

# RocksDB Festival

Supported by IITP, StarLab.

August 2, 2021

송인호, 한예진

[inhoinno@dankook.ac.kr](mailto:inhoinno@dankook.ac.kr) , [hbb97225@naver.com](mailto:hbb97225@naver.com)

TeamName : 멘탈모델을 만들고 싶어요

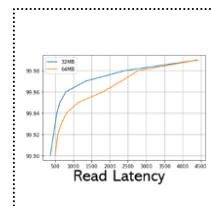
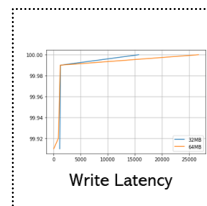
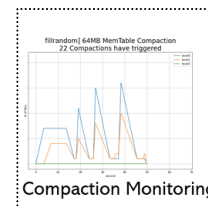
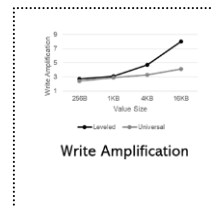
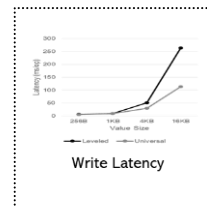
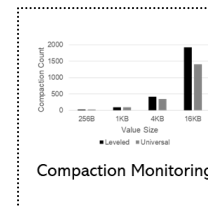
# Contents

## ■ Last Week

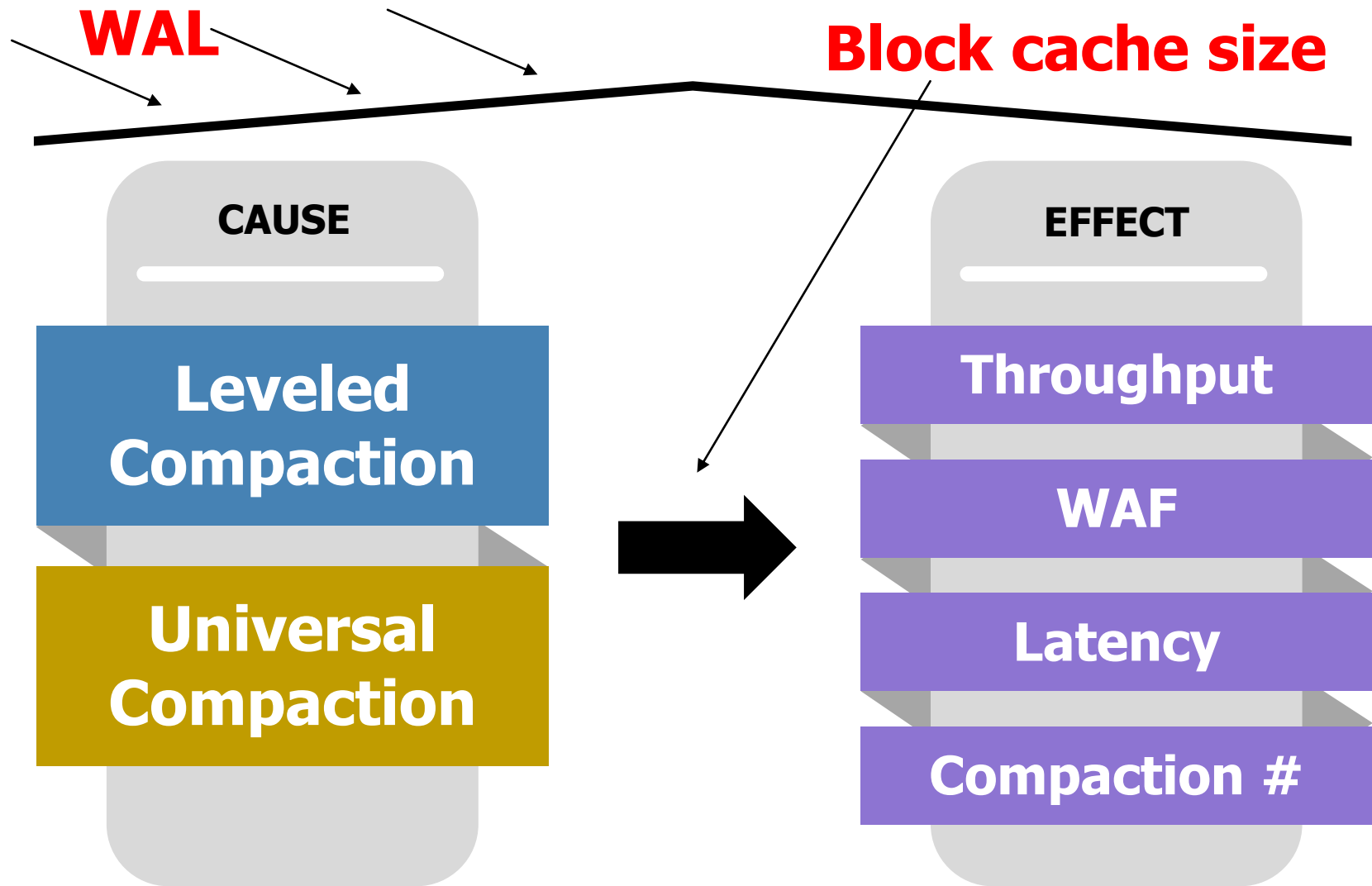
- ✓ Leveled vs Universal Compaction Comparison
  - Throughput
  - Write Amplification
  - Latency distribution + # of Compactions

## ■ This Week

- ✓ Leveled vs Universal Compaction Comparison
  - Block cache size – Hit ratio
  - Write-Ahead-Log – Throughput, Latency



# LVL vs Univ Compaction Comparison

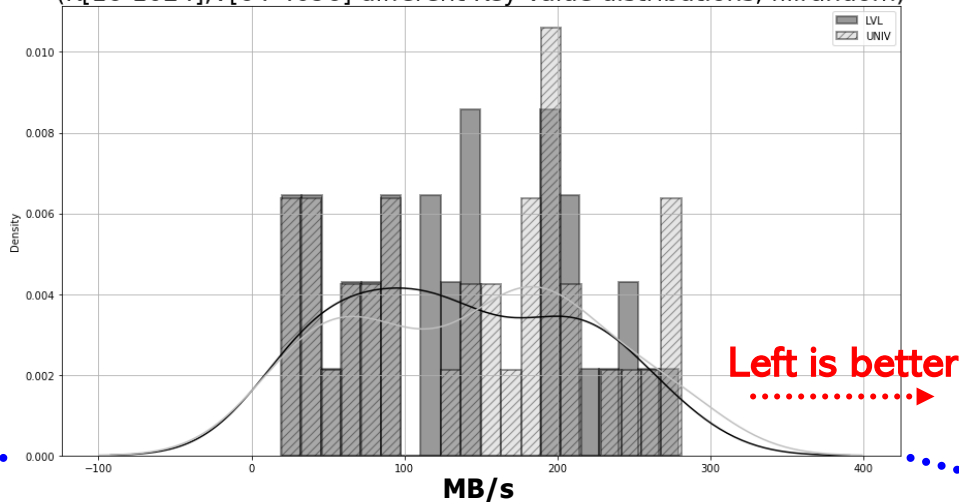


# LVL vs Univ Write Throughput: WAL\_OFF

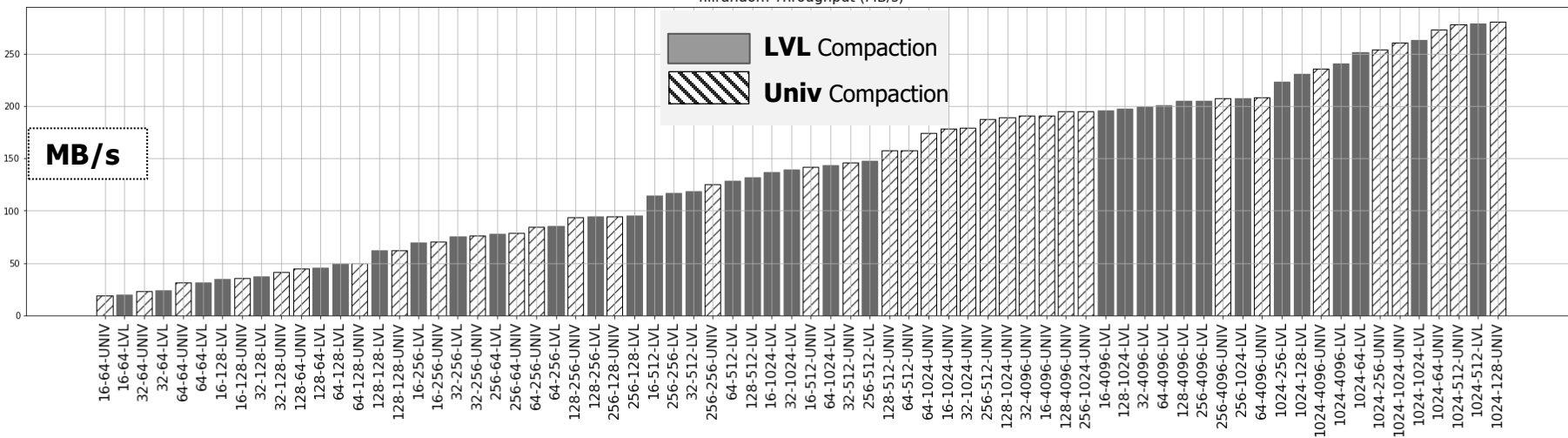
## Fillrandom

**Key** [16, 32, 64, 128, 256, 1024]  
**Value** [64, 128, 256, 512, 1024, 4096]  
**DB\_Size** 2.4GB  
**Storage** Samsung 512GB 860 Pro  
**File System** Ext4  
**CPU** Intel(R) Core(TM) i5-4440 CPU @ 3.10GHz

Throughput comparison  
 between different compaction Style  
 (K[16-1024],V[64-4096] different Key-Value distributions, fillrandom)

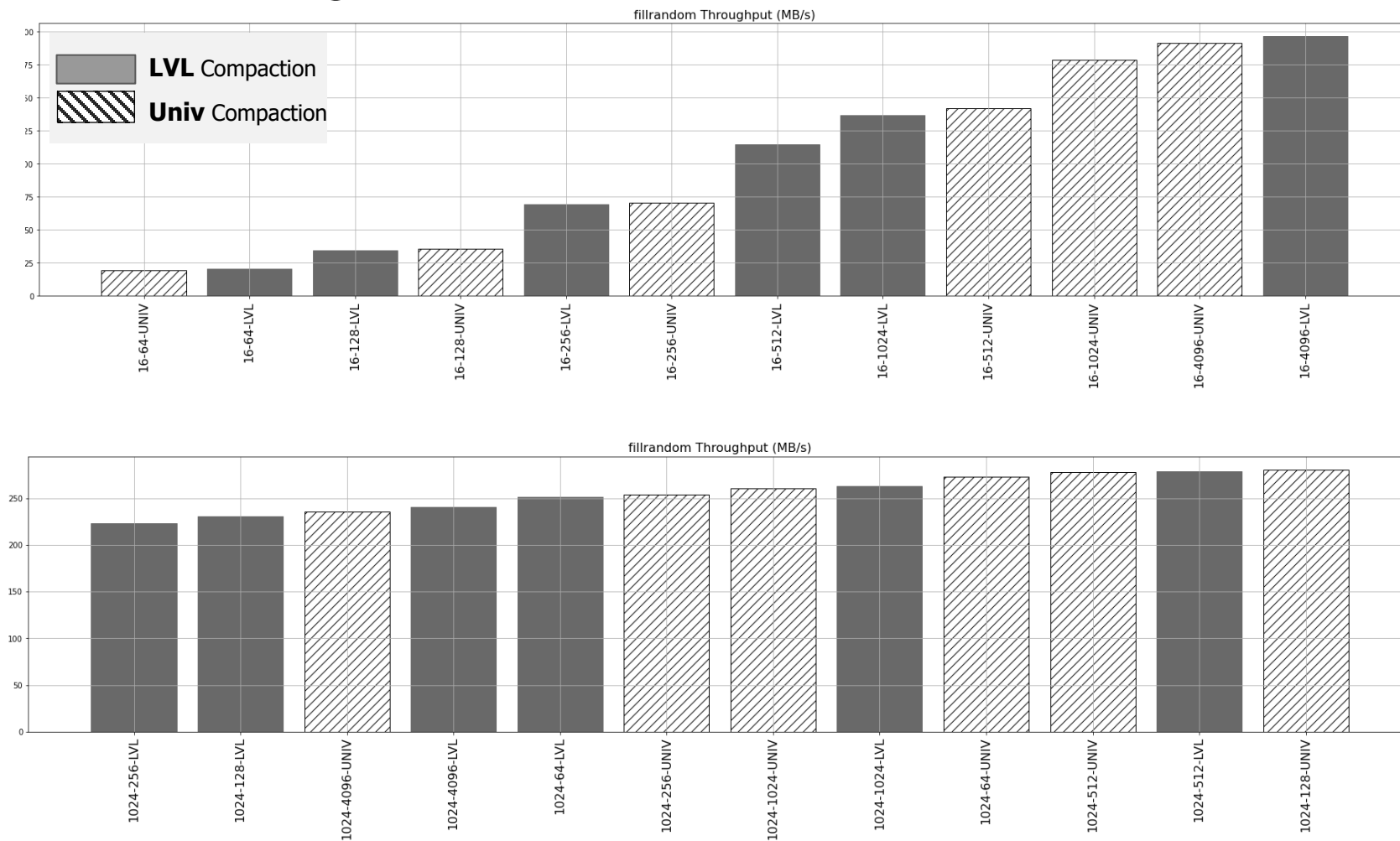


fillrandom Throughput (MB/s)



# LVL vs Univ Write Throughput: WAL\_OFF

## ■ Write Throughput: WAL\_OFF - K16, 1024 / V[64-4096]



# LVL vs Univ Read Throughput: WAL\_OFF

## Readrandom

--use\_existing\_db

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

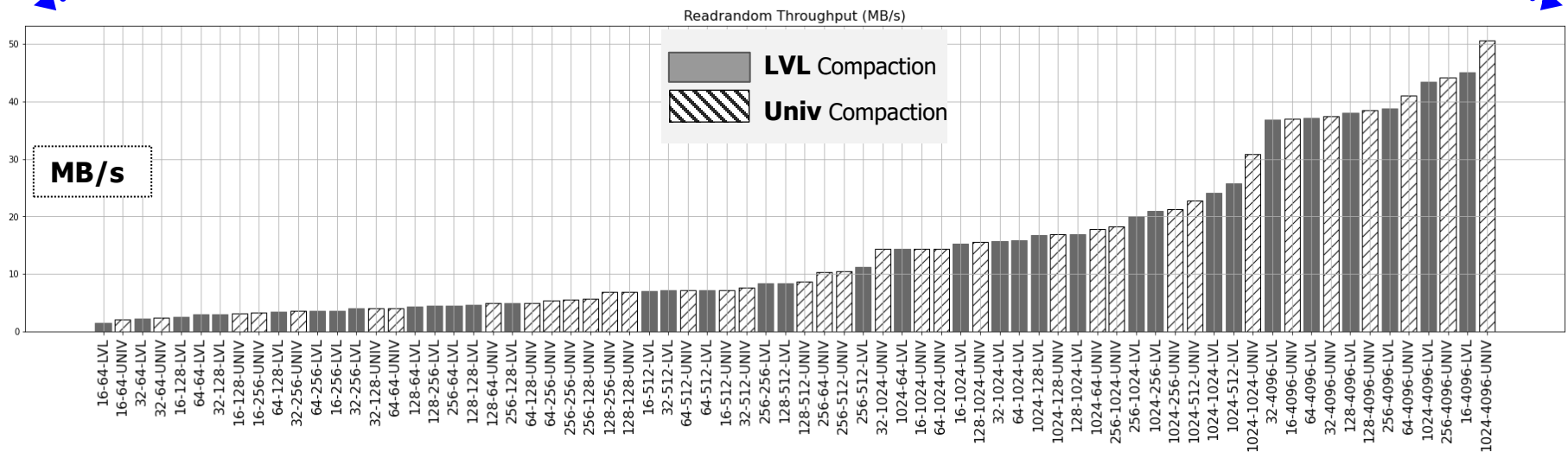
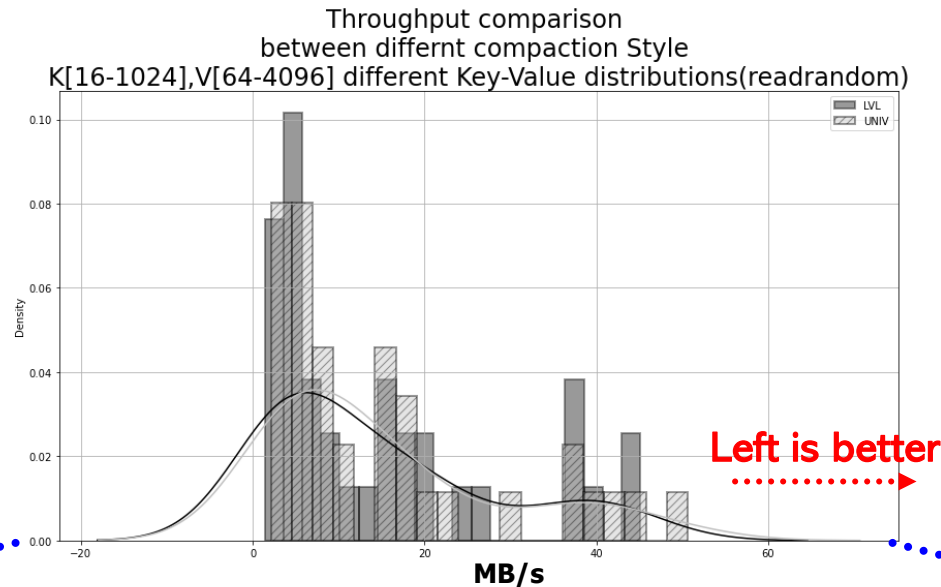
**Value** [64, 128, 256, 512, 1024, 4096]

**DB\_Size** 2.4GB

**Storage** Samsung 512GB 860 Pro

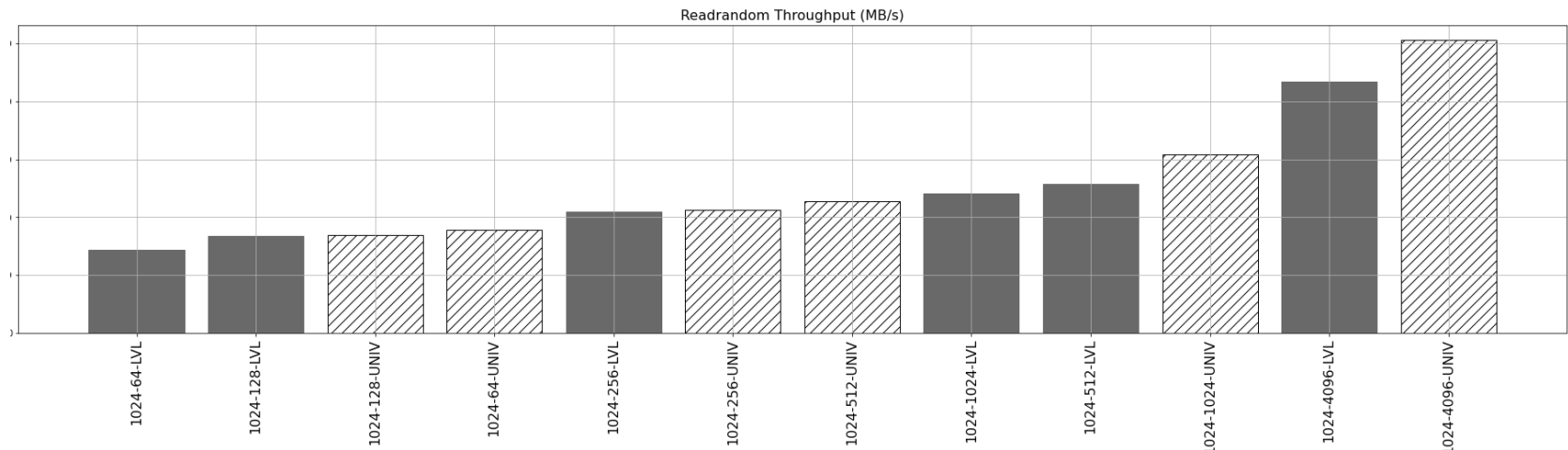
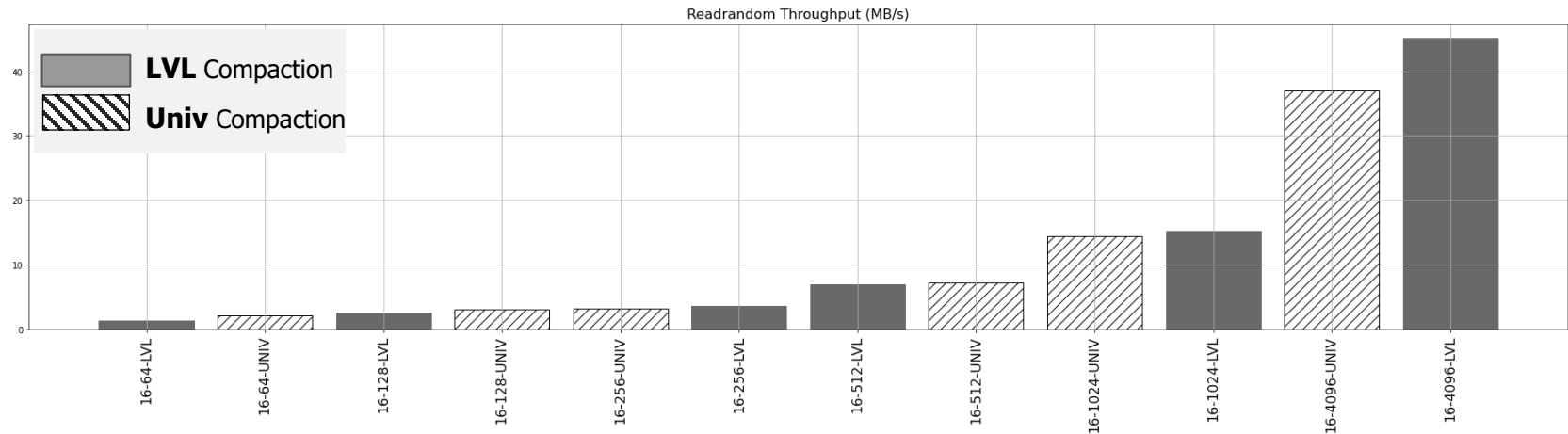
**File System** Ext4

**CPU** Intel(R) Core(TM) i5-4440 CPU @ 3.10GHz



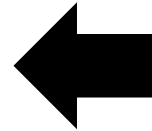
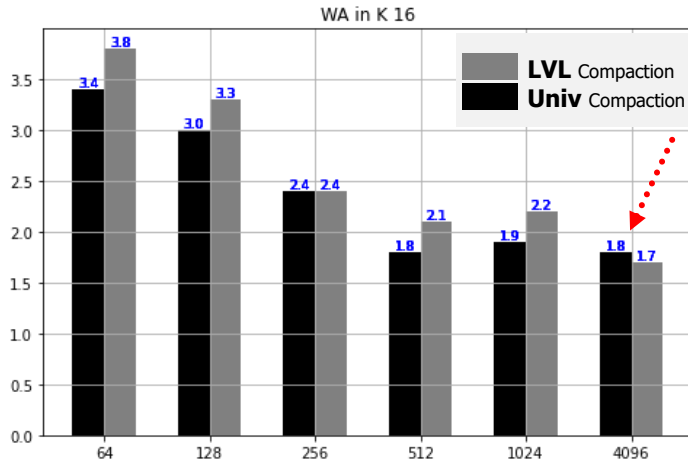
# LVL vs Univ Read Throughput: WAL\_OFF

## ■ Read Throughput: WAL\_OFF - K16, 1024 / V[64-4096]

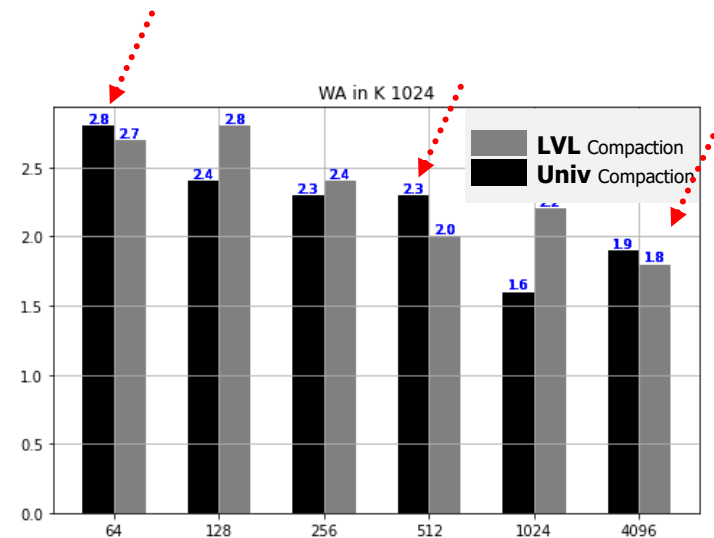
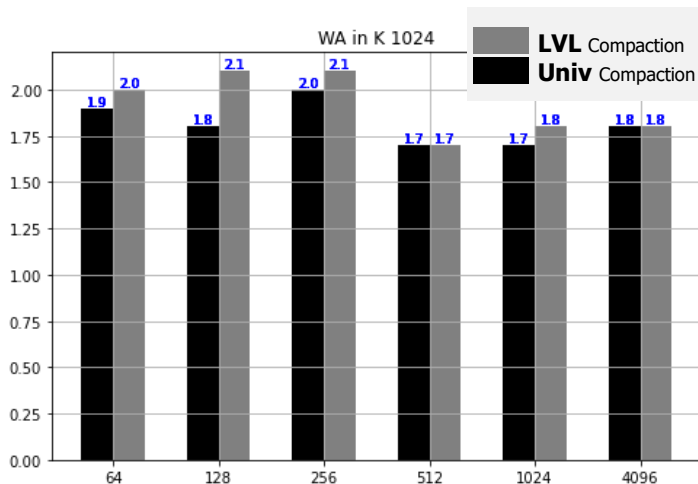
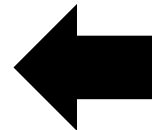
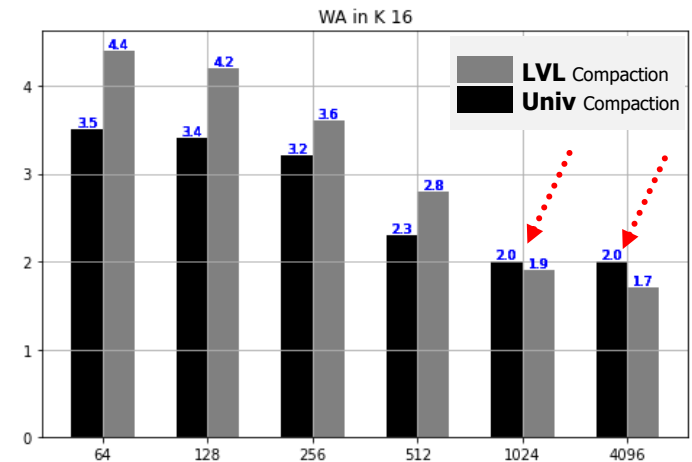


# LVL vs Univ WAF Comparison:WAL\_OFF

## ■ WAF: WAL\_OFF



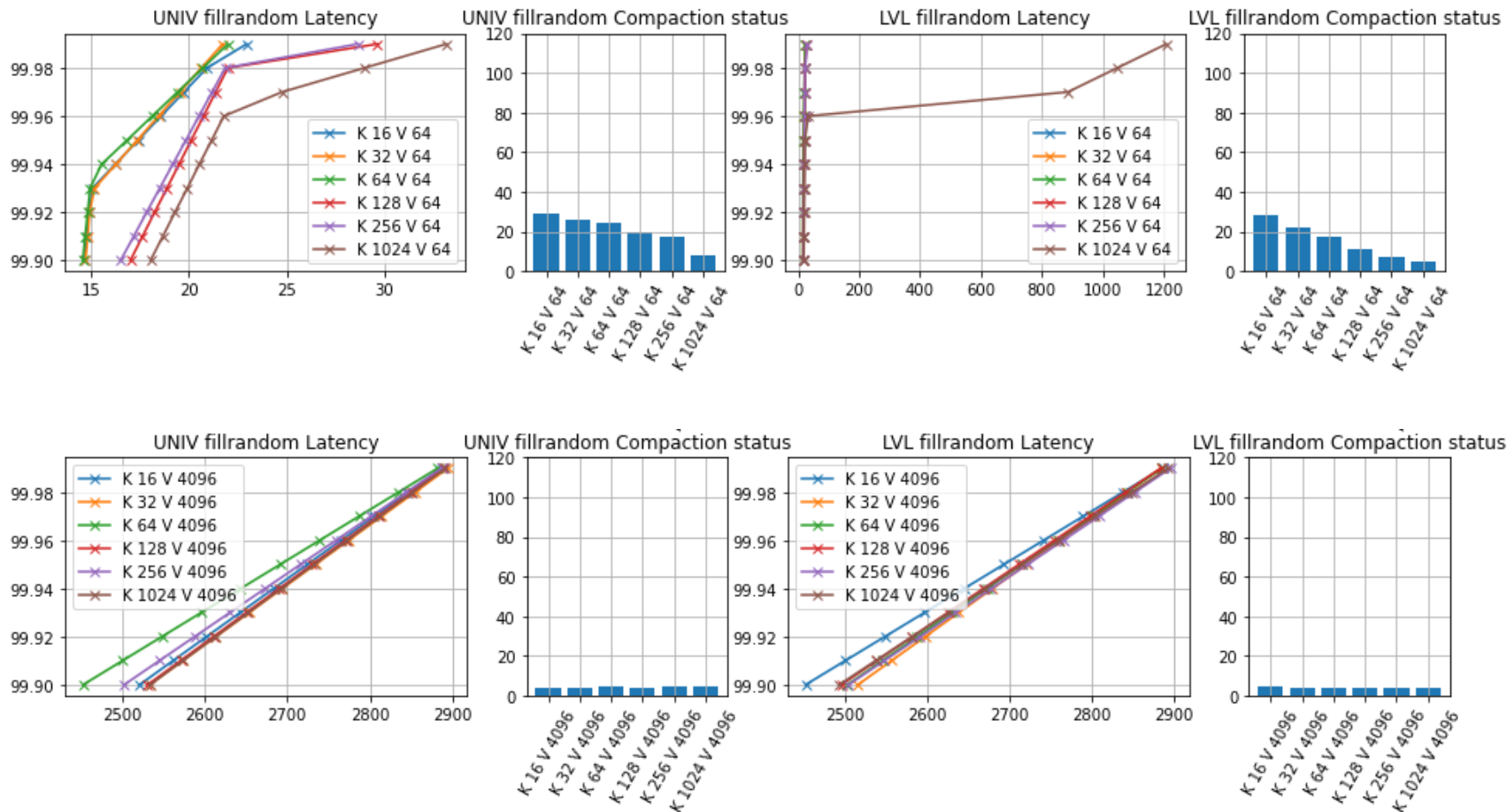
## ■ WAF: WAL\_ON





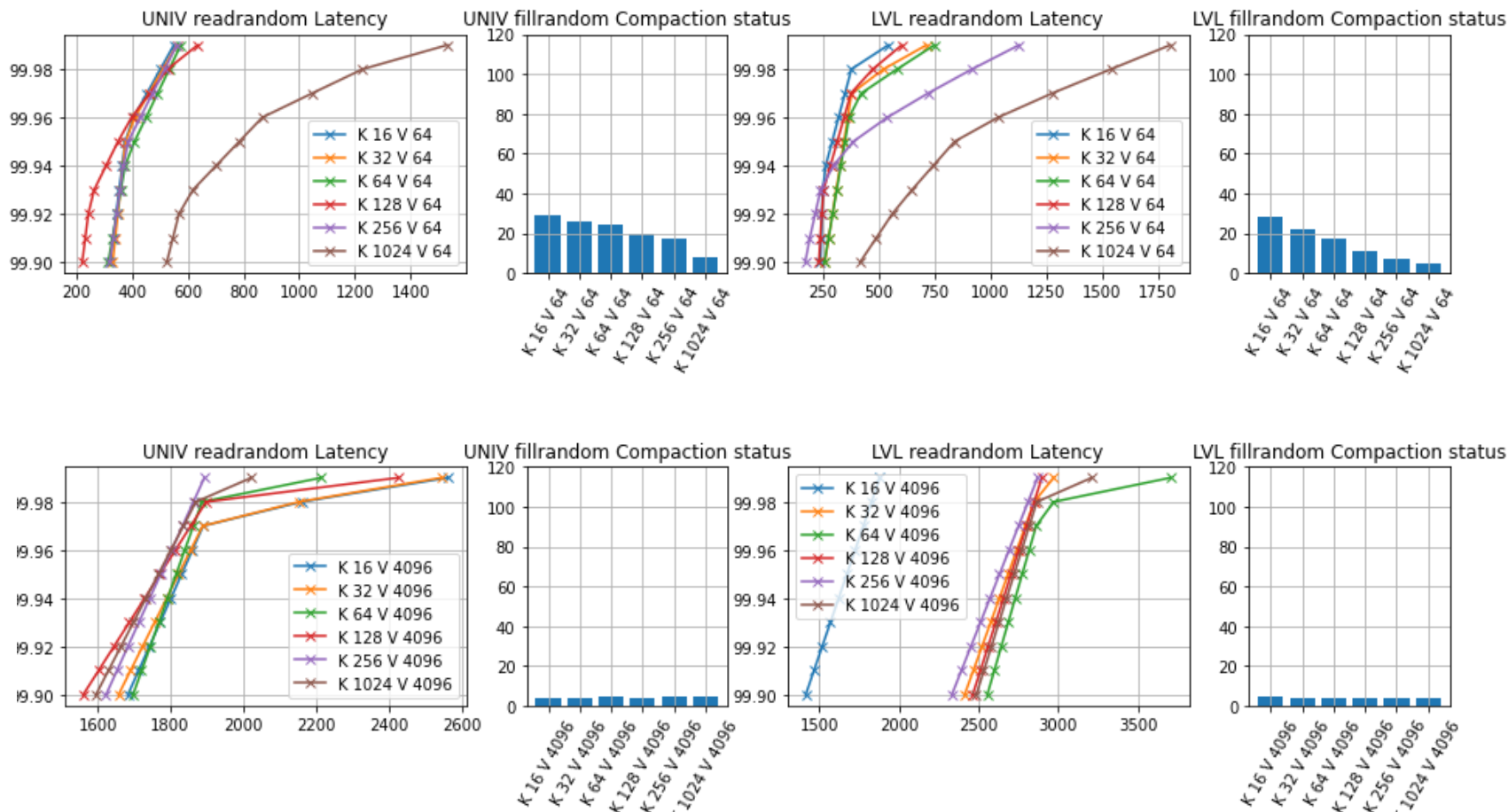
# LVL vs Univ # of Compactions , latency Comparison

## ■ Fillrandom latency 99.99%: WAL\_OFF



# LVL vs Univ # of Compactions , latency Comparison

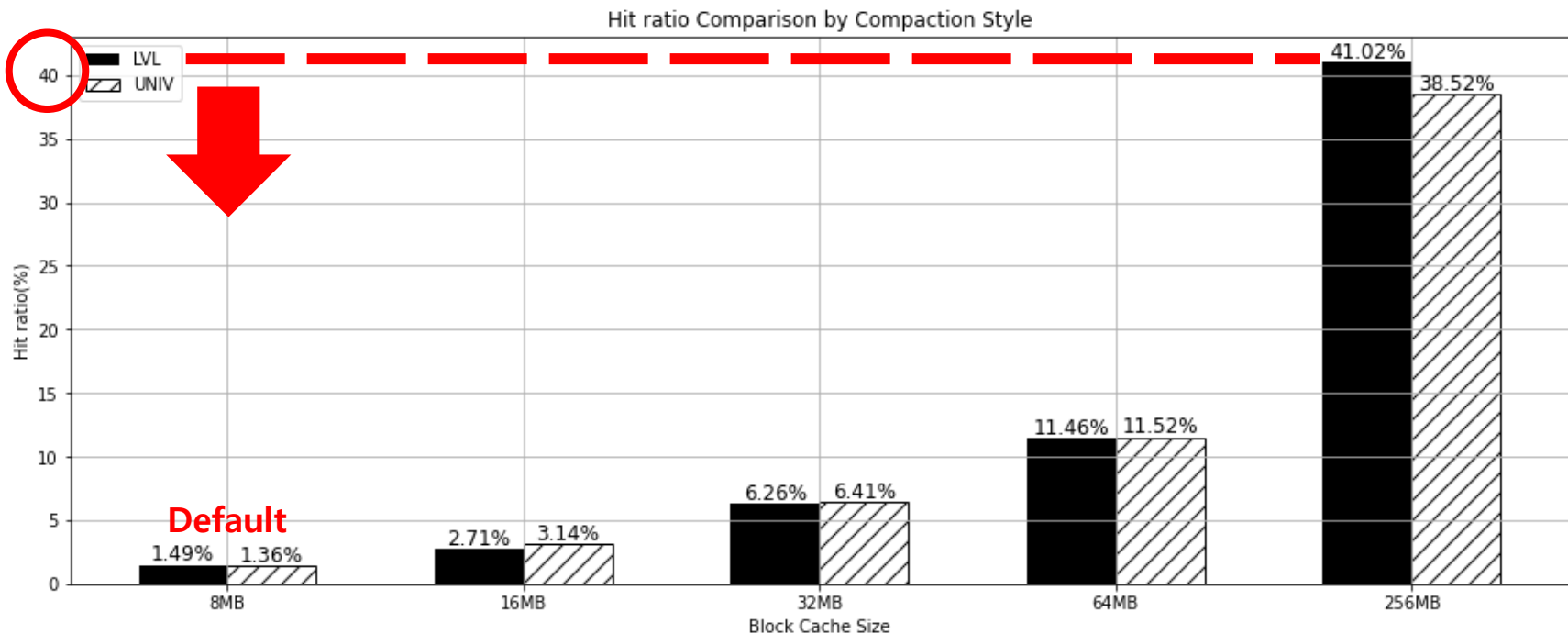
## ■ Readrandom latency 99.99%: WAL\_OFF



# LVL vs Univ Cache Hit ratio Comparison

## ■ Block Cache Hit ratio Comparison

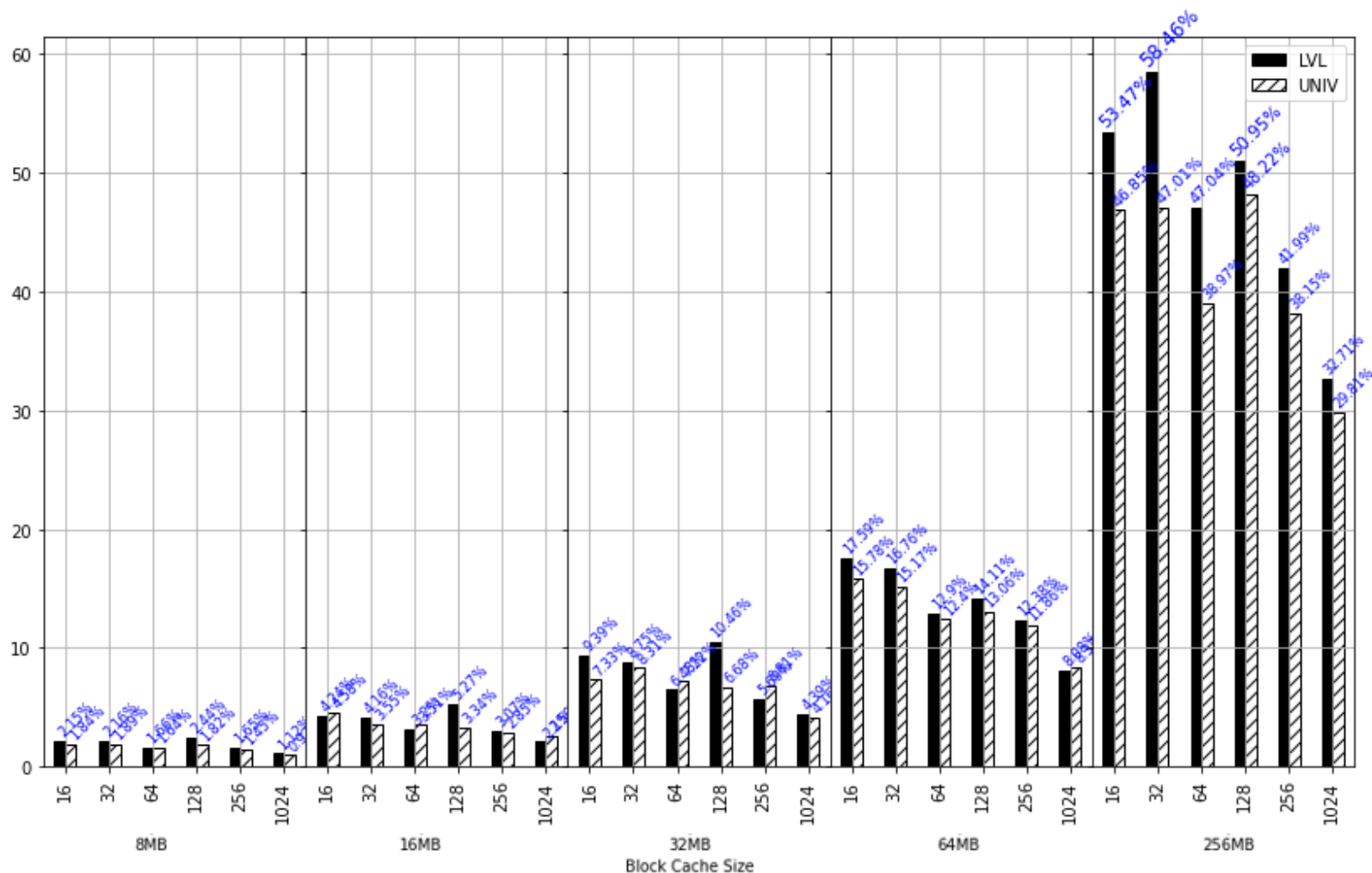
✓ 8MB, 16MB, 32MB, 64MB, 256MB → 512MB, 1GB, 2GB, 4GB 실험 중..



☞ Hit ratio is less than 50% under block cache size == 256MB

# LVL vs Univ Cache Hit ratio Comparison

## ■ Block Cache Hit ratio Comparison



# LVL vs Univ Cache Hit ratio Comparison

## Readrandom

--use\_existing\_db

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

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

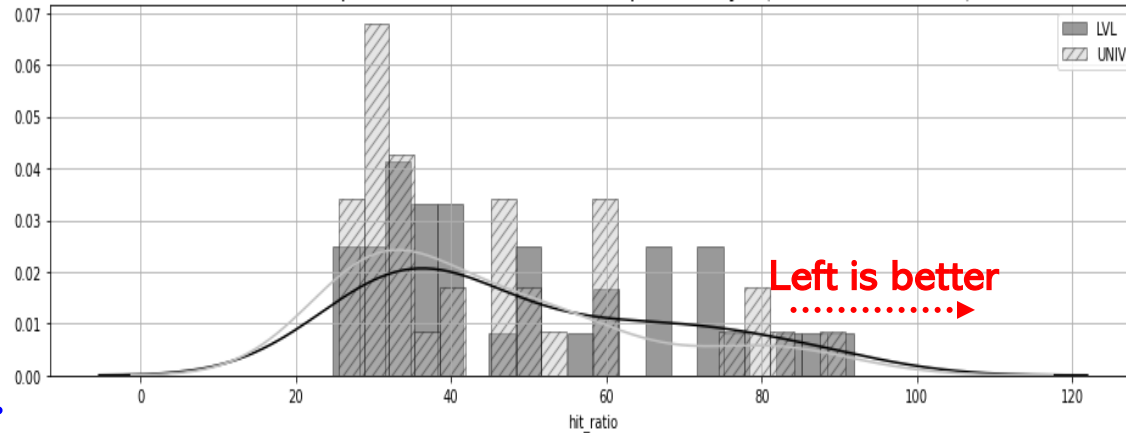
**Entries** 500 0000

**Storage** Samsung 512GB 860 Pro

**File System** Ext4

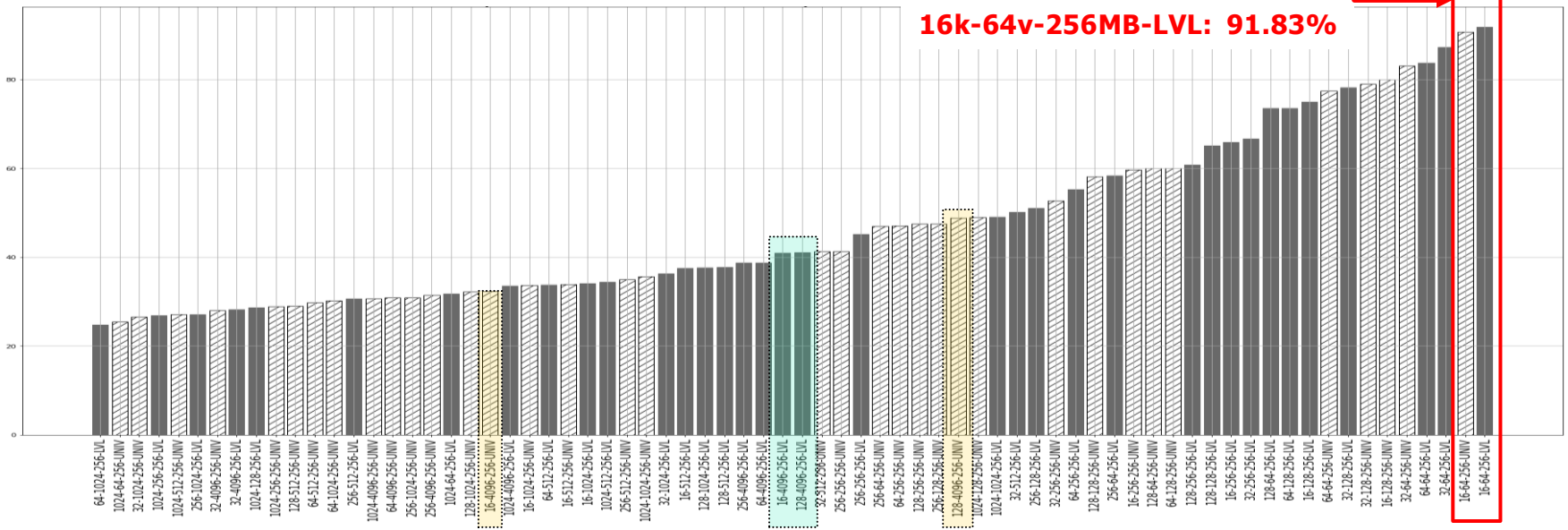
**CPU** Intel(R) Core(TM) i5-4440 CPU @ 3.10GHz

Hit ratio comparison between different compaction Style (Block Cache 256MB)



16k-64v-256MB-UNIV: 90.79%

16k-64v-256MB-LVL: 91.83%



# Mental Model

## ■ Quantative Experiment

- ✓ LvL vs Univ
  - KV distribution
    - Throughput, latency, Hit ratio, QPS(queries per second)
  - SST Table Size
    - Throughput, latency
  - WAL off
  - Adjusting block cache size

## ■ Qualitative Experiment

- ✓ Level Compaction's weak point
  - Write Amplification
  - Write Stall

→ Methods to overcome

**1~2 Week**

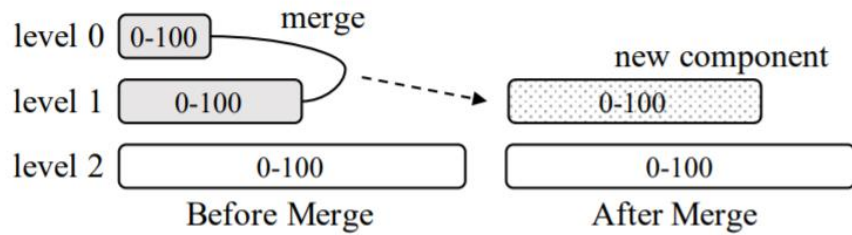
- ✓ Universal Compaction's weak point
  - Read Amplification
  - Space Amplification

→ Methods to overcome

**1~2 Week**

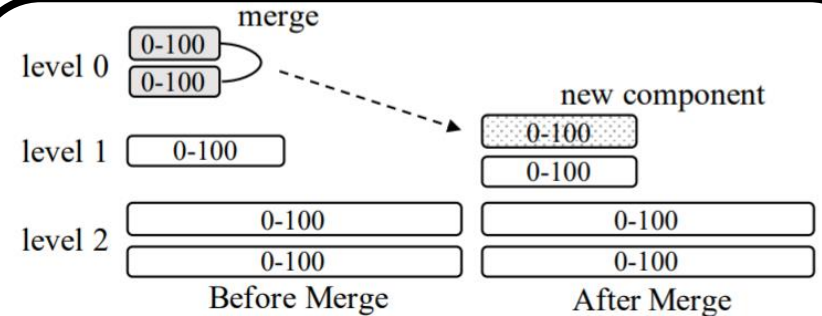
→ New Idea!

# Mental Model



(a) Leveling Merge Policy: one component per level

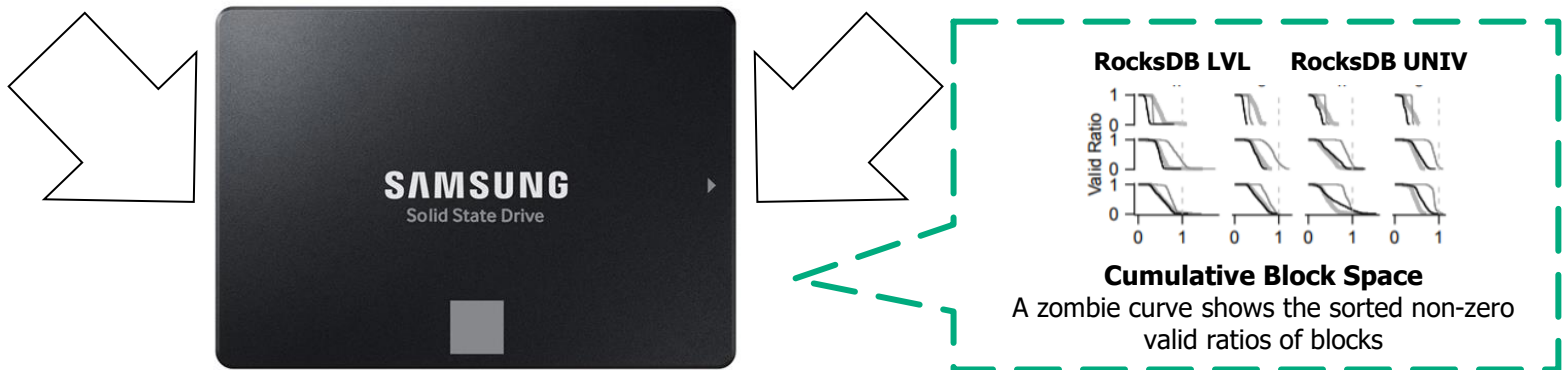
Level Compaction



(b) Tiering Merge Policy: up to T components per level

Universal Compaction

Fig: Luo, Chen, and Michael J. Carey. "LSM-based storage techniques: a survey." *The VLDB Journal* 29.1 (2020): 393-418.



Jun He, Sudarsun Kannan, Andrea C. Arpaci-Dusseau, Remzi H. Arpaci-Dusseau, The Unwritten Contract of Solid State Drives, EuroSys '17

<https://github.com/junhe/wiscsee>

# Discussion

---





---

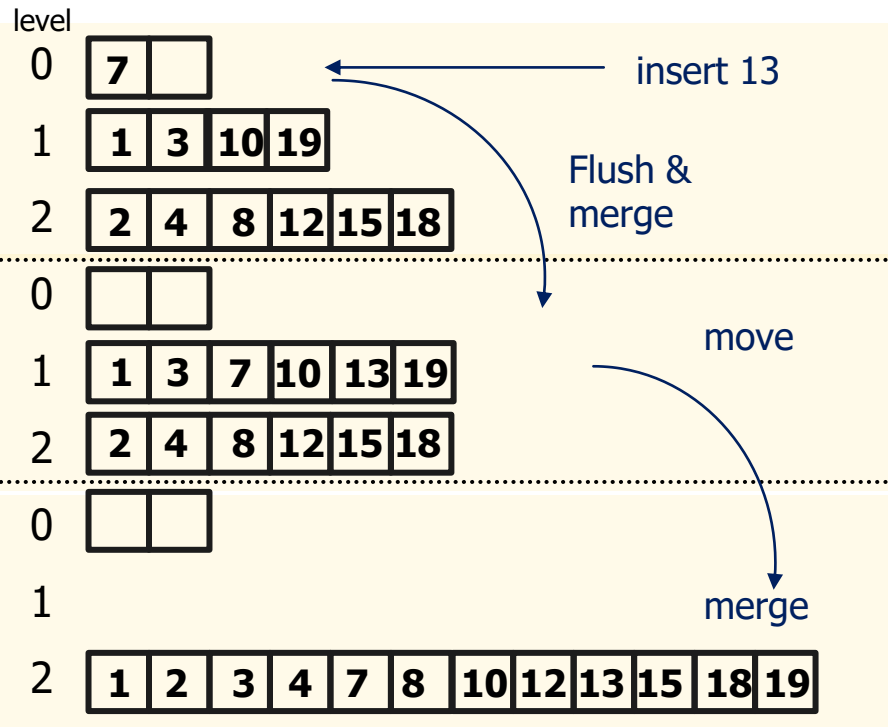
# Last Week

# Compaction Style

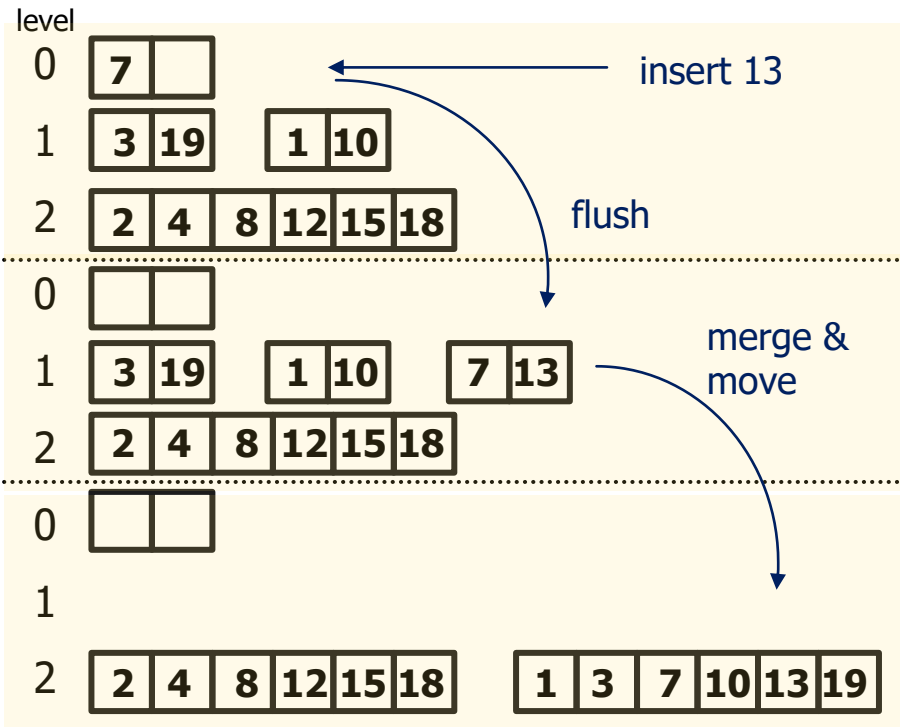
## ■ Leveled Compaction, Universal Compaction

✓ Sorted Level vs Sorted Run

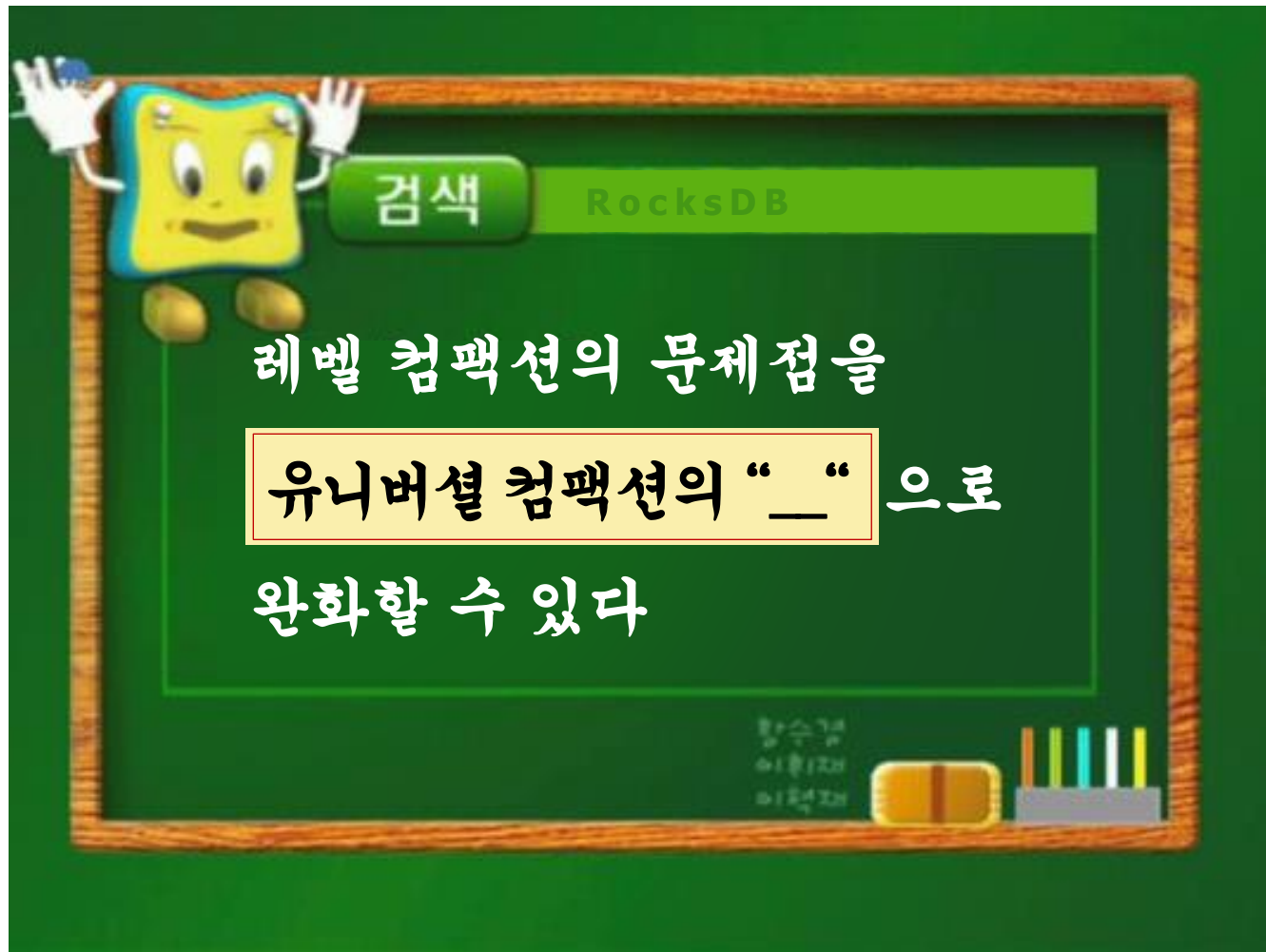
### Example of LeveledCompaction



### Example of Universal Compaction



# Mental Model



# LVL vs Univ Throughput Comparison

## Fillrandom

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

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

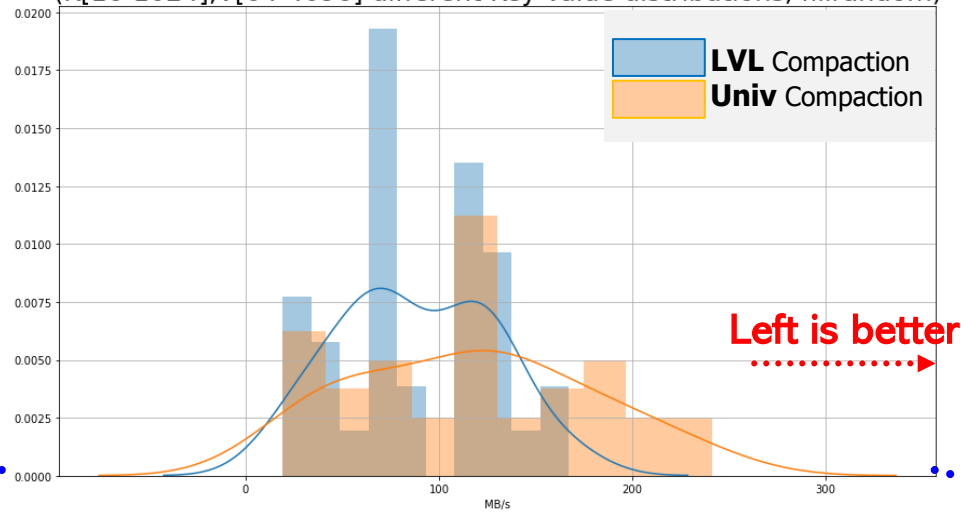
**Entries** 500 0000

**Storage** Samsung 1TB 860 Pro

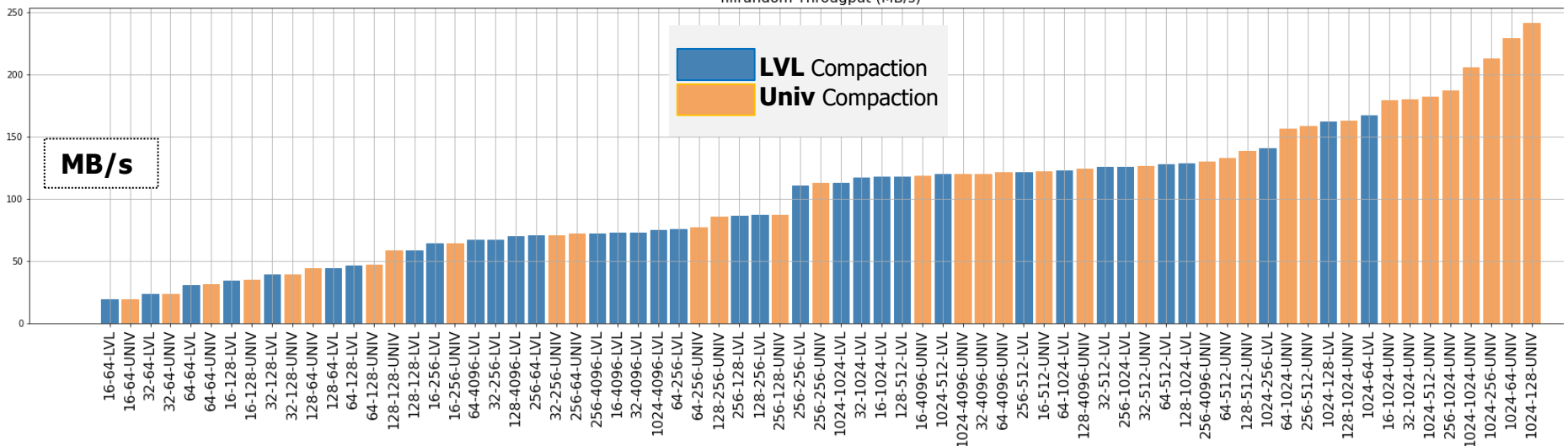
**File System** Ext4

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

Throughput comparison  
between different compaction style  
(K[16-1024],V[64-4096] different Key-Value distributions, fillrandom)



fillrandom Throughput (MB/s)



MB/s

LVL Compaction  
Univ Compaction

# LVL vs Univ Throughput Comparison

Throughput comparison  
between different compaction style

K[16-1024], V[64-4096] different Key-Value distributions(readrandom)

## Readrandom

--use\_existing\_db

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

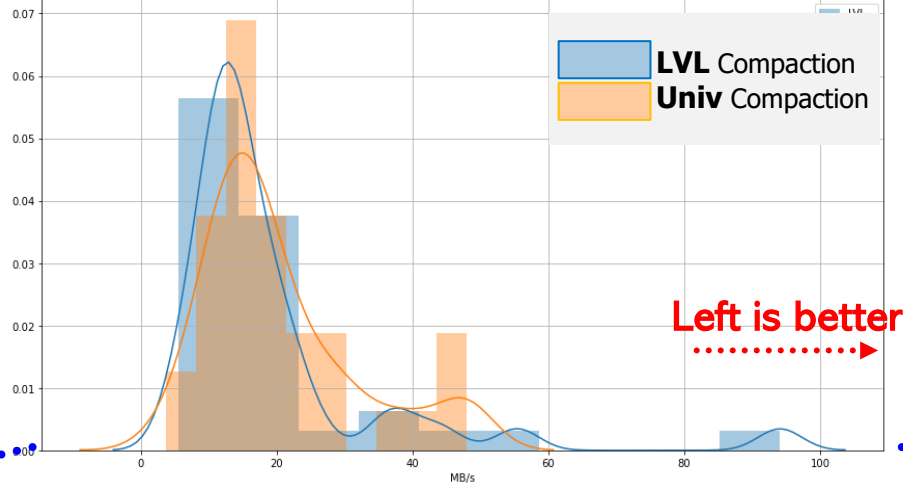
**Value** [64, 128, 256, 512, 1024, 4096]

**Entries** 500 0000

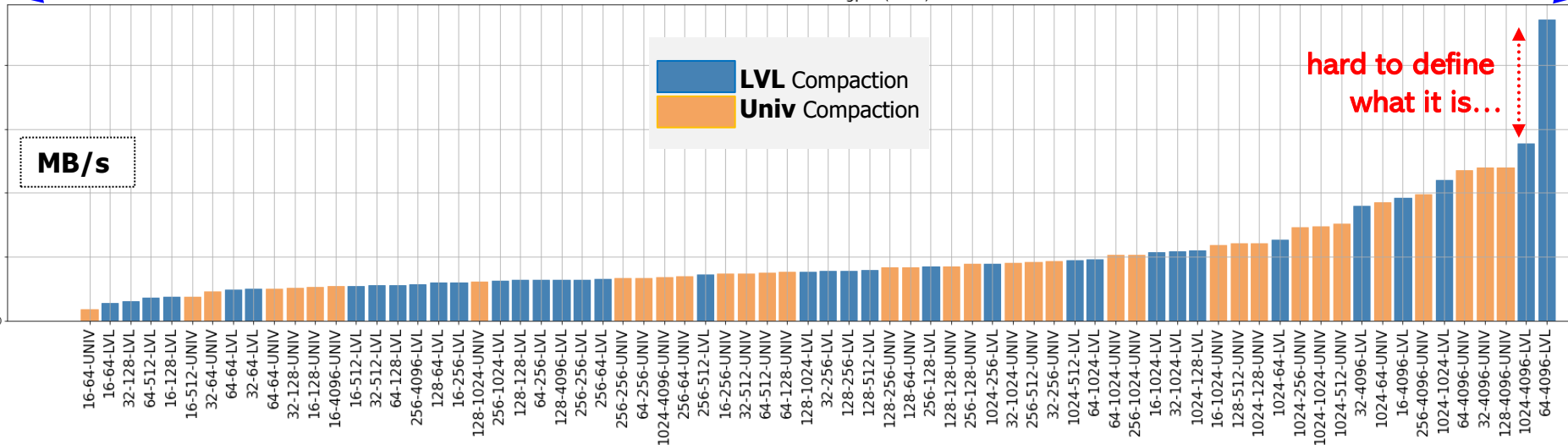
**Storage** Samsung 1TB 860 Pro

**File System** Ext4

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



Readrandom Throughput (MB/s)



MB/s

16-64-UNIV, 16-64-LVL, 32-128-UNIV, 32-128-LVL, 64-512-UNIV, 64-512-LVL, 16-512-UNIV, 16-512-LVL, 32-64-UNIV, 32-64-LVL, 64-64-UNIV, 64-64-LVL, 32-128-UNIV, 32-128-LVL, 16-128-UNIV, 16-128-LVL, 16-4096-UNIV, 16-4096-LVL, 32-512-UNIV, 32-512-LVL, 64-128-UNIV, 64-128-LVL, 256-4096-UNIV, 256-4096-LVL, 128-1024-UNIV, 128-1024-LVL, 256-1024-UNIV, 256-1024-LVL, 128-128-UNIV, 128-128-LVL, 64-256-UNIV, 64-256-LVL, 256-256-UNIV, 256-256-LVL, 1024-4096-UNIV, 1024-4096-LVL, 256-512-UNIV, 256-512-LVL, 16-256-UNIV, 16-256-LVL, 32-512-UNIV, 32-512-LVL, 64-512-UNIV, 64-512-LVL, 64-128-UNIV, 64-128-LVL, 128-1024-UNIV, 128-1024-LVL, 256-128-UNIV, 256-128-LVL, 128-256-UNIV, 128-256-LVL, 1024-256-UNIV, 1024-256-LVL, 32-256-UNIV, 32-256-LVL, 1024-512-UNIV, 1024-512-LVL, 64-1024-UNIV, 64-1024-LVL, 256-1024-UNIV, 256-1024-LVL, 16-1024-UNIV, 16-1024-LVL, 32-1024-UNIV, 32-1024-LVL, 16-1024-UNIV, 16-1024-LVL, 128-512-UNIV, 128-512-LVL, 1024-128-UNIV, 1024-128-LVL, 1024-64-UNIV, 1024-64-LVL, 1024-256-UNIV, 1024-256-LVL, 1024-1024-UNIV, 1024-1024-LVL, 32-4096-UNIV, 32-4096-LVL, 1024-64-UNIV, 1024-64-LVL, 16-4096-UNIV, 16-4096-LVL, 256-4096-UNIV, 256-4096-LVL, 1024-1024-UNIV, 1024-1024-LVL, 64-4096-UNIV, 64-4096-LVL, 32-4096-UNIV, 32-4096-LVL, 128-4096-UNIV, 128-4096-LVL, 1024-4096-UNIV, 1024-4096-LVL, 64-4096-UNIV, 64-4096-LVL

# LVL vs Univ Throughput Comparison

## Readrandom

--use\_existing\_db

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

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

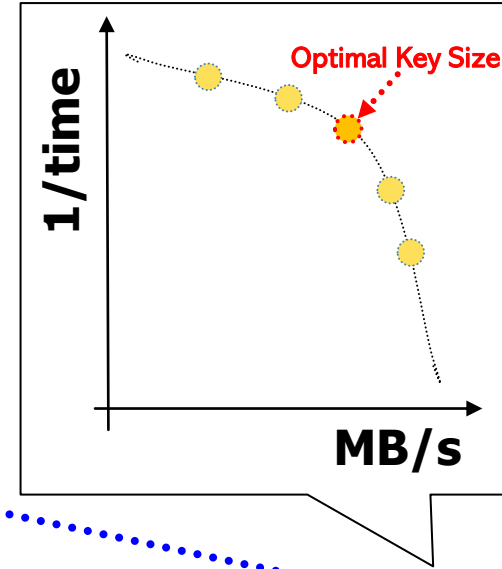
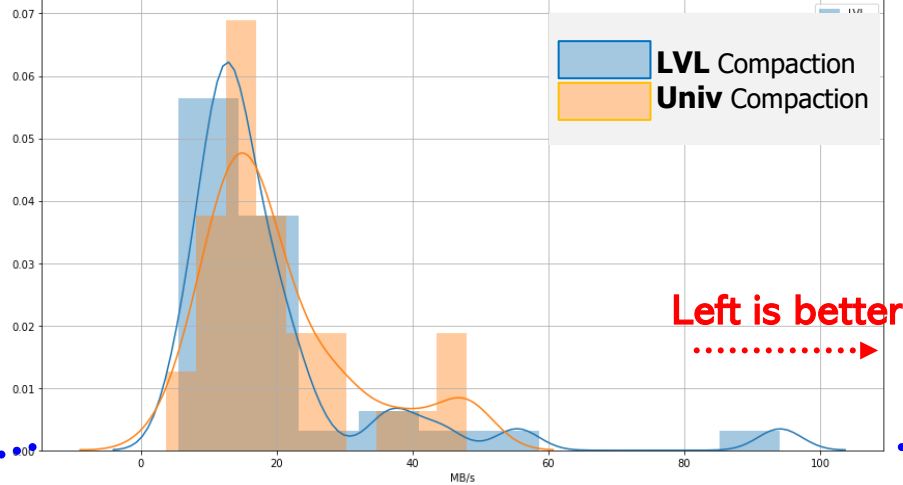
**Entries** 500 0000

**Storage** Samsung 1TB 860 Pro

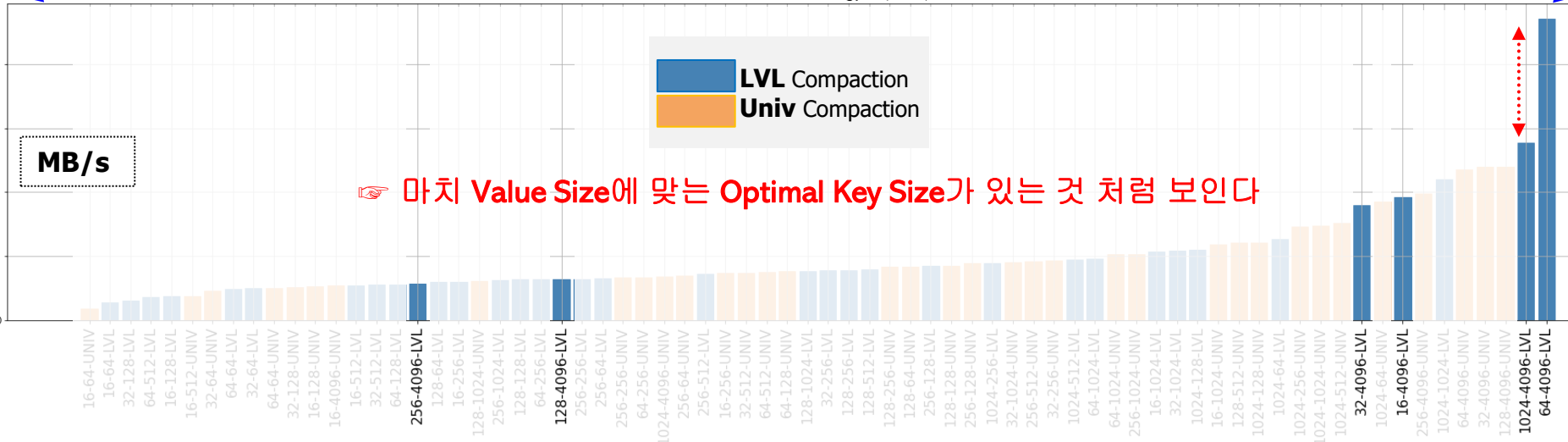
**File System** Ext4

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

Throughput comparison  
between different compaction style  
K[16-1024], V[64-4096] different Key-Value distributions(readrandom)

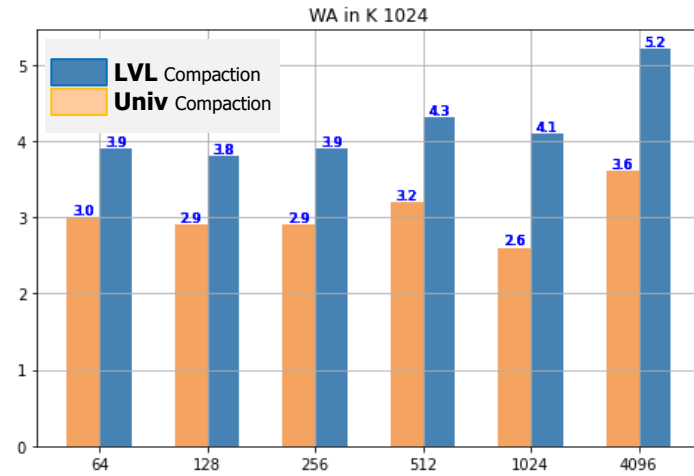
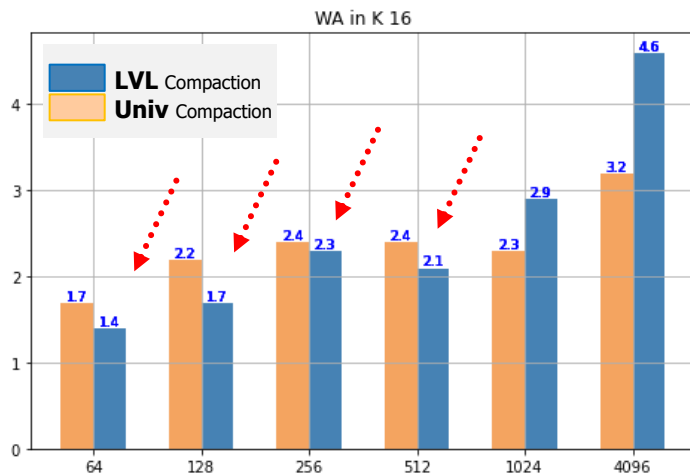
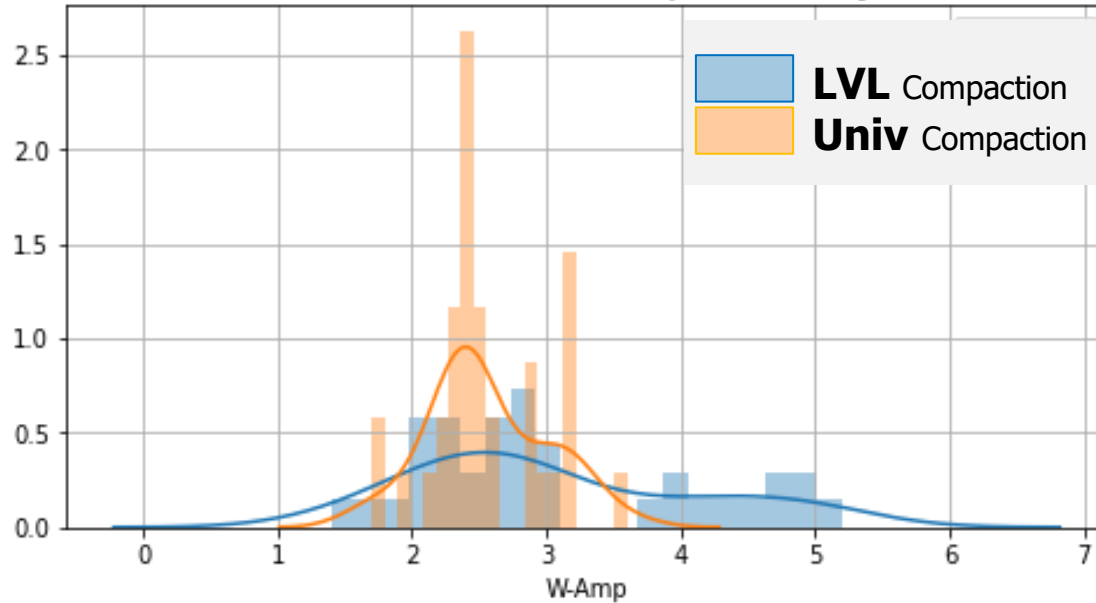


Readrandom Throughput (MB/s)



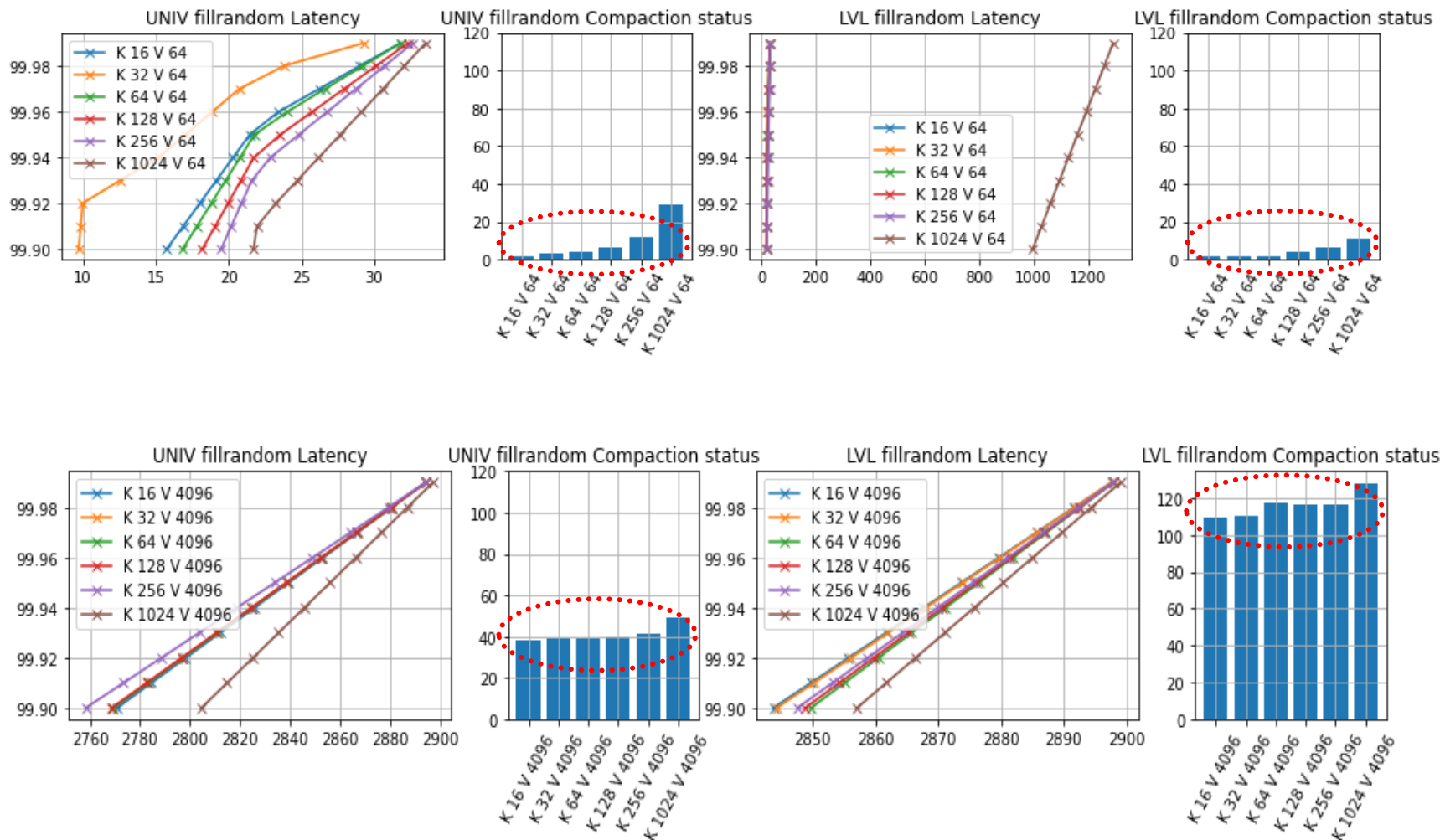
# LVL vs Univ WAF Comparison

Write Amplification comparison  
between different compaction Style



# LVL vs Univ # of Compactions , latency Comparison

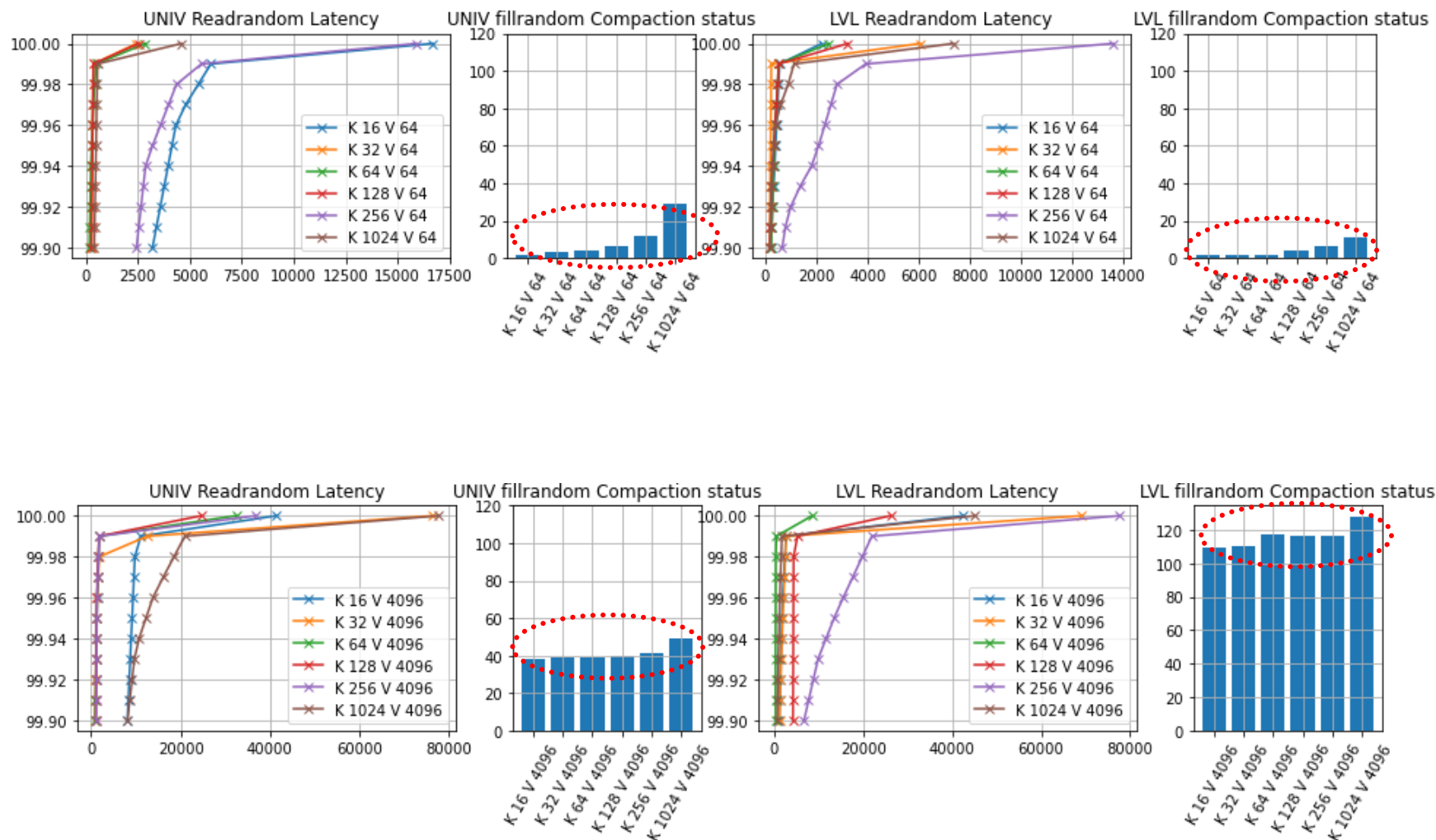
## ■ Fillrandom



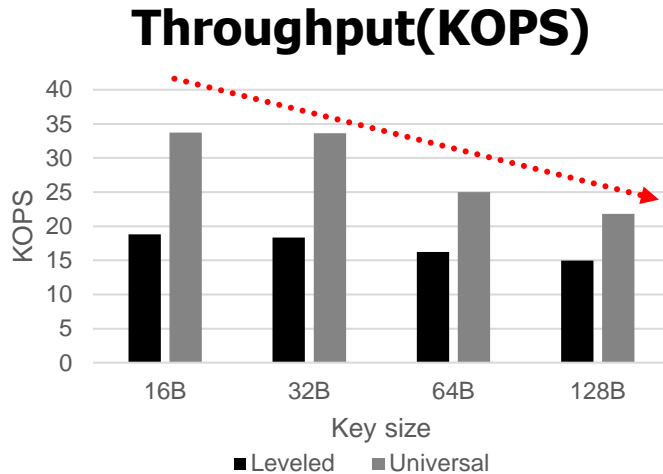


# LVL vs Univ # of Compactions / latency Comparison

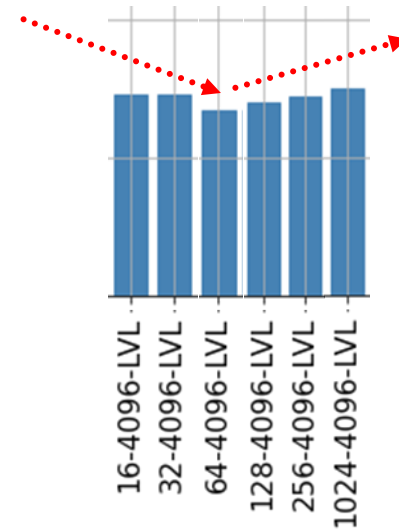
## Readrandom



## ■ Issue on Last week



### 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