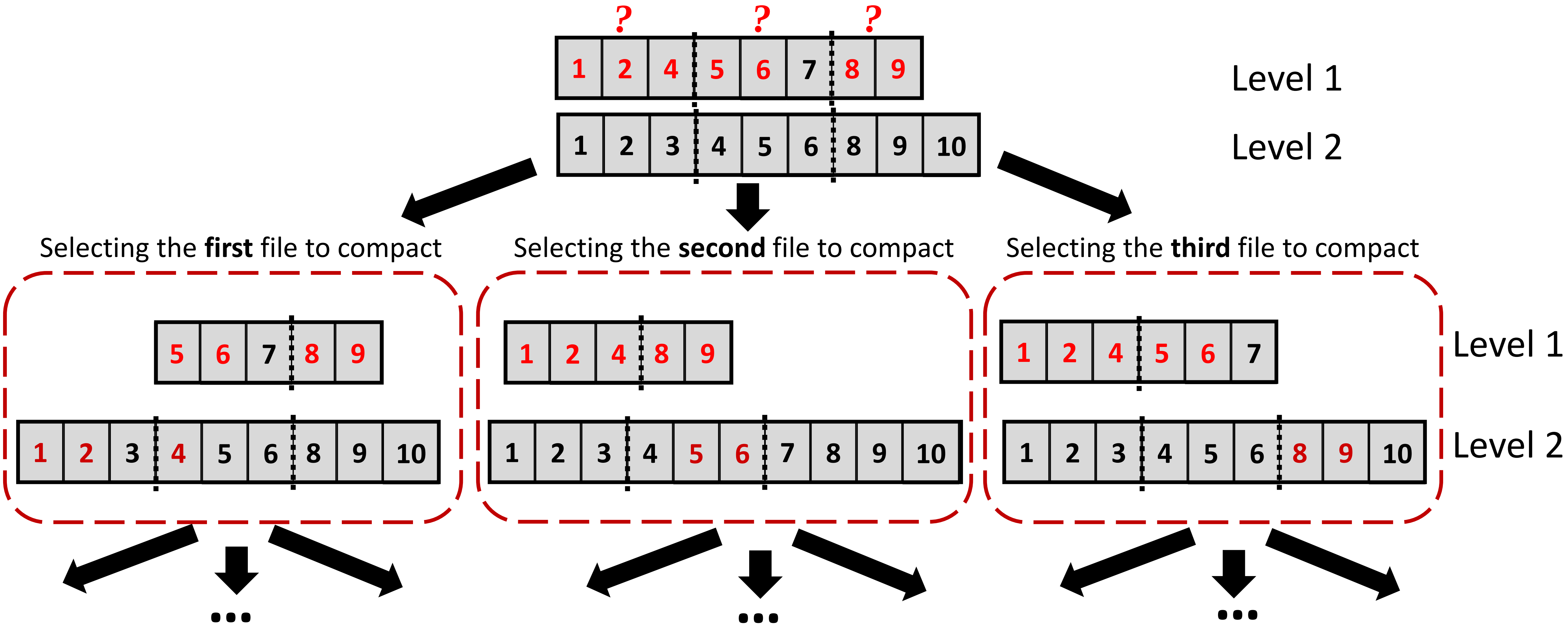
Key alignment reduces unnecessary rewritings in the same level

What is the Minimum Write-Amp?



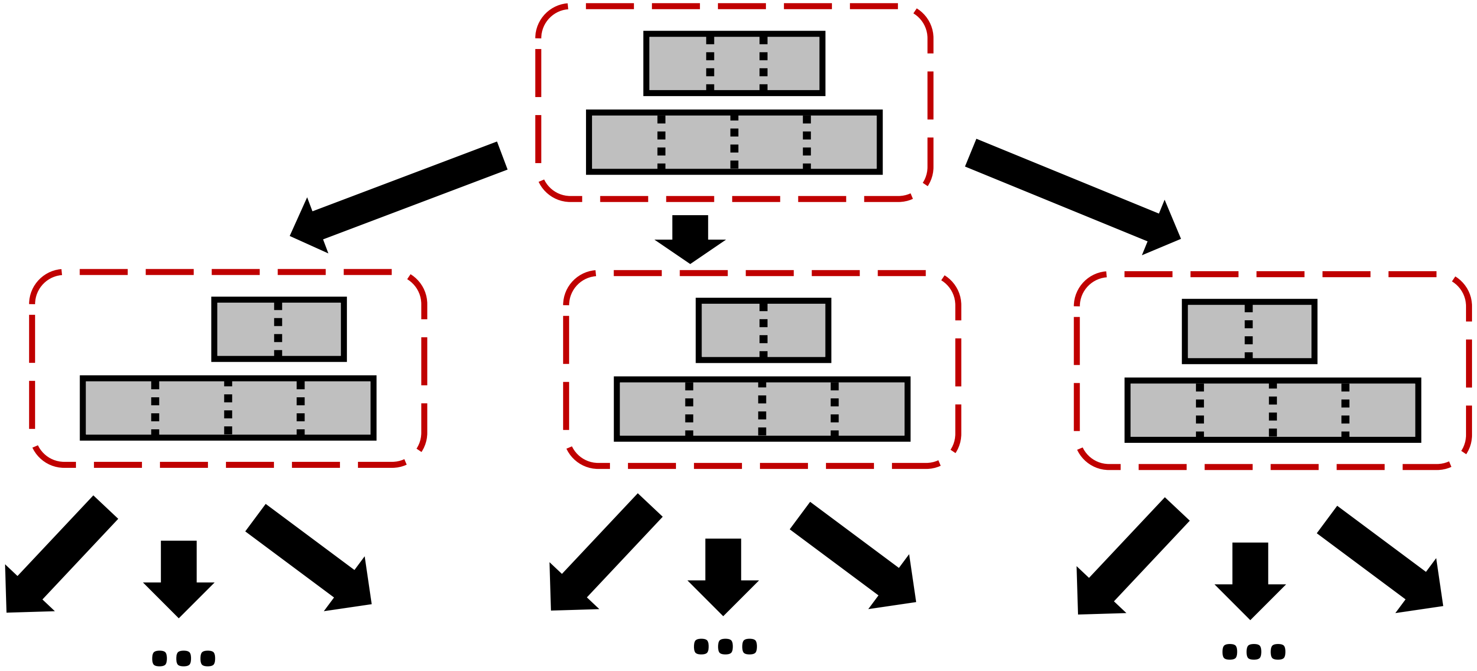
What is the *minimum write amplification* if we can smartly choose the file to compact?

Write-Amp is Hard to Model



Picking a different file can have cascadingly different impact over the LSM-tree shape

Write-Amp is Hard to Optimize

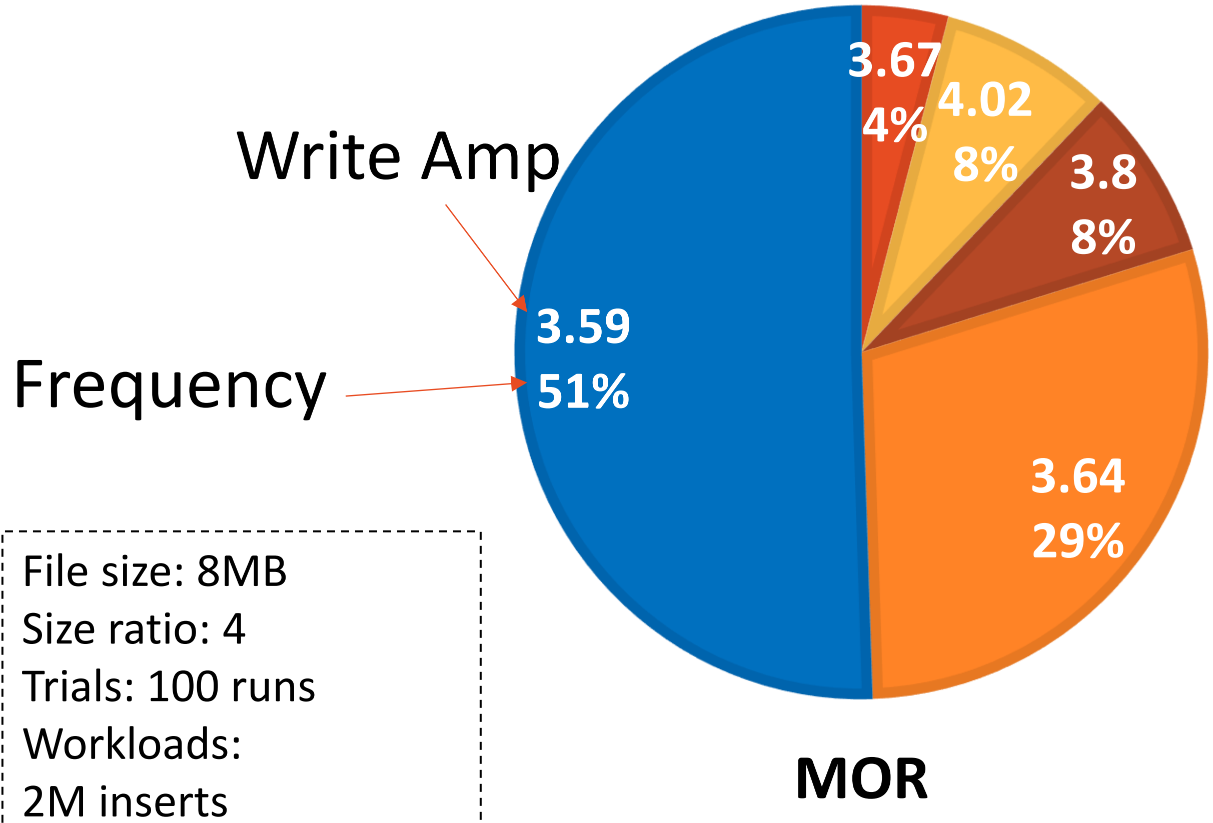


Brute Force: Depth-First Searching/Breadth-First Searching

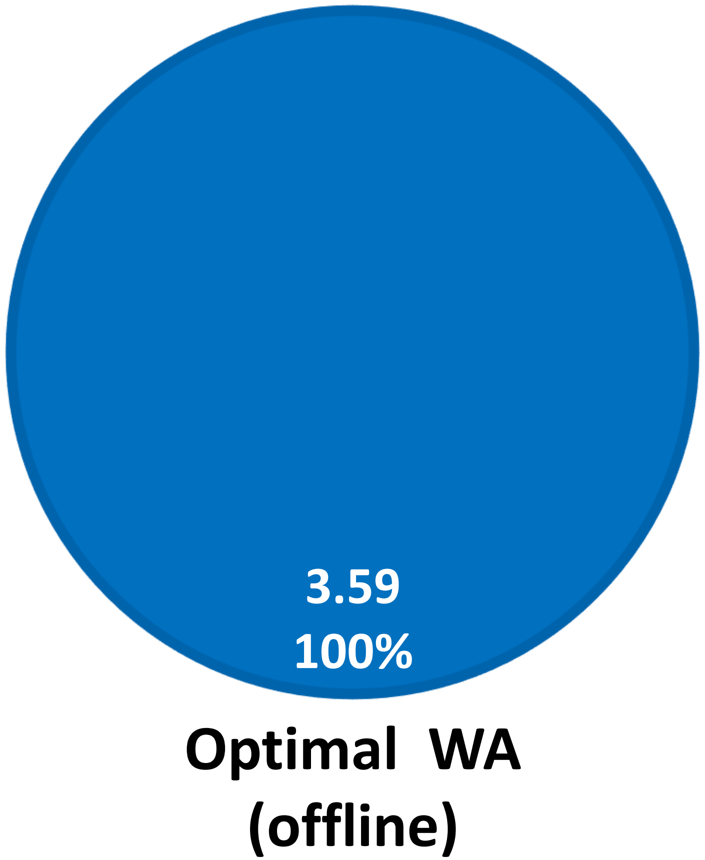
⇓ ⇓ ⇓

Exponential searching space!

MinOverlappingRatio (MOR)

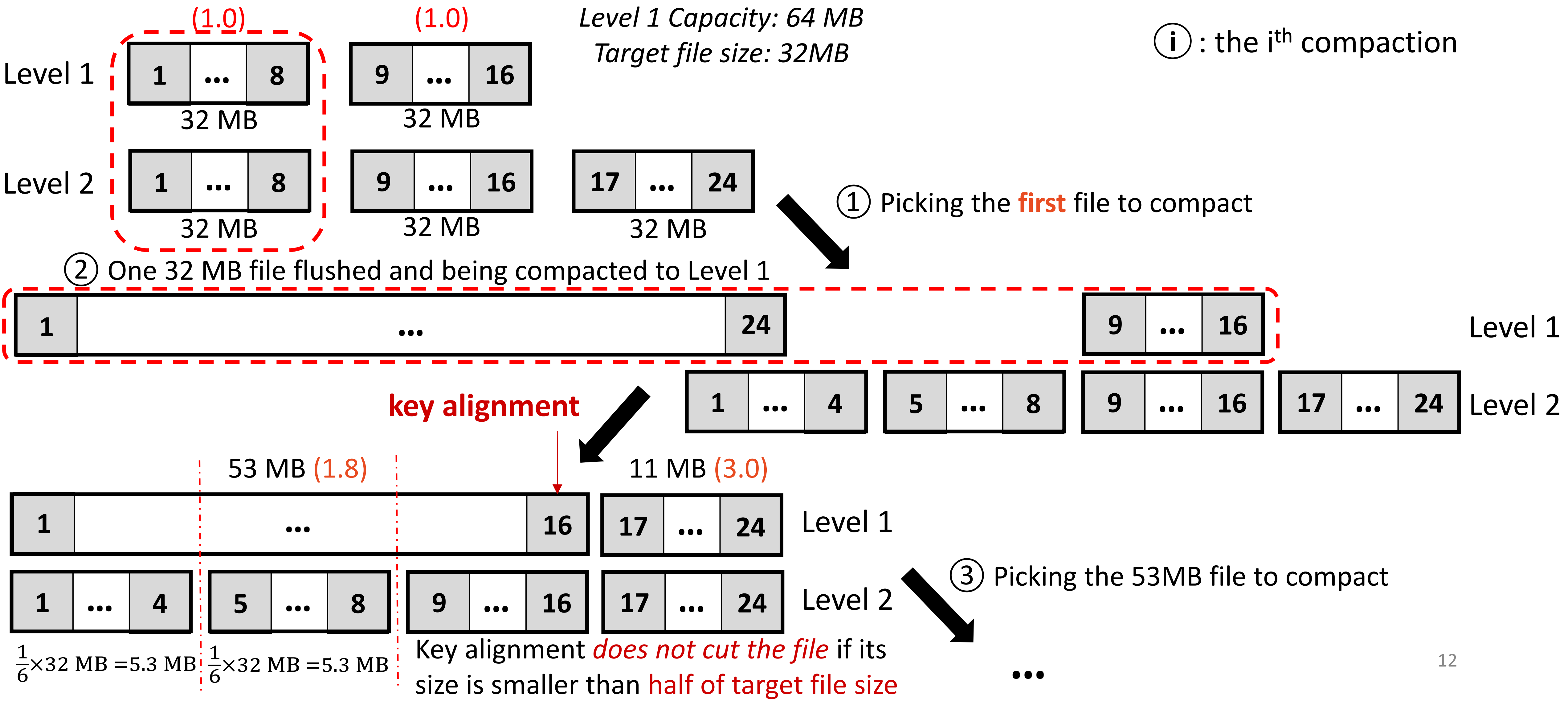


File size: 8MB
Size ratio: 4
Trials: 100 runs
Workloads:
2M inserts
8-byte key
56-byte value



MinOverlappingRatio (MOR) file picking policy can **reach the minimum write amp** but it is **unstable**. ?

Example of MinOverlappingRatio



Level 1 Capacity: 64 MB
Target file size: 32MB

ⓘ : the ith compaction

① Picking the **first** file to compact

② One 32 MB file flushed and being compacted to Level 1

key alignment

$\frac{1}{6}$×32 MB =5.3 MB

$\frac{1}{6}$×32 MB =5.3 MB

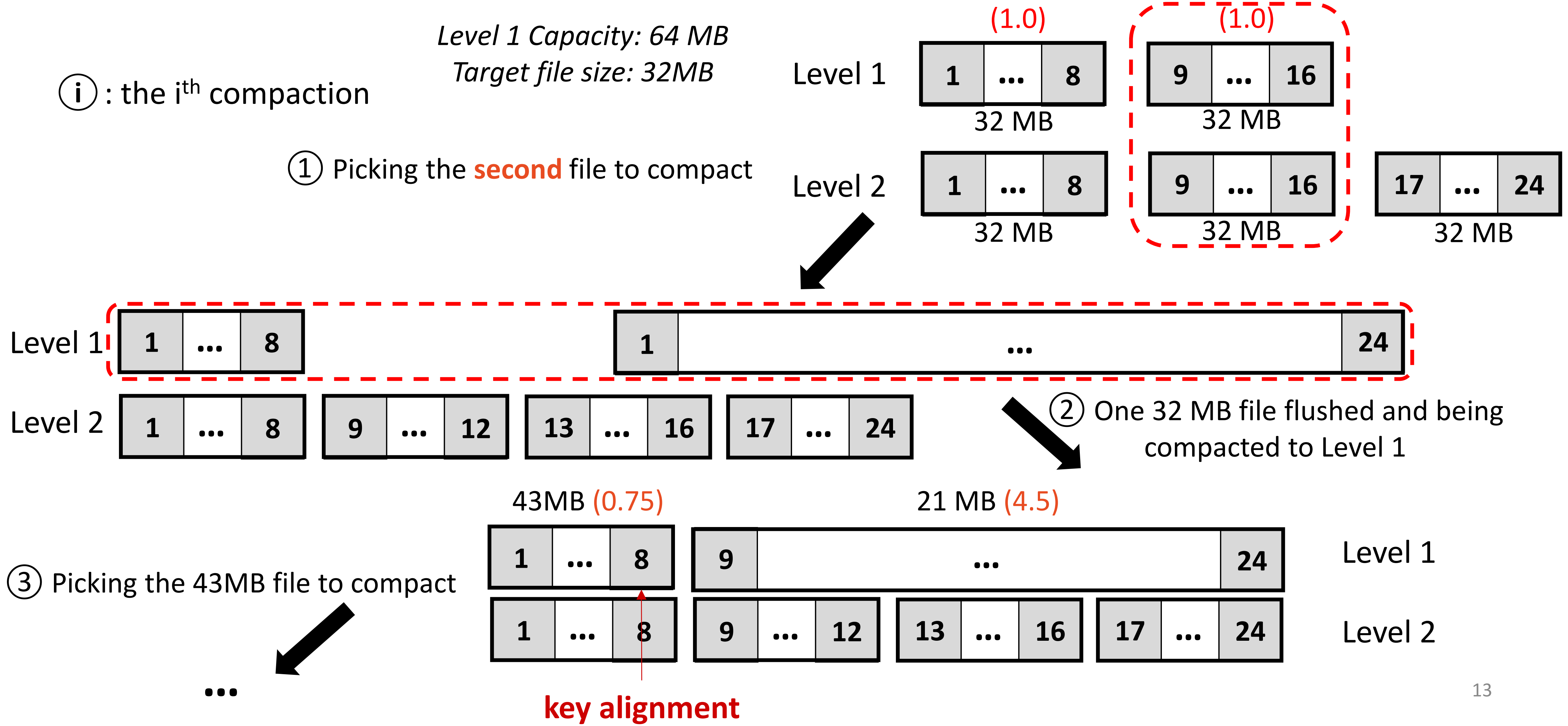
Key alignment *does not cut the file* if its size is smaller than half of target file size

③ Picking the 53MB file to compact

Example of MinOverlappingRatio

ⓘ : the ith compaction

Level 1 Capacity: 64 MB
Target file size: 32MB



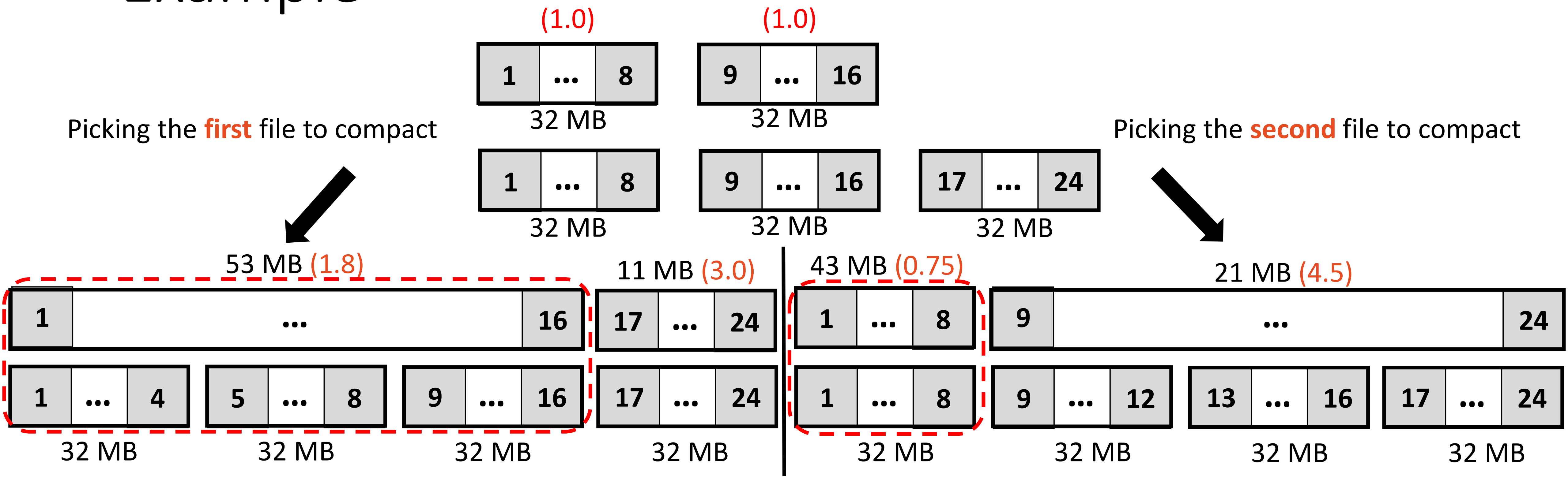
① Picking the **second** file to compact

② One 32 MB file flushed and being compacted to Level 1

③ Picking the 43MB file to compact

key alignment

Example

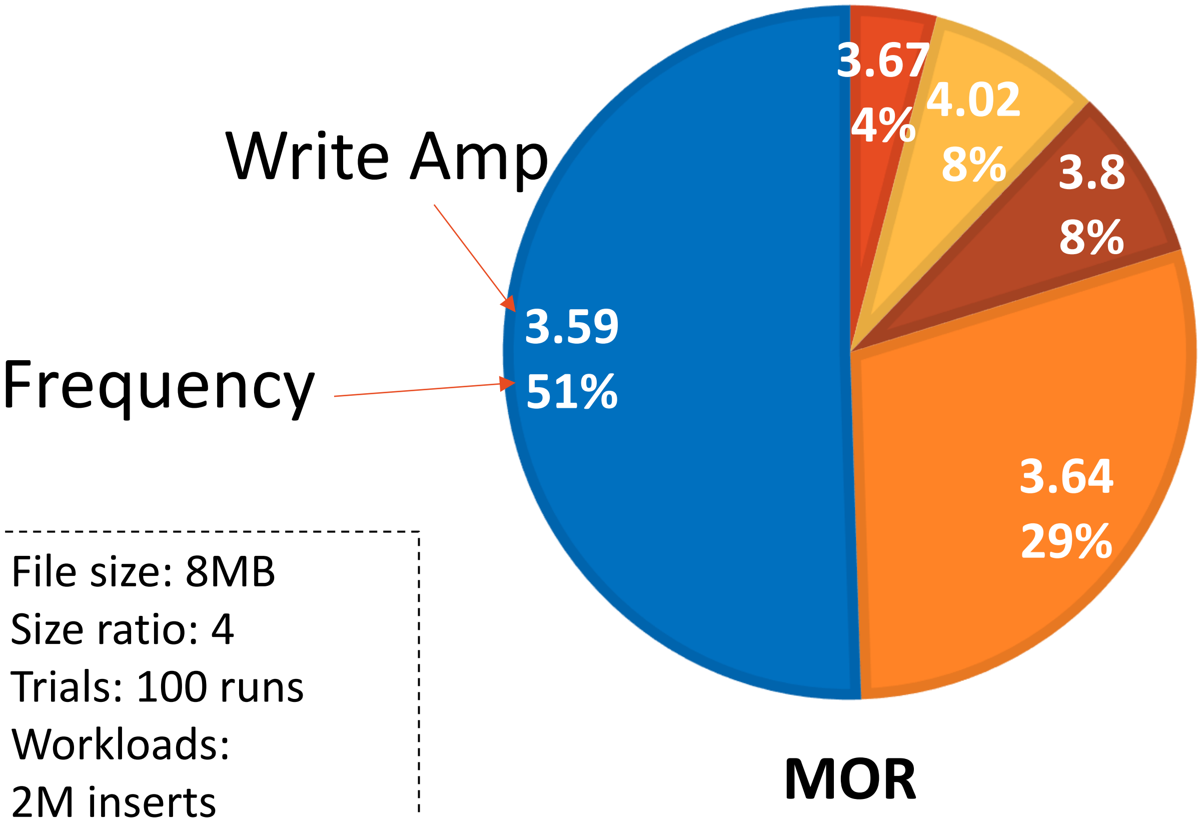


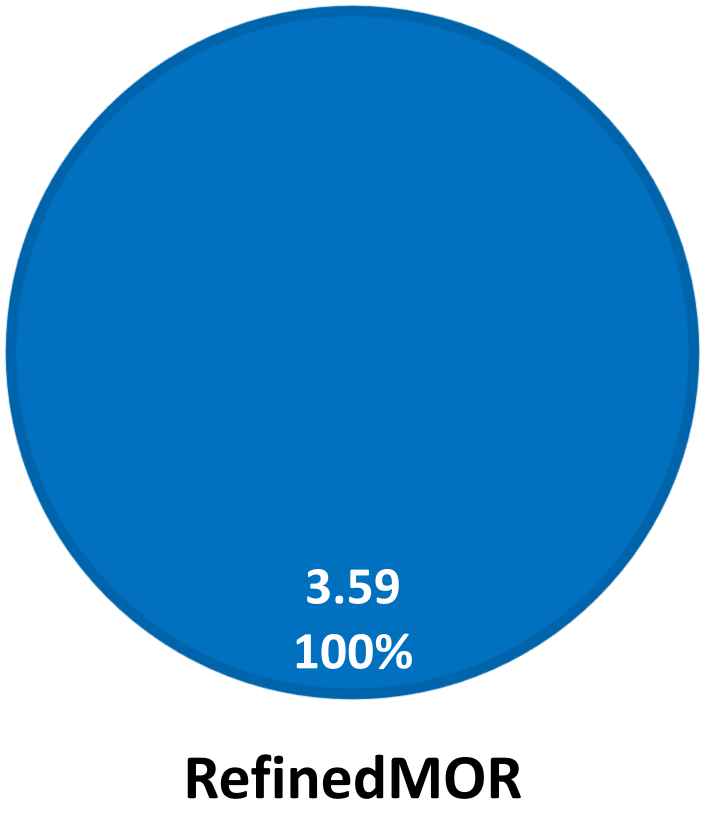
To achieve key alignment, future compaction merges the subsequent file of the picked one.

⇓ ⇓ ⇓

RefinedMOR: Pick the one of which the subsequent file (if exists) has the largest overlapping ratio from minimum ones.

Experimental Results

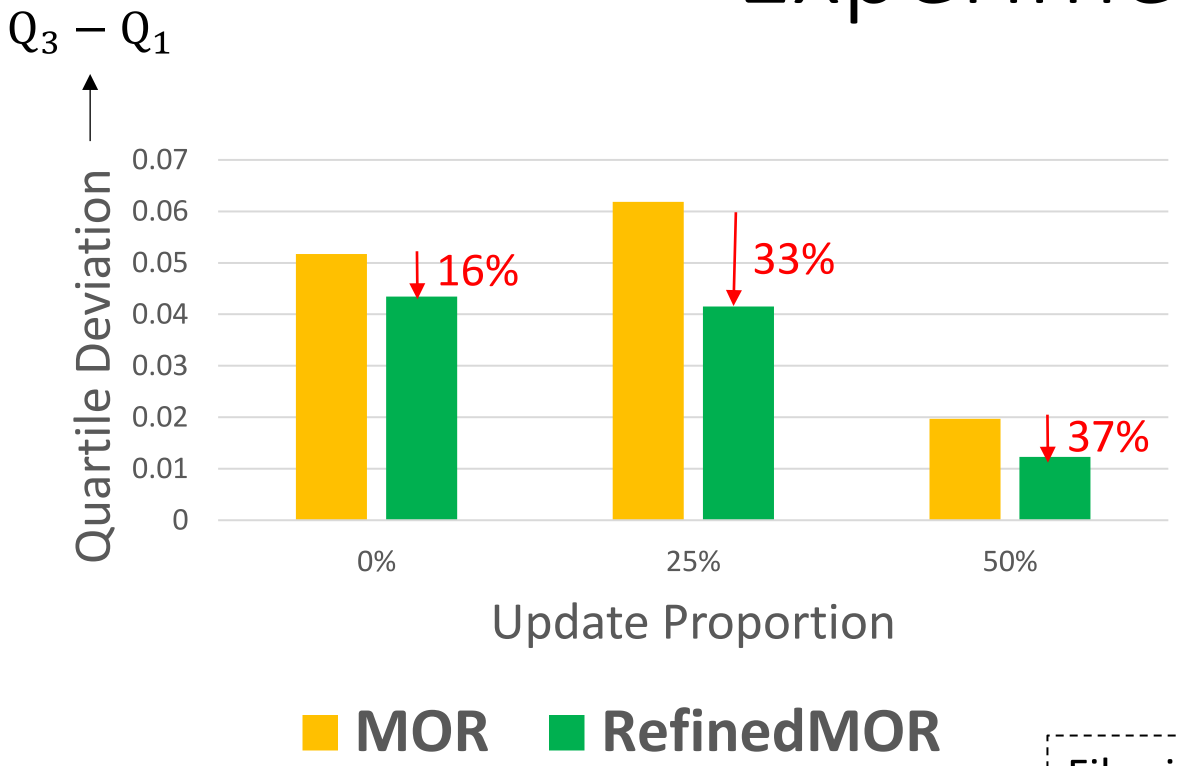




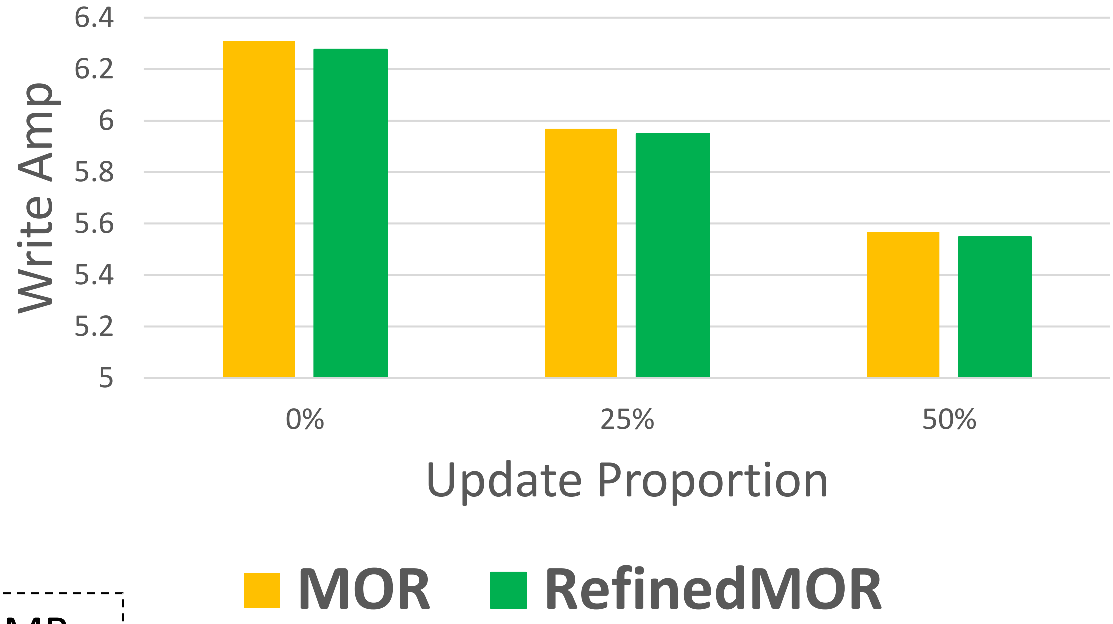
File size: 8MB
Size ratio: 4
Trials: 100 runs
Workloads:
2M inserts
8-byte key
56-byte value

RefinedMOR achieves the minimum write amp in all 100 runs for small workloads.

Experimental Results



File size: 8MB
Size ratio: 4
Trials: 20 runs

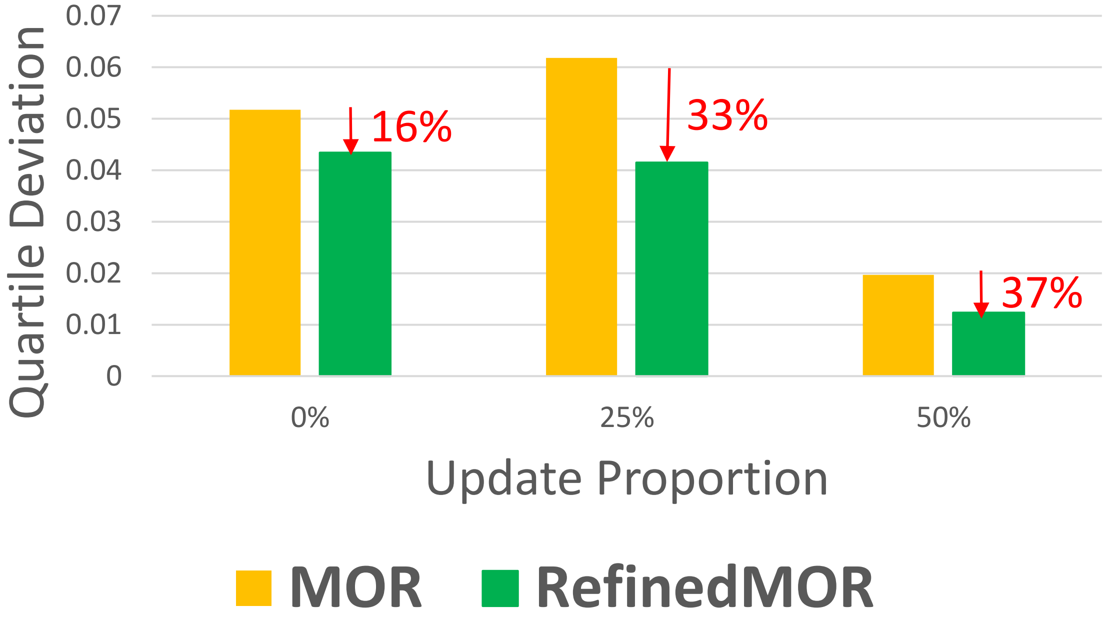


Workload:
1KB entries
40M operations
Uniform distribution

RefinedMOR has **similar average write amp** and **much lower quartile deviation** compared to MOR.

Summary of RefinedMOR

RefinedMOR reduces the quartile deviation (i.e., $Q_3 - Q_1$) by up to 37% without worsening the average write amp.



Other Observations

Round-Robin selection policy favors workloads with update skew.

Slower storage devices have lower WA but higher space amplification.

Picking policy has low impact over WA for update-intensive workloads.

Trivial move should be always prioritized to compaction.

More can be found in our full paper *Benchmarking, Analyzing, and Optimizing Write Amplification of Partial Compaction in RocksDB.*

Q & A