RocksDB Festival

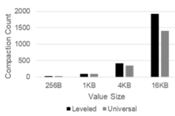
Supported by IITP, StarLab.

July 19, 2021
송인호, 한예진
inhoinno@dankook.ac.kr , hbb97225@naver.com
TeamName : 멘탈모델을 만들고 싶어요

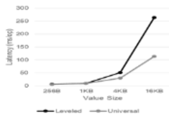
Contents

- Last Week
  - ✓ 컴팩션에 영향을 미치는 녀석들 #1 KV Size
  - ✓ 컴팩션에 영향을 미치는 녀석들 #2 SST Size

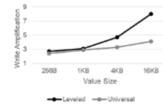
- This Week
  - ✓ Mental Model
    - Alleviating *[PROBLEMS]* of Level Compaction by *[Method]* in Universal Compaction
  - ✓ Leveled vs Universal Compaction Comparison
    - Throughput
    - Write Amplification
    - Latency distribution + # of Compactions
    - ~~Read/Space Amplification~~



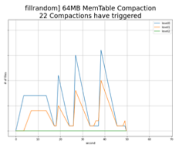
Compaction Monitoring



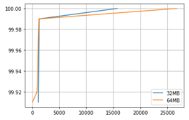
Write Latency



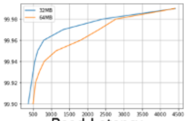
Write Amplification



Compaction Monitoring



Write Latency

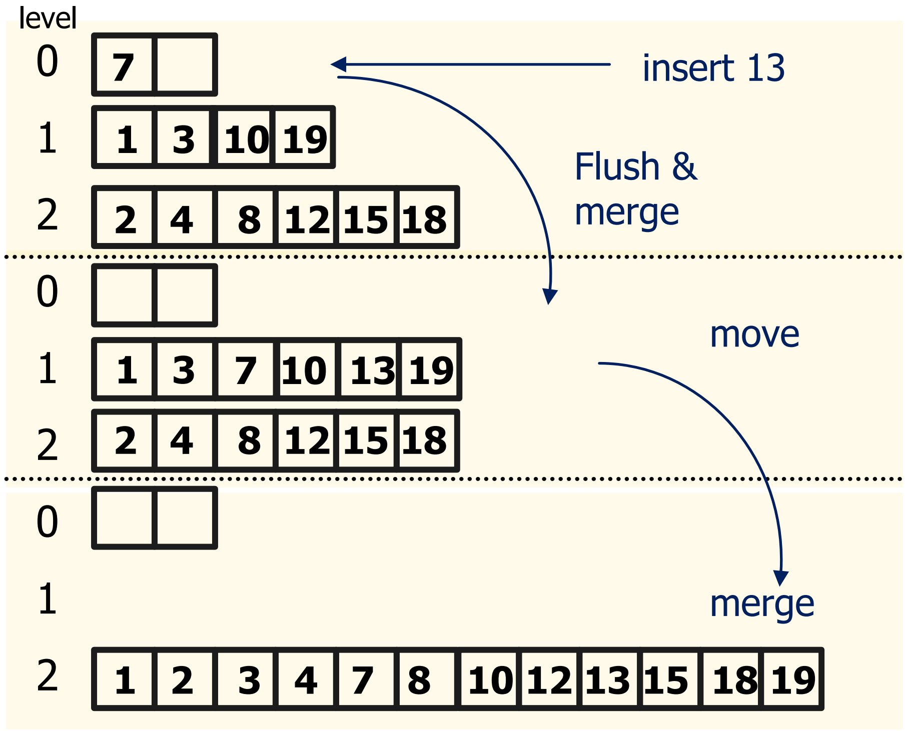


Read Latency

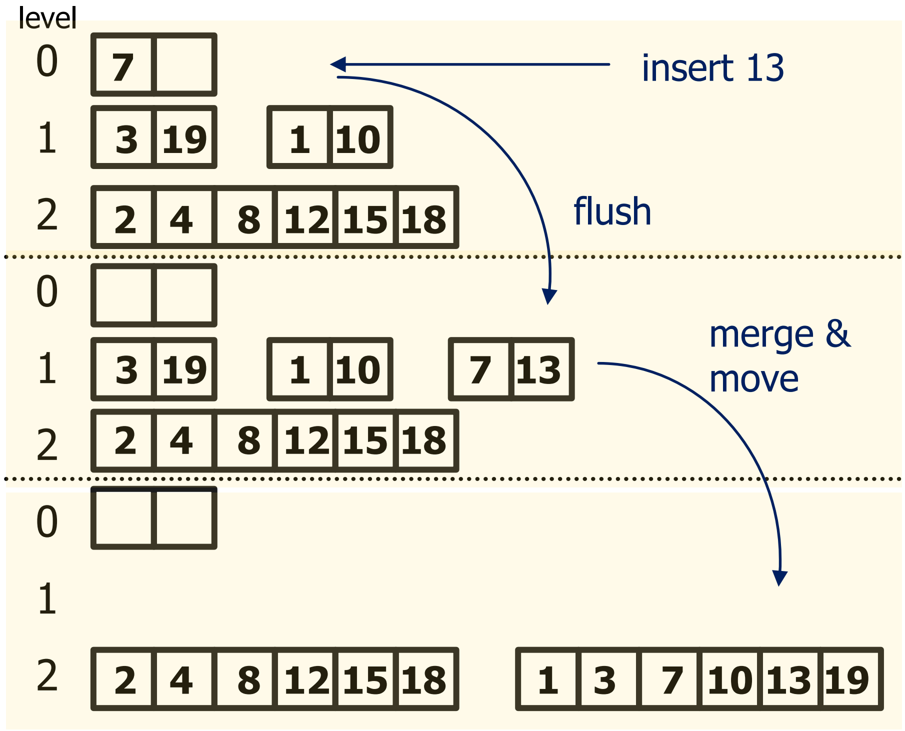
Compaction Style

- Leveled Compaction, Universal Compaction
  - ✓ Sorted Level vs Sorted Run

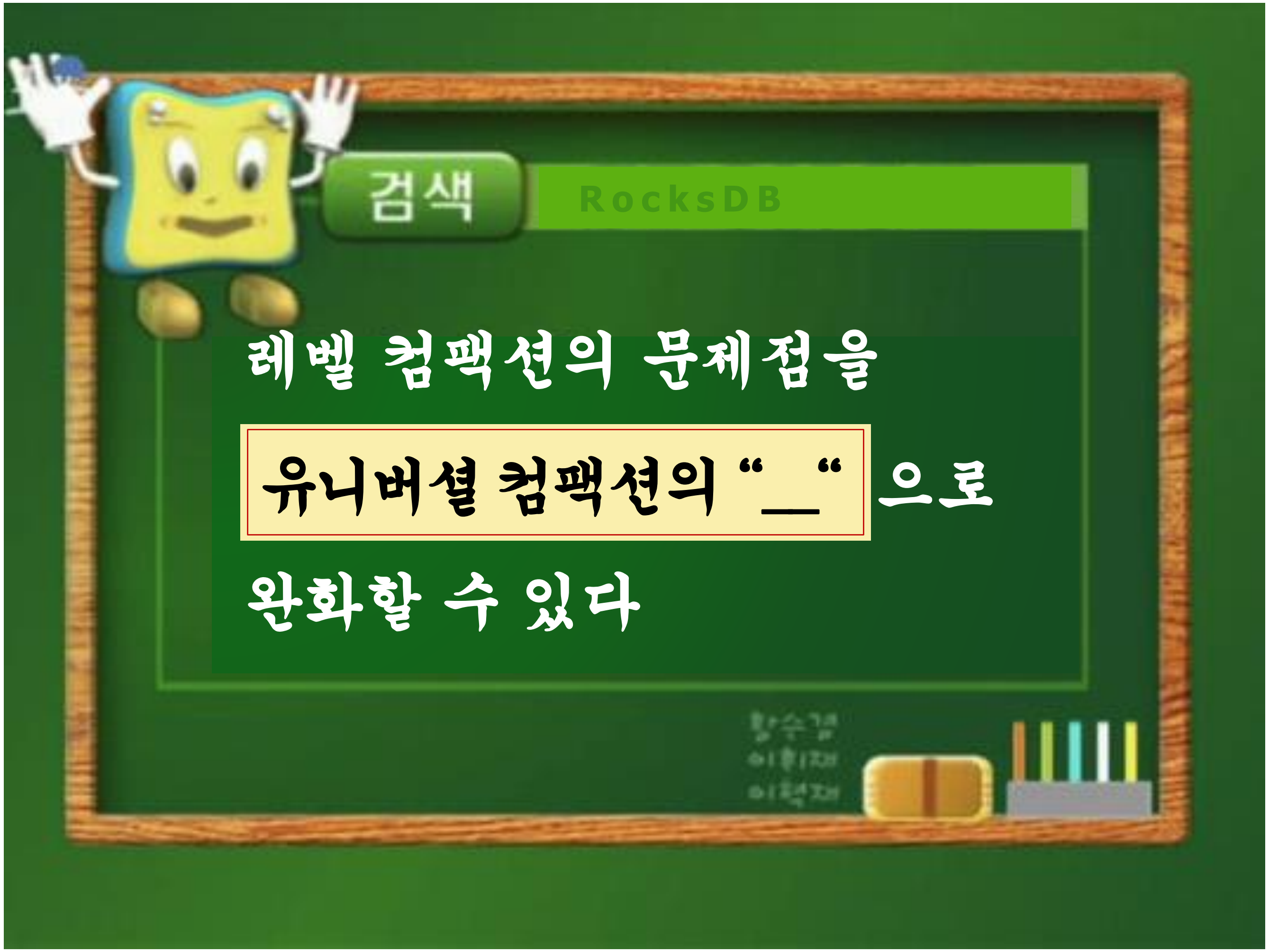
Example of LeveledCompaction



Example of Universal Compaction



Mental Model



레벨 컴팩션의 문제점을

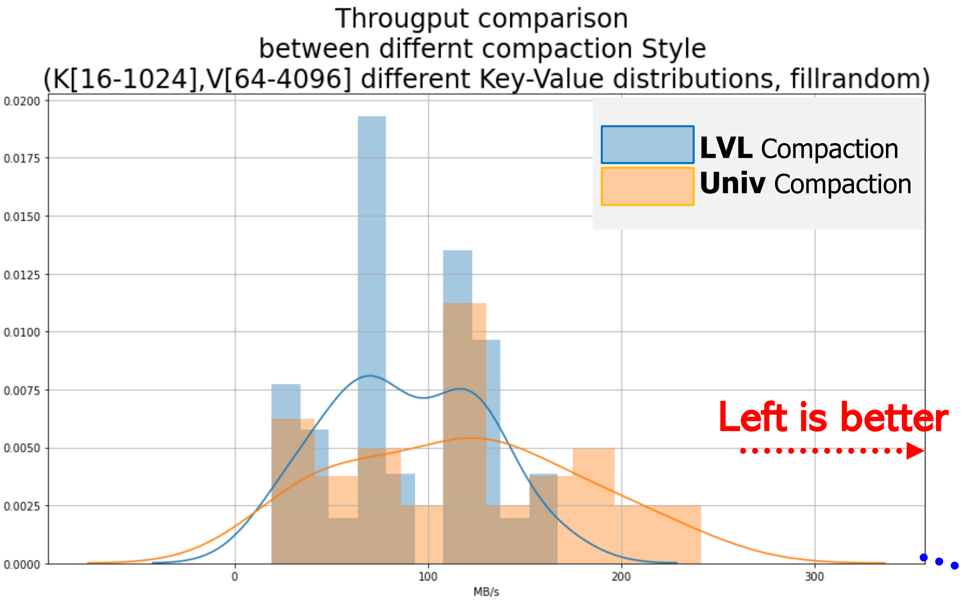
유니버셜 컴팩션의 "__" 으로

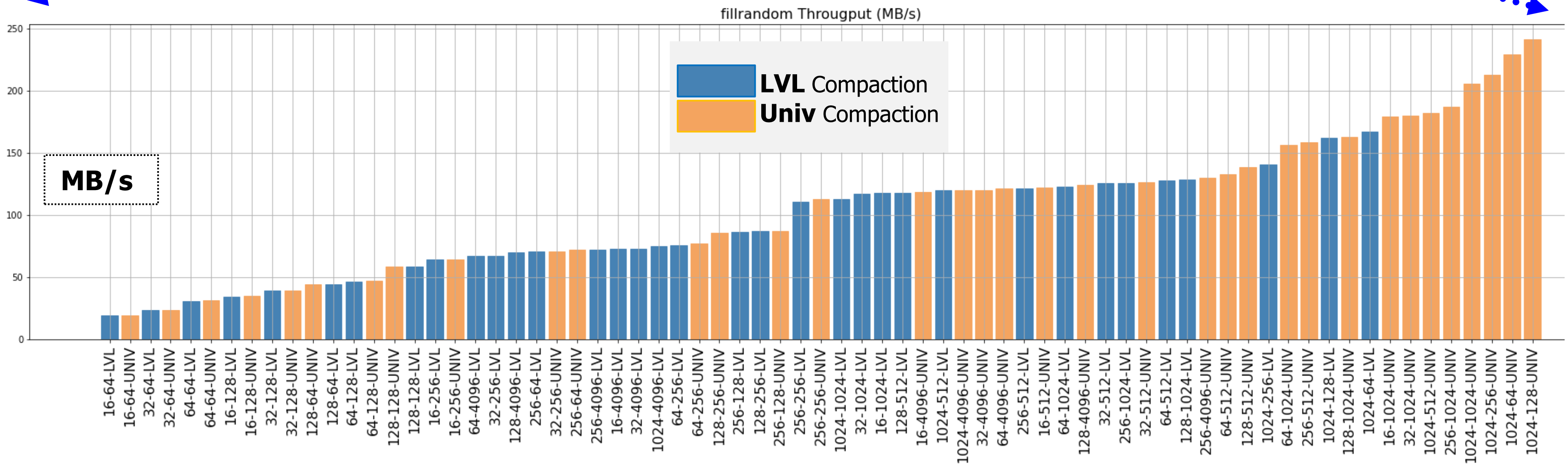
완화할 수 있다

LVL vs Univ Throughput Comparison

Fillrandom

Key [16, 32, 64, 128, 256, 1024]
Value [64, 128, 256, 512, 1024, 4096]
Entries 500 0000
Storage Samsung 1TB 860 Pro
File System Ext4
CPU Intel(R) Core(TM) i7-10700K CPU @ 3.80GHz





LVL vs Univ Throughput Comparison

Readrandom

--use_existing_db

Key [16, 32, 64, 128, 256, 1024]

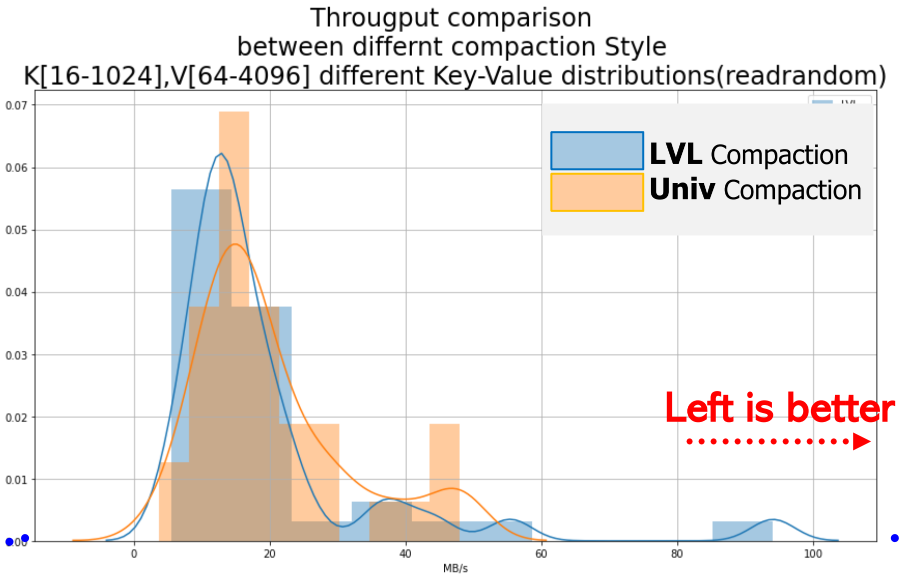
Value [64, 128, 256, 512, 1024, 4096]

Entries 500 0000

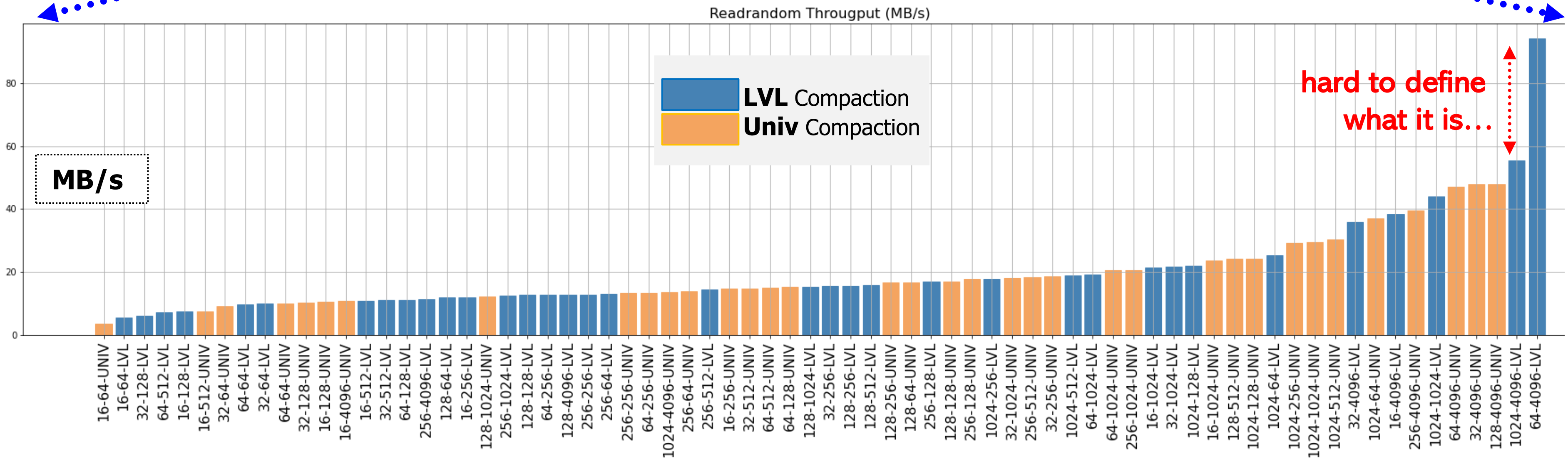
Storage Samsung 1TB 860 Pro

File System Ext4

CPU Intel(R) Core(TM) i7-10700K CPU @ 3.80GHz



Left is better



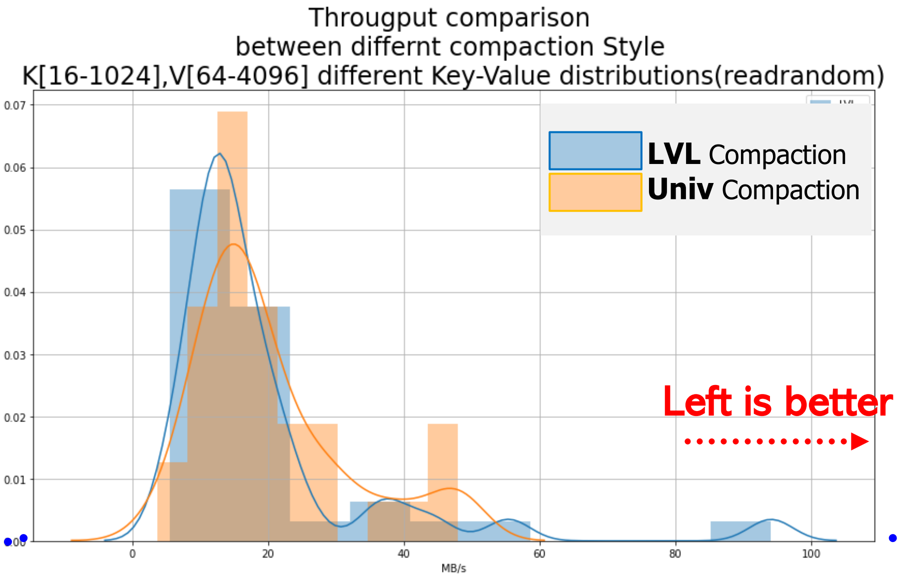
hard to define what it is…

LVL vs Univ Throughput Comparison

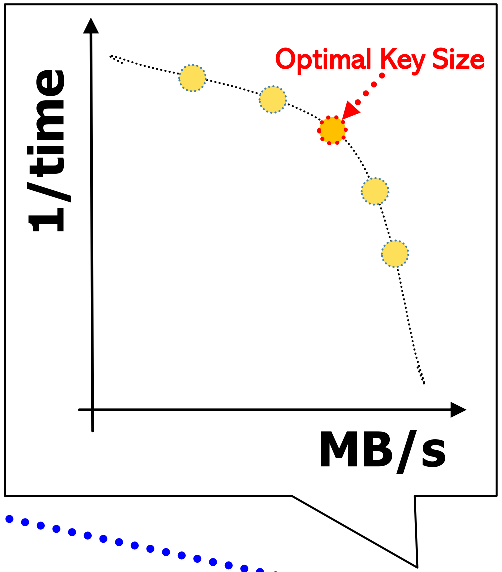
Readrandom

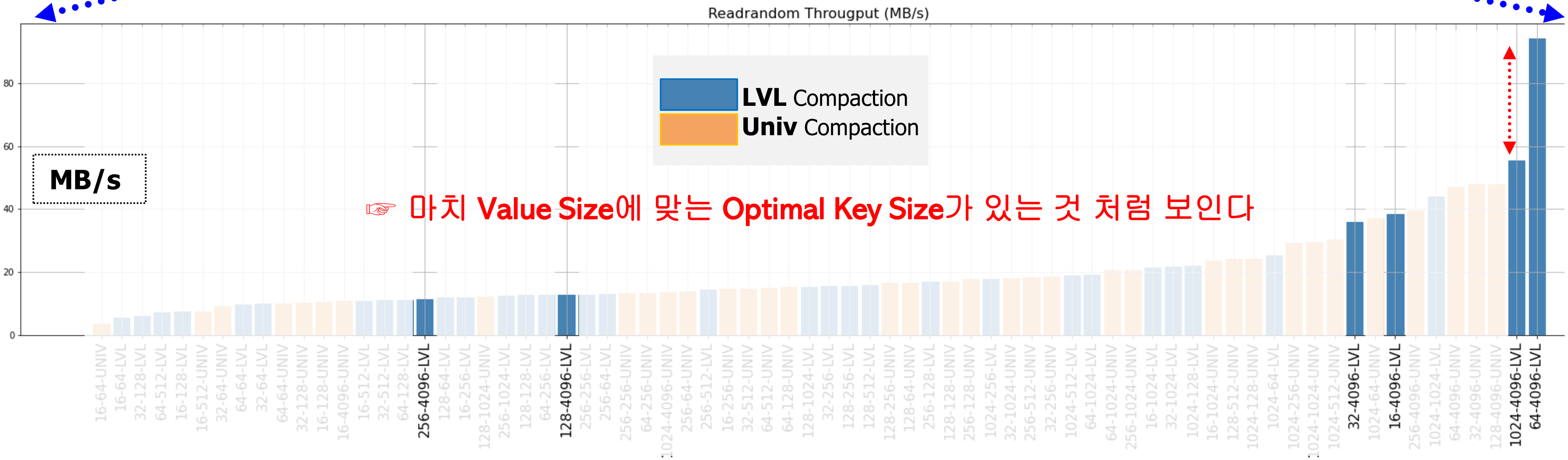
--use_existing_db

Key [16, 32, 64, 128, 256, 1024]
Value [64, 128, 256, 512, 1024, 4096]
Entries 500 0000
Storage Samsung 1TB 860 Pro
File System Ext4
CPU Intel(R) Core(TM) i7-10700K CPU @ 3.80GHz



Left is better

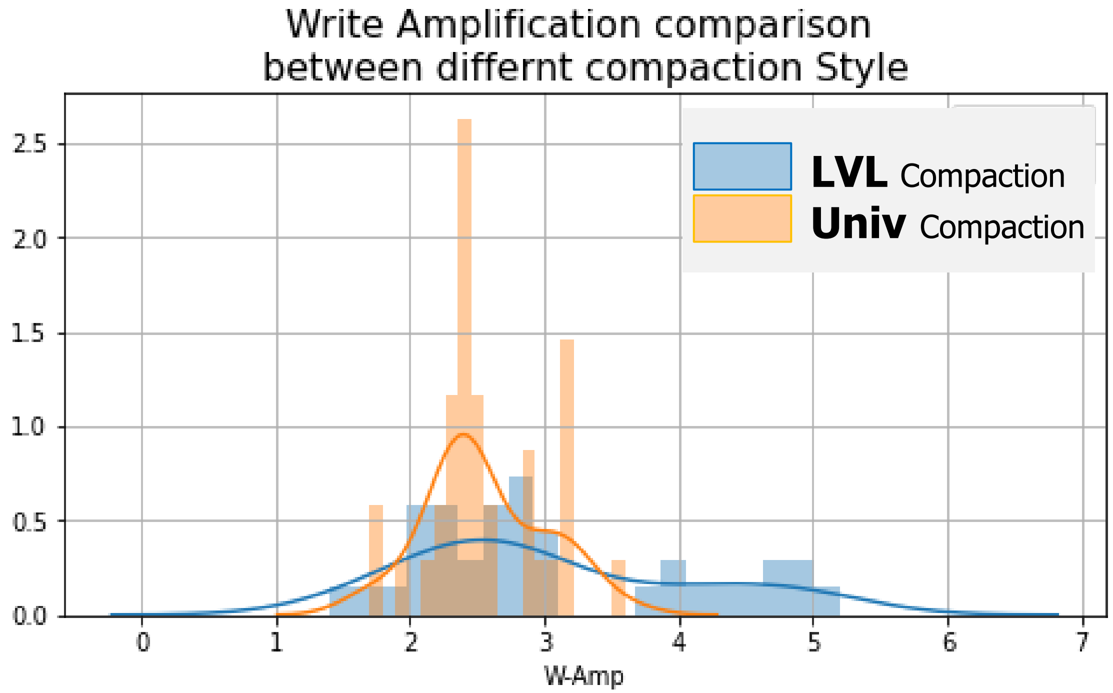


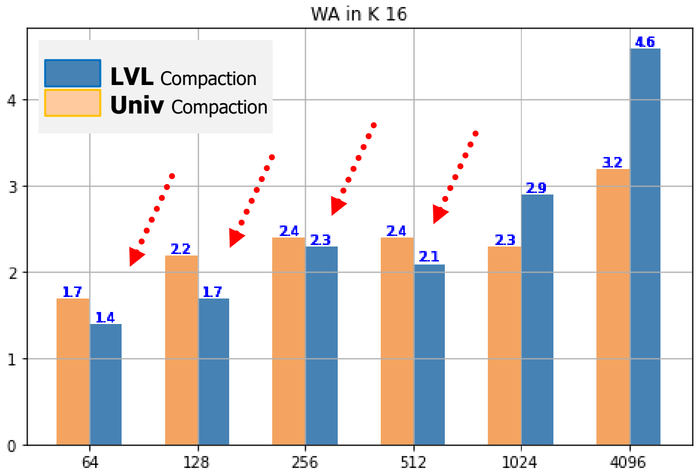


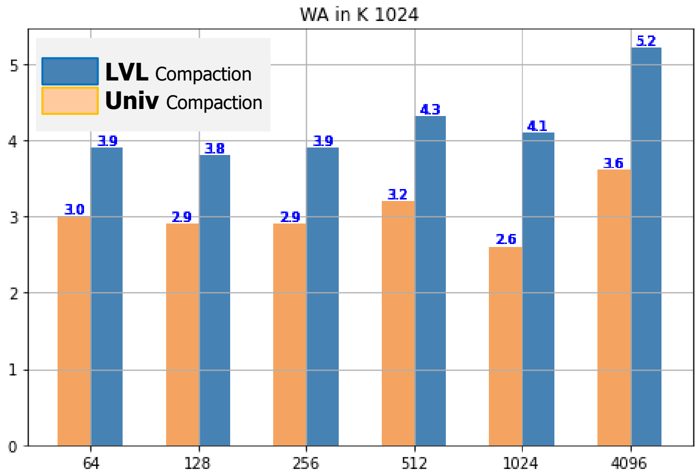
MB/s

☞ 마치 Value Size에 맞는 Optimal Key Size가 있는 것 처럼 보인다

LVL vs Univ WAF Comparison

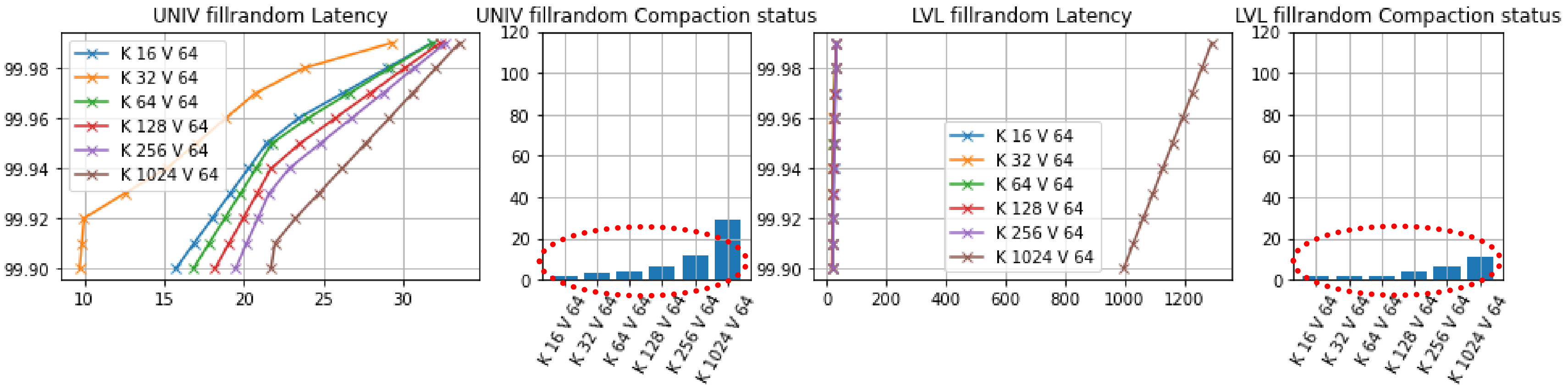


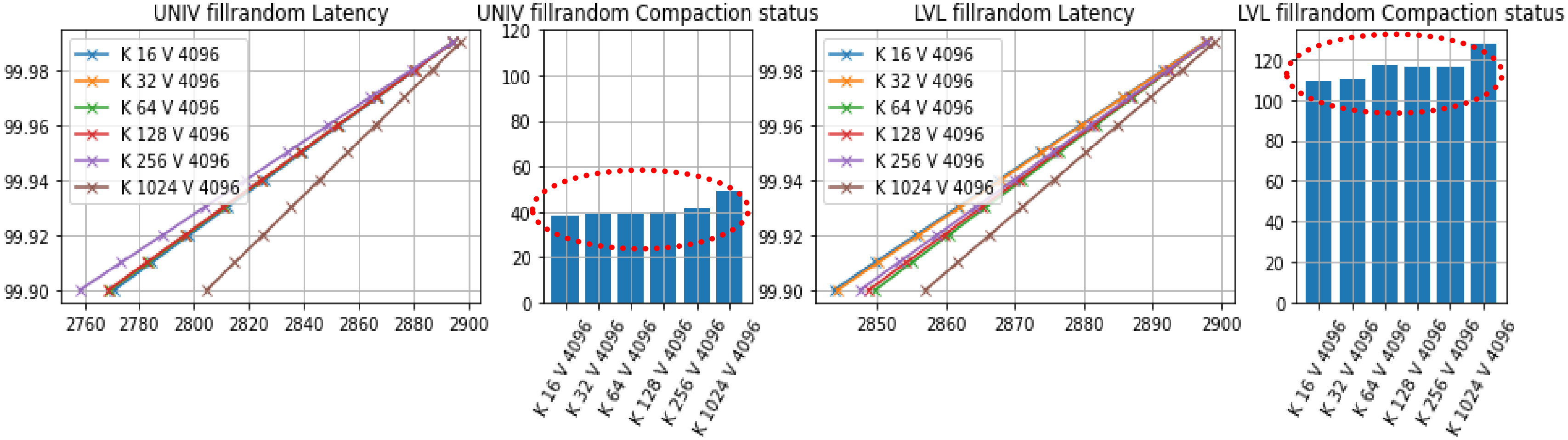




LVL vs Univ # of Compactions , latency Comparison

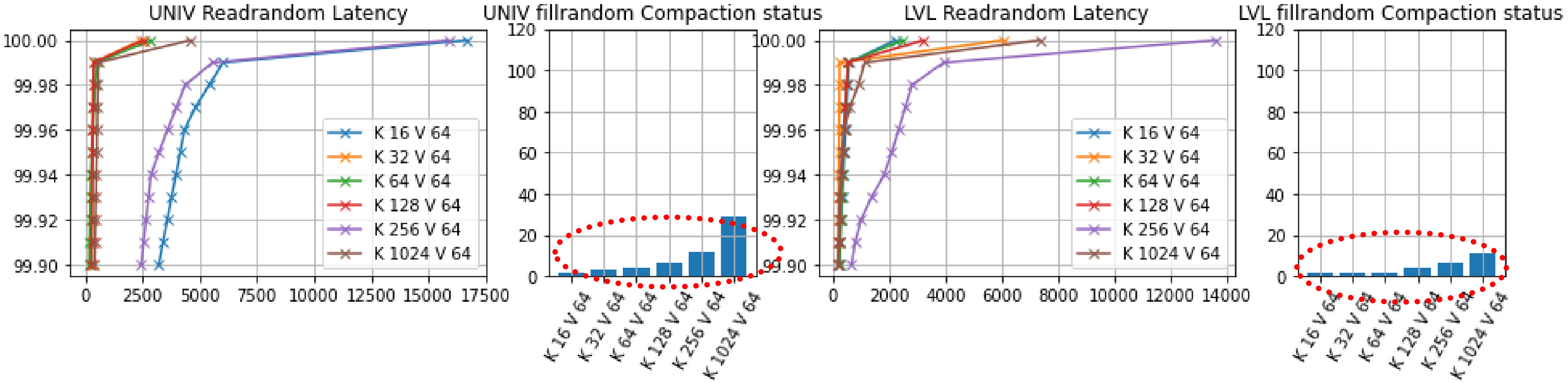
- Fillrandom

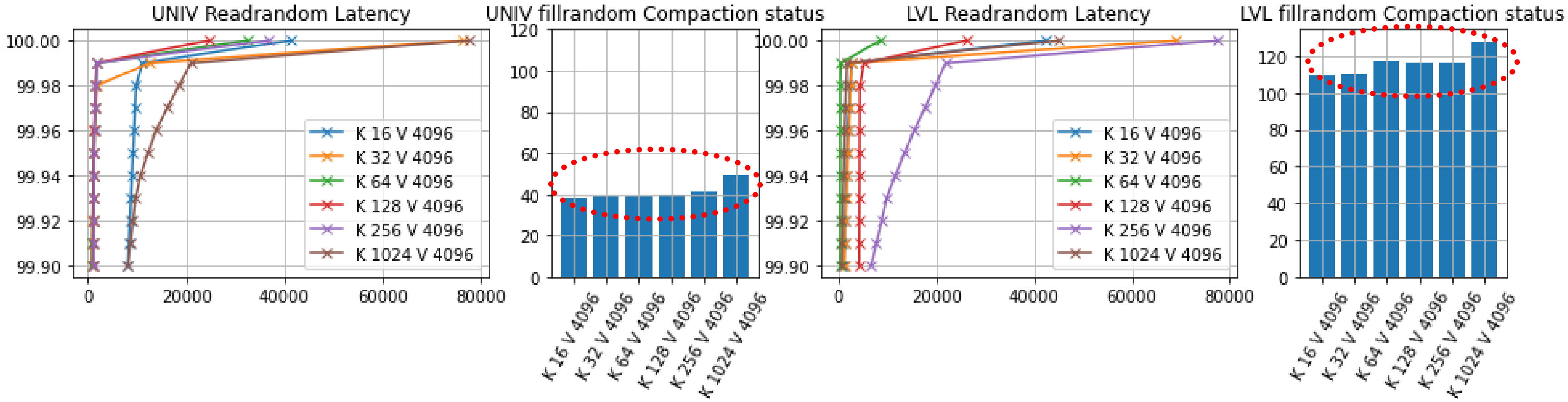




LVL vs Univ # of Compactions / latency Comparison

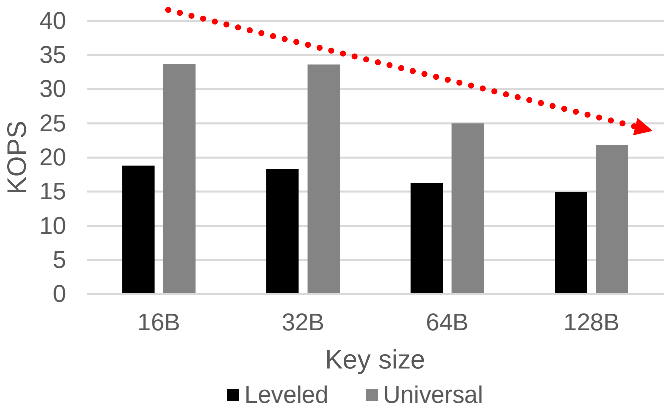
- Readrandom



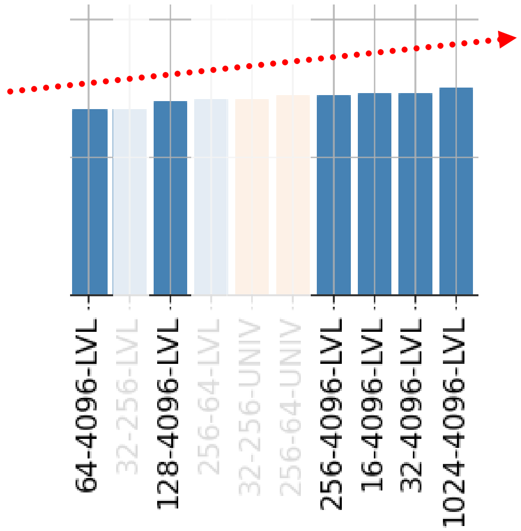


▪ Issue on Last week

Throughput(KOPS)



Throughput(MB/s)



☞ 한 쪽은 **OPS**, 한 쪽은 **MB/s** 임.

즉, **KV Size**가 늘어날 수록 **MB/s**는 늘어나고, **OPS**는 줄어들게 됨

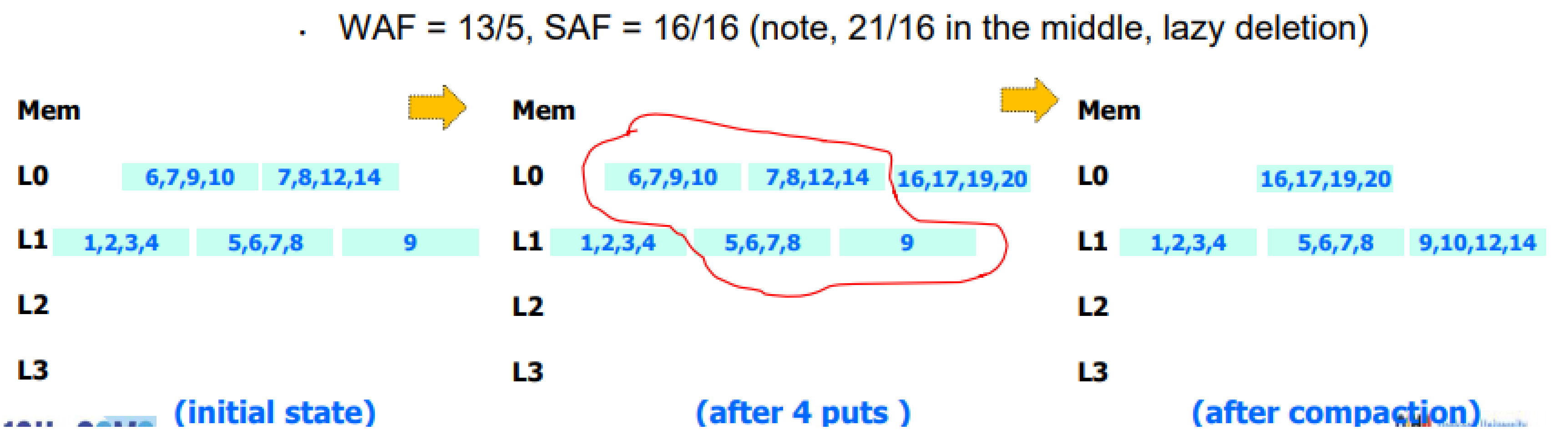
- Next Week
  - ✓ Mental Model
    - Leveled vs Universal Compaction 관련 논문 조사
    - Hybrid Compaction 고민
  - ✓ Leveled vs Universal Compaction Comparison
    - Space Amplification 측정
    - Read Amplification 측정

Discussion

Appendix

■ How to estimate Space Amplification Factor?

$$SAF = \frac{Actual\ used\ space}{Required\ space} = \frac{DB\text{에 실제로 저장된 공간}}{\text{최소로 필요한 공간}}$$ ☞ **Not so simple**



- WAF = 13/5, SAF = 16/16 (note, 21/16 in the middle, lazy deletion)

Mem
L0 6,7,9,10 | 7,8,12,14
L1 1,2,3,4 | 5,6,7,8 | 9
L2
L3
(initial state)

Mem
L0 6,7,9,10 | 7,8,12,14 | 16,17,19,20
L1 1,2,3,4 | 5,6,7,8 | 9
L2
L3
(after 4 puts)

Mem
L0 16,17,19,20
L1 1,2,3,4 | 5,6,7,8 | 9,10,12,14
L2
L3
(after compaction)

Jongmoo Choi, Key-Value Store: Database for Unstructured Bigdata (Focusing on RocksDB), 12th CSMS 융합·스마트미디어 시스템 워크샵

☞ **중첩된 KV 를 고려하지 않아야함**

☞ ☞ **따라서, (바라건대,) Sequential pattern 으로 쓰인 Size를** '최소로 필요한 공간' **으로 두고 구할 예정.**

☞ **Comment Plz**

Last Week

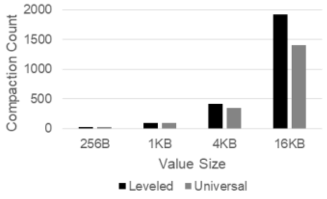
Compaction에 영향을 미치는 녀석들

✓ #1 KV-Size

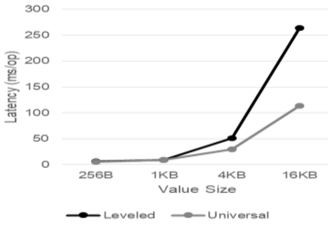
- Various Key Size
  - Key: 16B, 32B, 64B, 128B
  - Value: 8K
  - fillrandom, readrandom, range query, 5000000
  - Leveled Compaction vs. Universal Compaction
  - Write Amplification

- Various Value Size
  - Key: 16B
  - Value: 256B, 1KB, 4KB, 16KB
  - fillrandom, readrandom, range query, 5000000
  - Leveled Compaction vs. Universal Compaction
  - Write Amplification

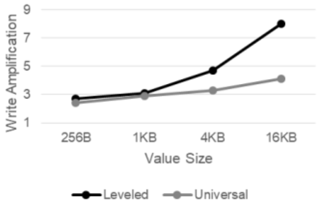
+YCSB Workload, compare Read/Space Amplification



Compaction Monitoring



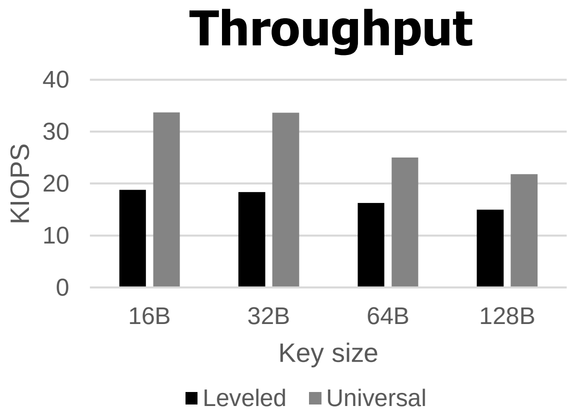
Write Latency

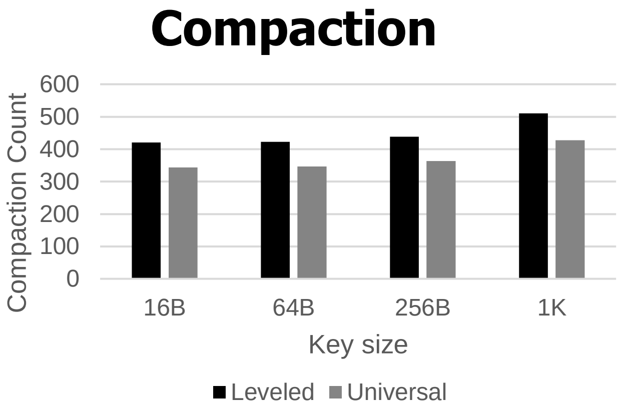


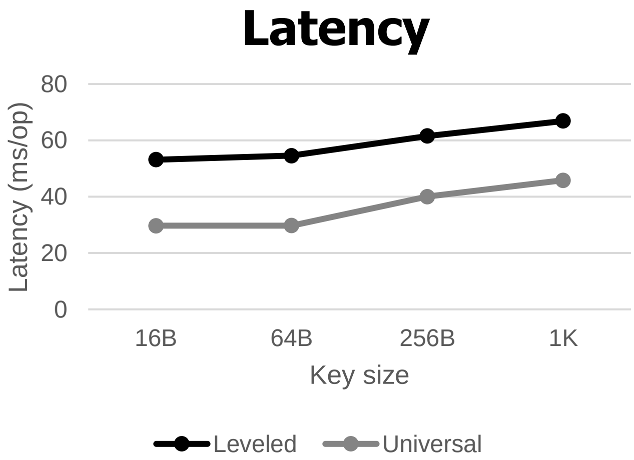
Write Amplification

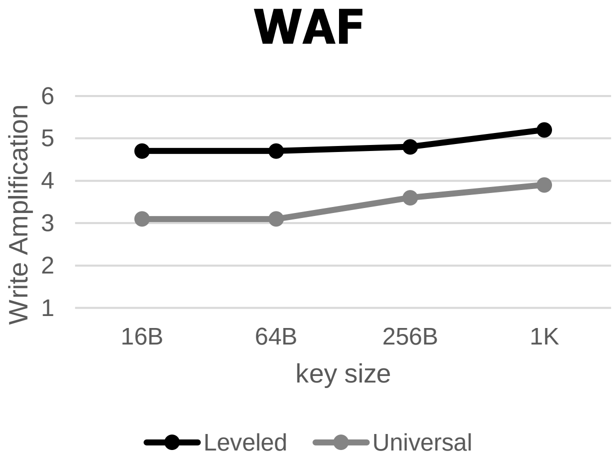
RocksDB Festival

- RocksDB::Compaction
  - ✓ Trial#3 Compaction on various Key size (random write)


Throughput


Compaction

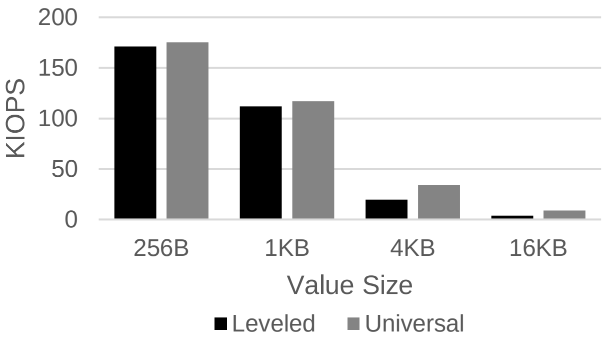

Latency


WAF

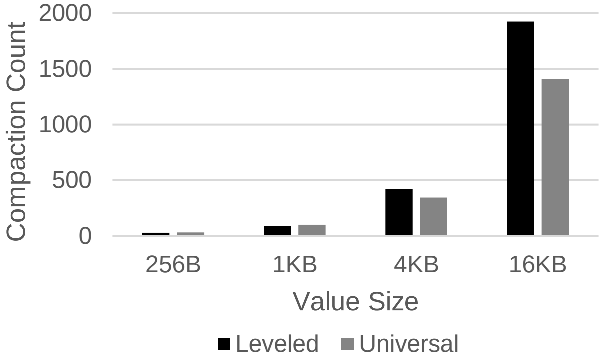
RocksDB Festival

- RocksDB::Compaction
  - ✓ Trial#4 Compaction on various Value size (random write)

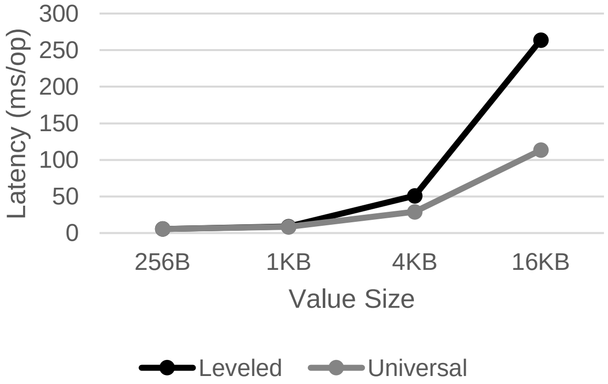
Throughput



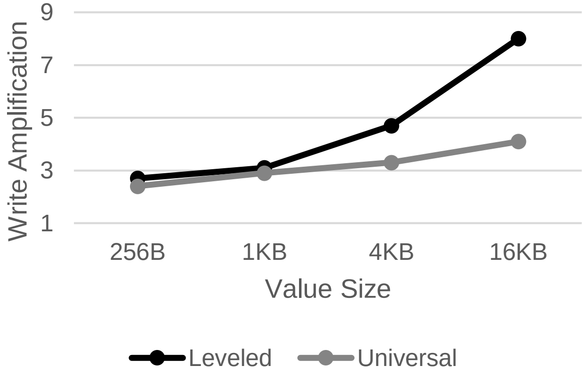
Compaction



Latency

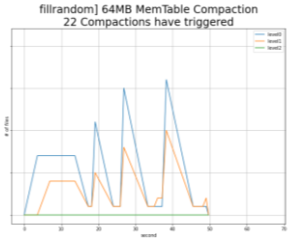


WAF

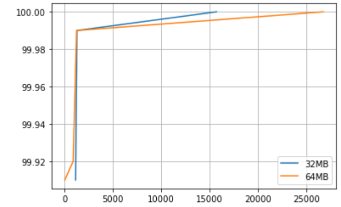


Compaction에 영향을 미치는 녀석들

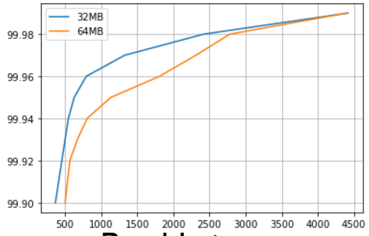
- ✓ #2 MemTable, SSTable
  - Various MemTable + Various SST
    - 64MB, 32MB
    - fillrandom, readrandom, 16-512, 10000000
  - Various MemTable + 64MB SST
    - 64MB, 32MB, ~~16MB, 8MB, 4MB, 2MB~~
    - fillrandom, readrandom, 16-512, 10000000



Compaction Monitoring



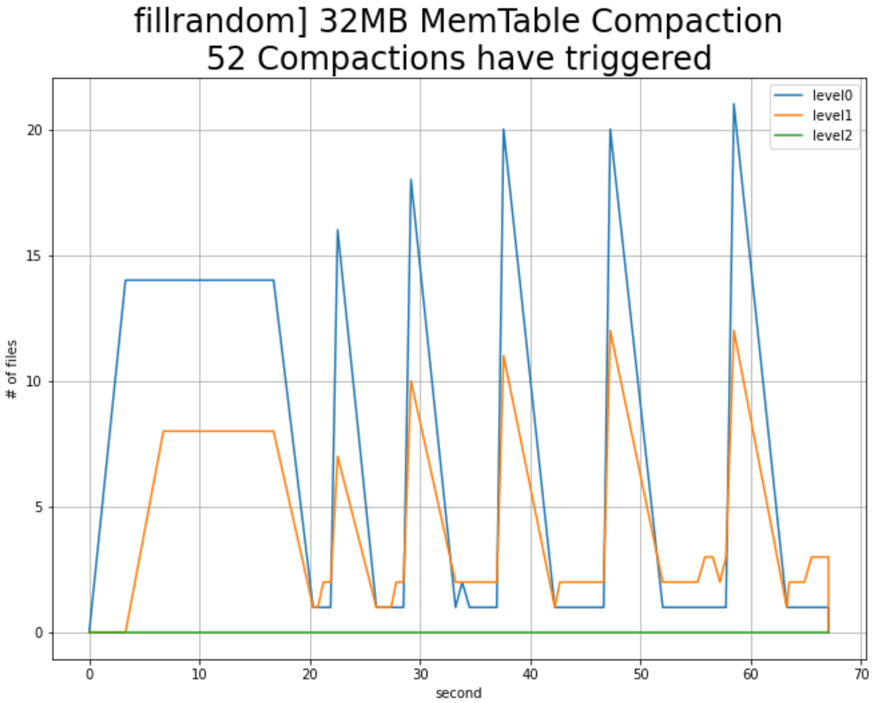
Write Latency

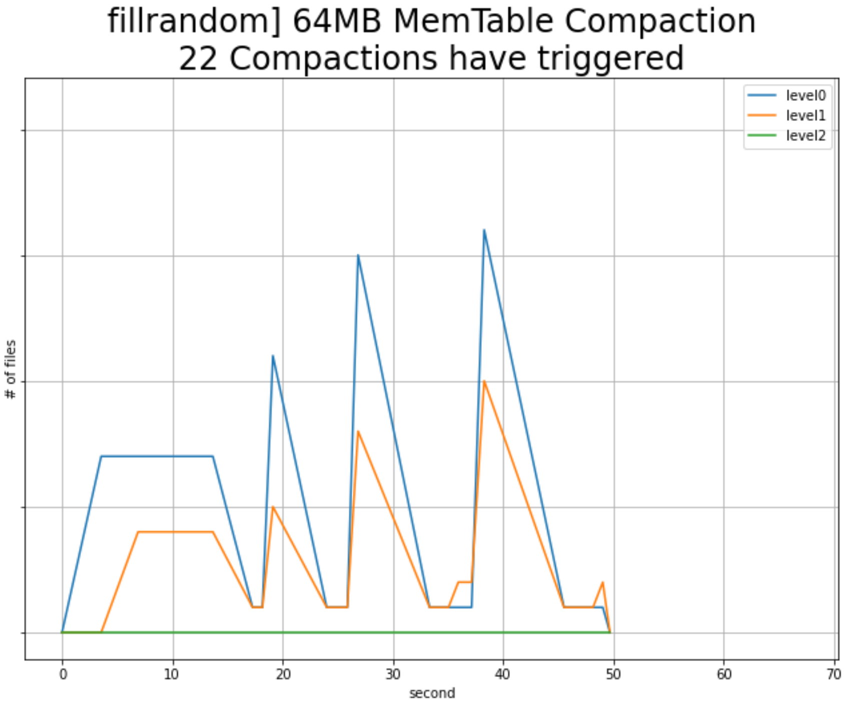


Read Latency

RocksDB Festival

- Compaction::SSTable
  - ✓ Trial#1 Compaction on MemTable size&Target File Size (32MB vs 64MB)

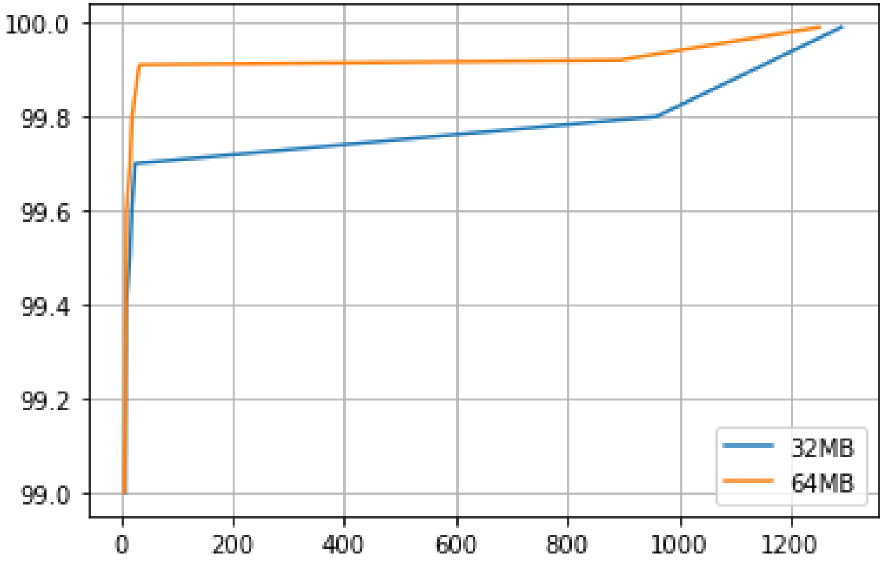




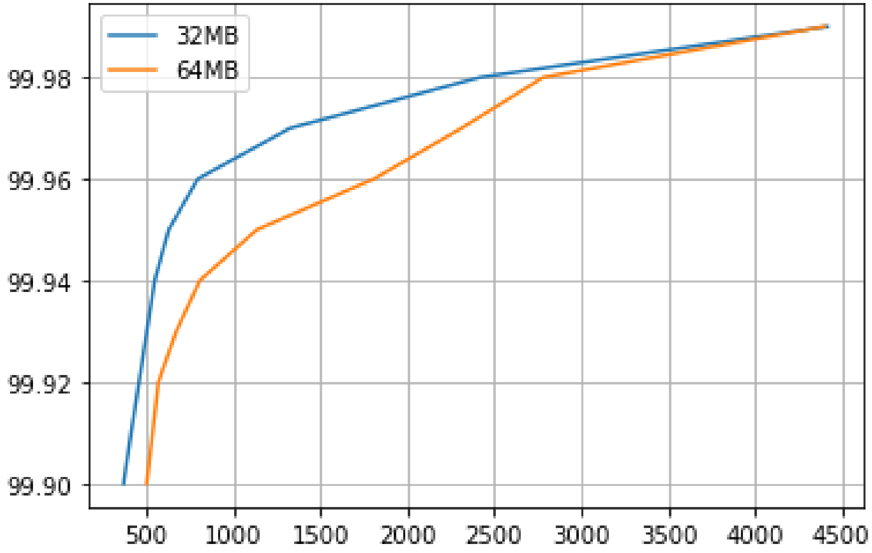
☞

RocksDB Festival

- ■ Compaction::SSTable
  - ✓ Trial#1 Compaction on MemTable size&Target File Size (32MB vs 64MB)



fillrandom] Write Latency (99%)

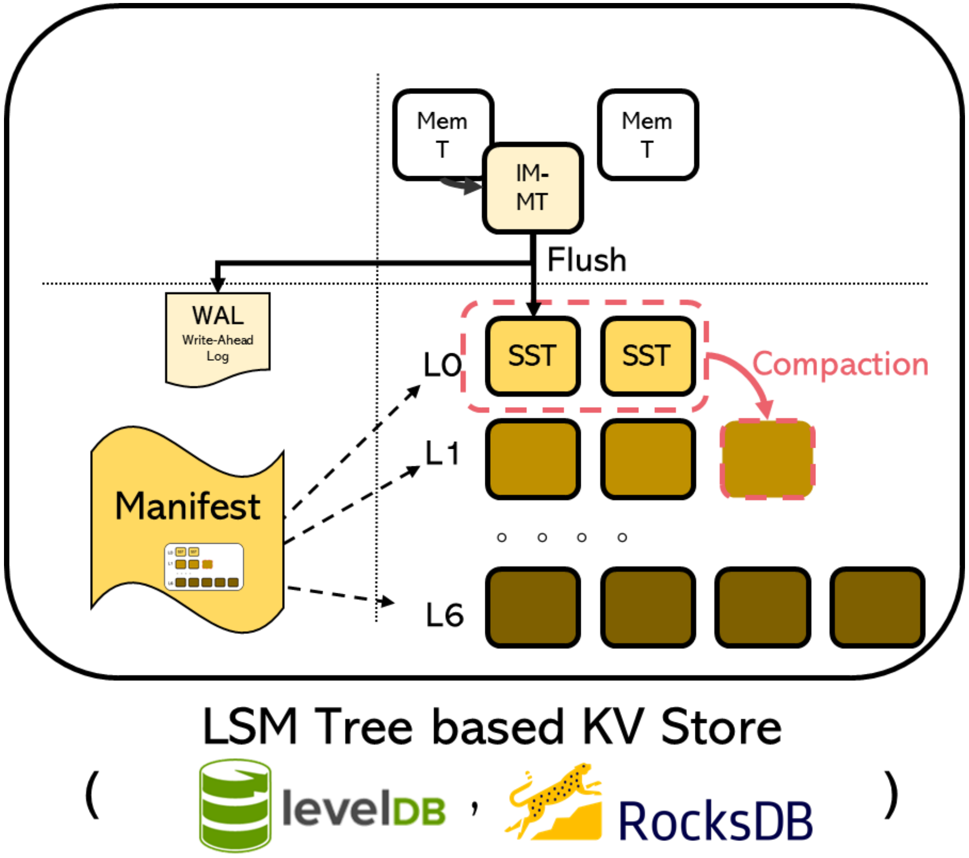


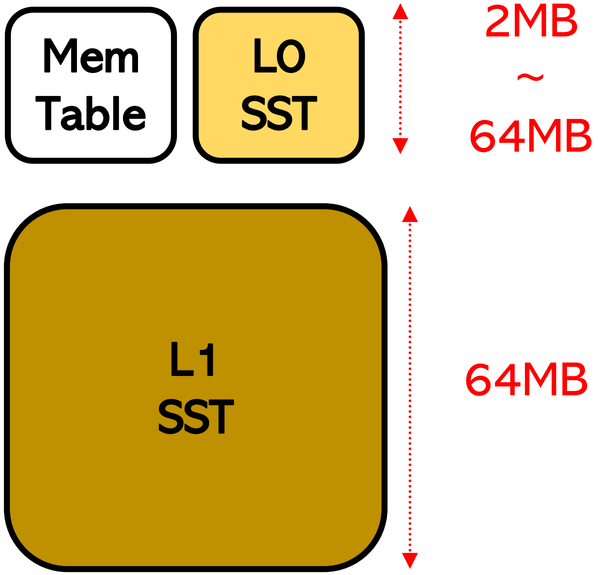
readrandom] Read Latency (99%)

☞ **Read/Write latency Trade-off on MemTable Size**

RocksDB Festival

- Compaction::SSTable
  - ✓ Trial#2 Compaction on MemTable size, but Target File Size 64MB (MemT=[2,4,8,16,32,64]MB, SST_Level1 = 64MB)



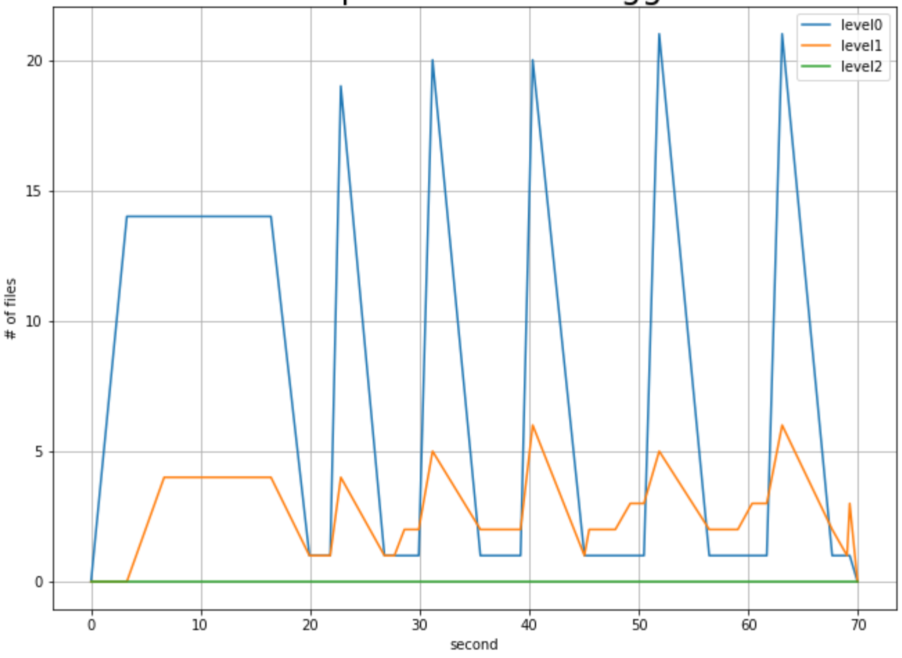


☞

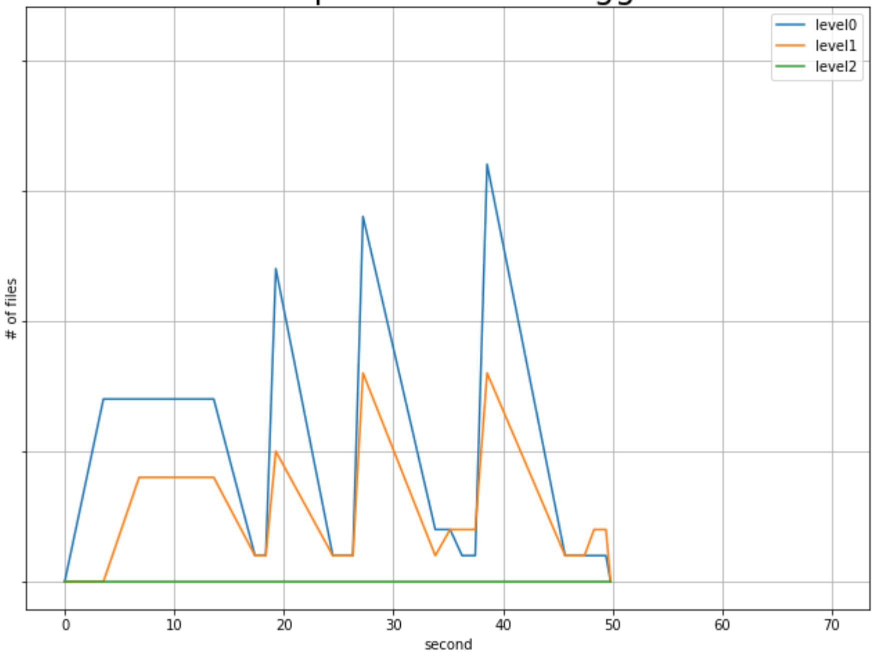
RocksDB Festival

- Compaction::SSTable
  - ✓ Trial#2 Compaction on MemTable size, but Target File Size 64MB (MemT=[2,4,8,16,32,64]MB, SST_Level1 = 64MB)

fillrandom] M:32MB SST_lvl1:64MB Compaction
36 Compactions have triggered



fillrandom] M:64MB SST_lvl1:64MB Compaction
23 Compactions have triggered

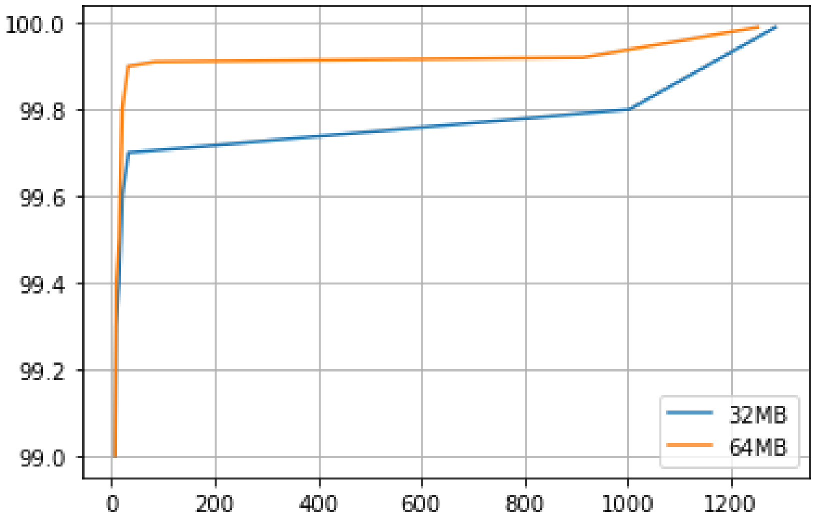


☞ **No difference between previous experiment**

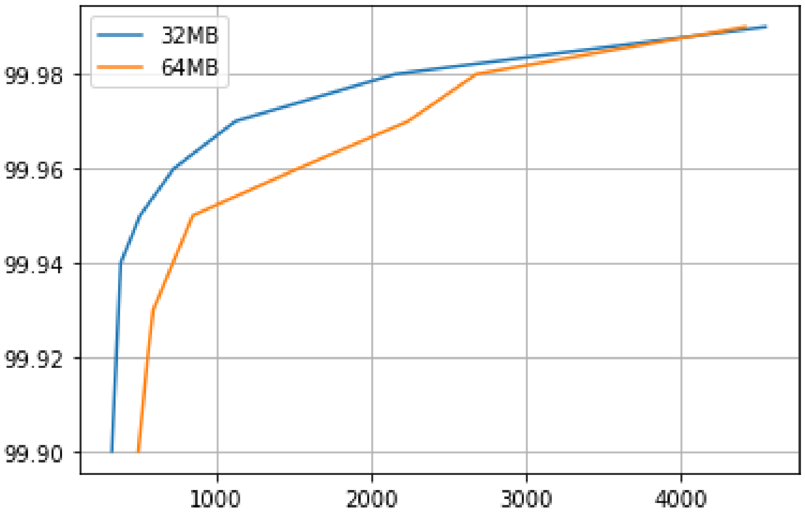
RocksDB Festival

■ Compaction::SSTable

- ✓ Trial#2 Compaction on MemTable size&Target File Size (32MB vs 64MB)



fillrandom] Write Latency (99%)

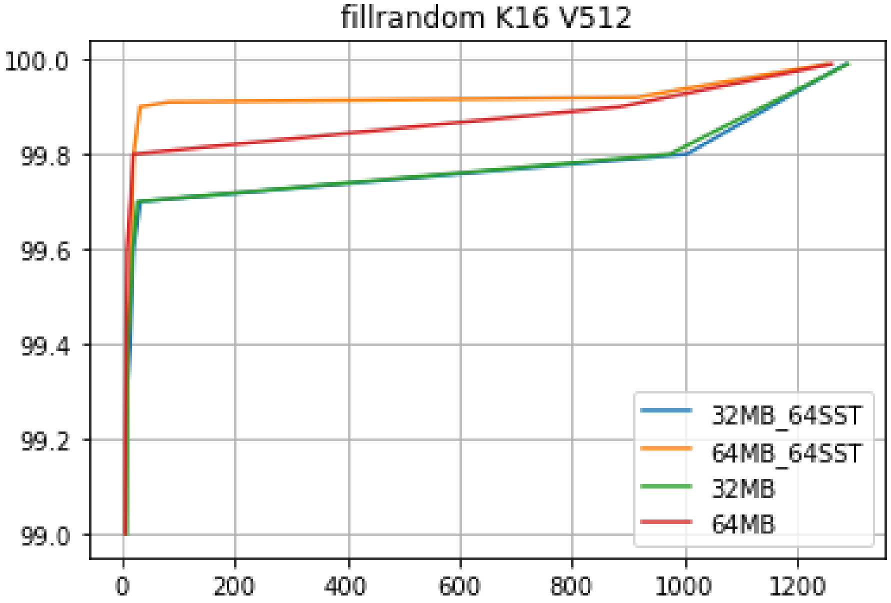


readrandom] Read Latency (99%)

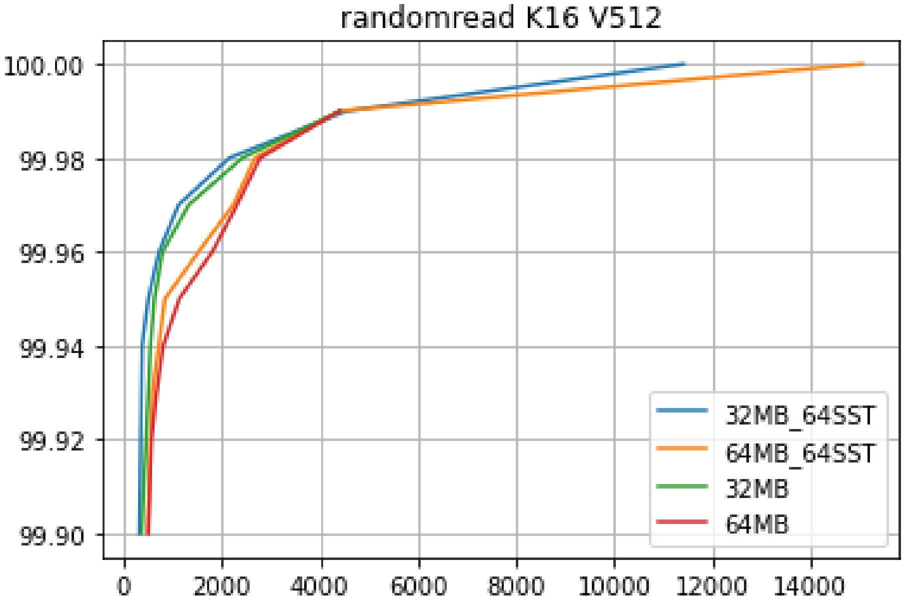
☞ **No difference between previous experiment**

RocksDB Festival

- Compaction::SSTable
  - ✓ Trial#1 vs Trial#2



fillrandom] Write Latency (99%)



readrandom] Read Latency (99%)

☞ ..? 😇