Blobdb & Compaction

2026. 01. 28

Presentation by Jonghyeon Lee, Seungyeol Choi, Junseo Park, Damin Jung

suyeanjong0323@dankook.ac.kr

Contents

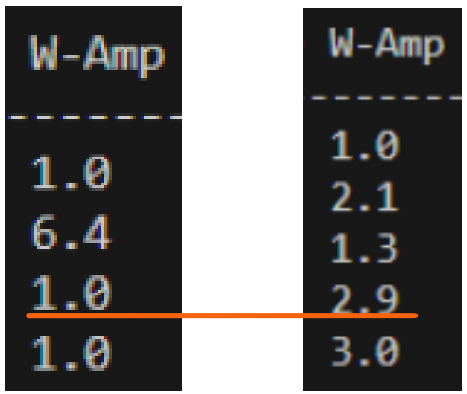
Blobdb Overview

- **Blobdb**: 큰 데이터를 저장하기 위한, Rocksdb의 KV Separation storage engine
- **KV Separation**: LSM 트리에 키와 포인터만 기록하고 실제 데이터는 따로 보관

  -> blobdb: .blob 파일

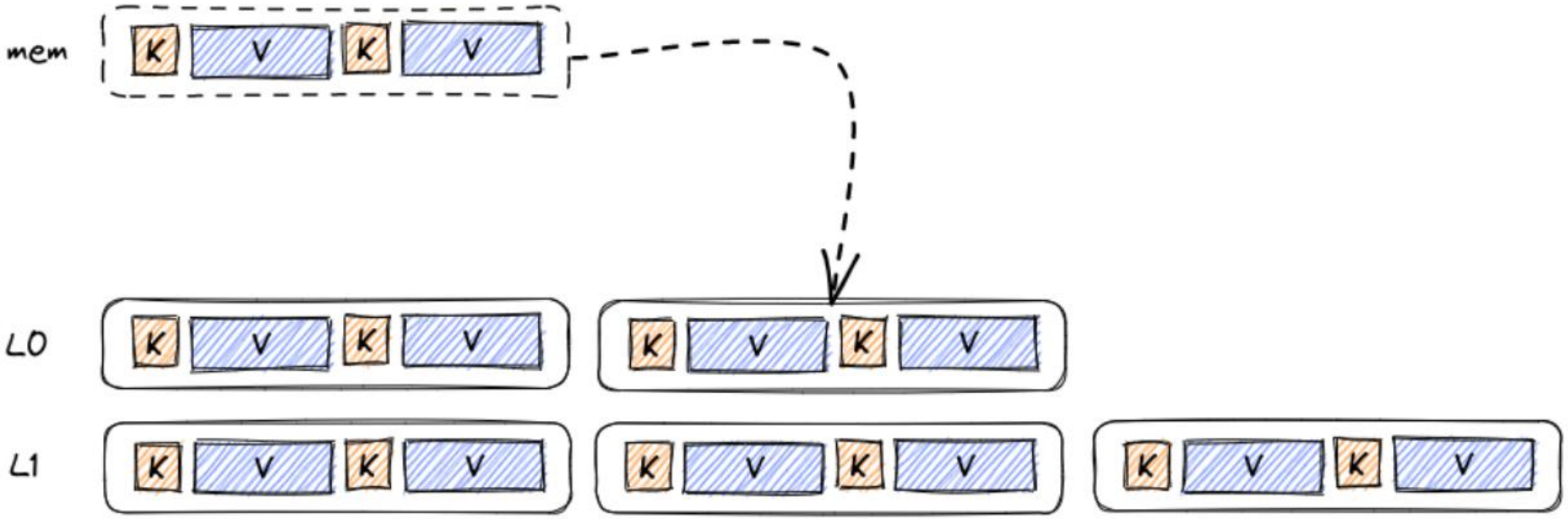
```
Blob file count: 156, total size: 9.7 GB, garbage size: 0.0 GB, space amp: 1.0
```

- 장점: reduced W-Amp(compaction 부하 감소), SSD 수명 연장
- 단점: Space amplification, 스캔 성능 저하



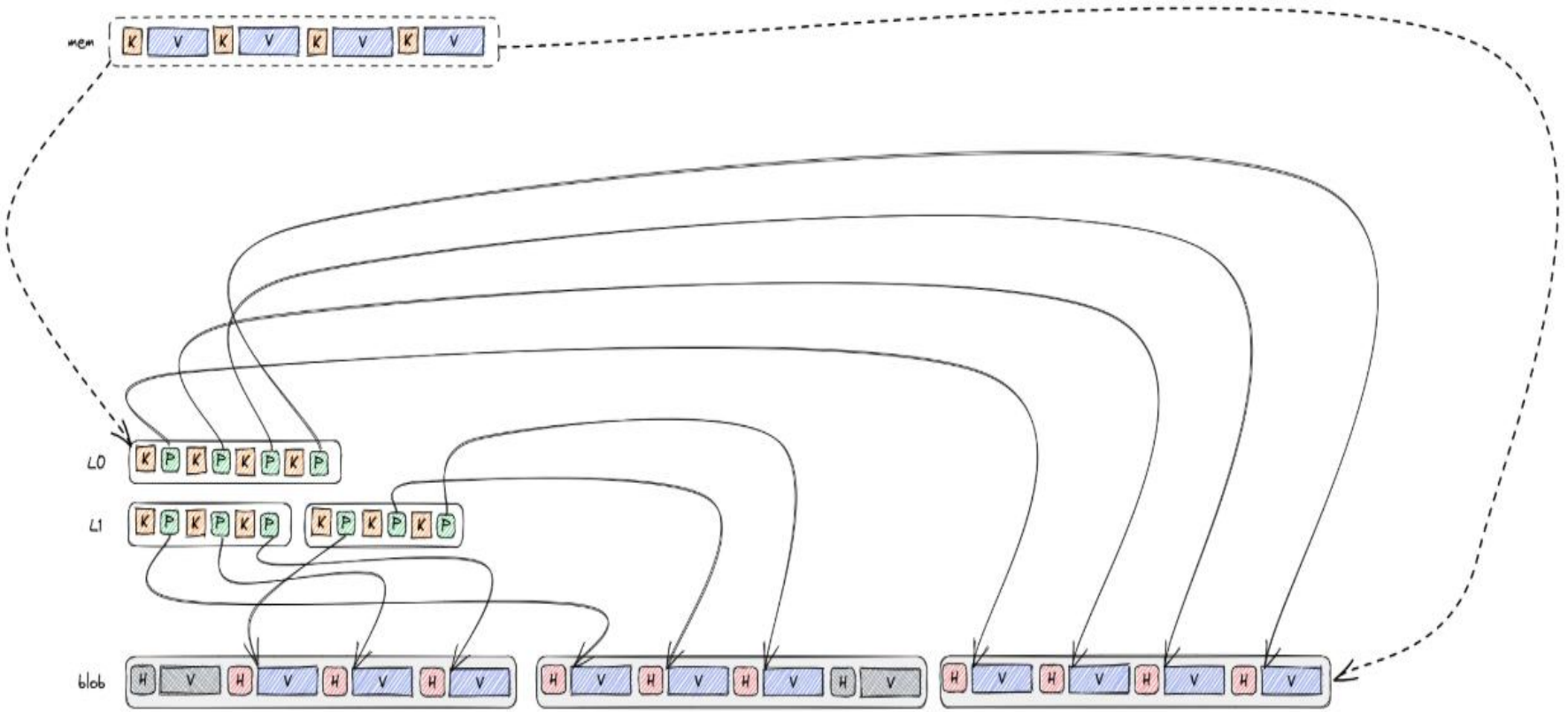
Blobdb Overview

- **Traditional LSM-TREE**



Blobdb Overview

- **KV Separation**
- **메모리**: KV 쌍 그대로 기록 (Flush할 때 blob file(실제 value), LSM TREE(pointer))




```
** Compaction Stats [default]
Level    Files   Size     Sco
----------------------------
  L0    156/0    830.97 KB
 Sum    156/0    830.97 KB
 Int      0/0      0.00 KB
```

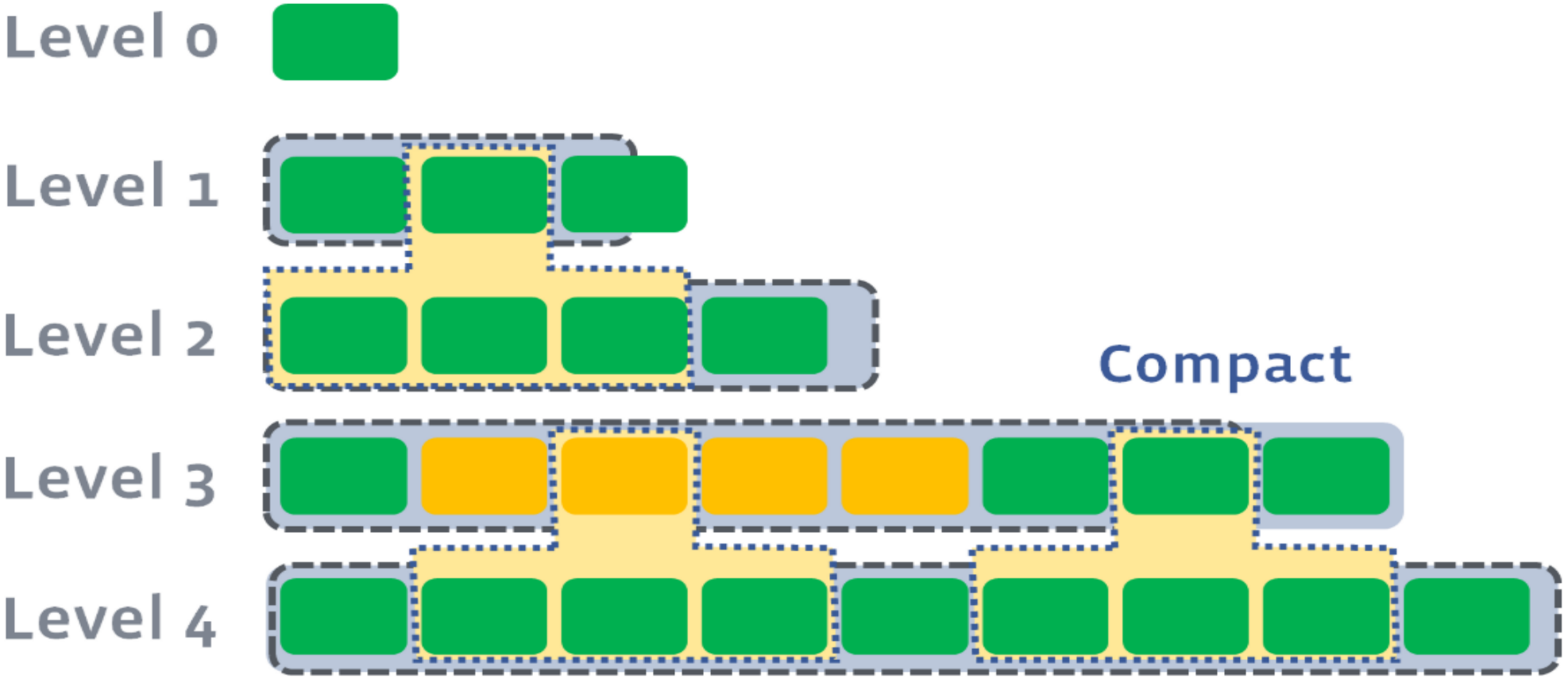
```
** Compaction Stats [default]
Level    Files   Size     Sco
----------------------------
  L0    156/0      9.68 GB
 Sum    156/0      9.68 GB
 Int      0/0      0.00 KB
```

9.77GB / 64MB ≒ 156
(압축X, Compaction X 확인용)

Compaction Overview

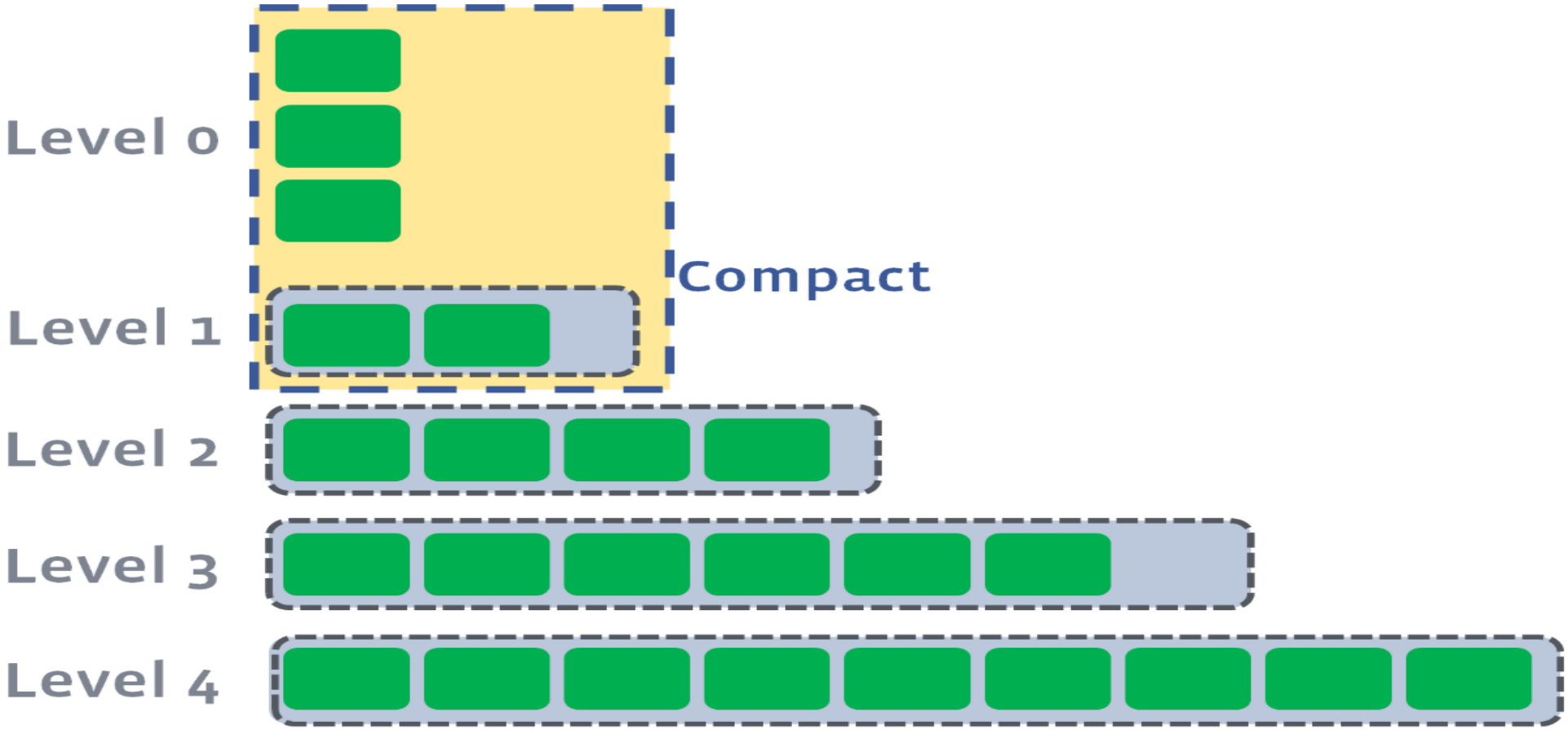
Compaction: Merging and organizing multiple SST files -> Reduce read amplification

-> 쓰기 증폭 증가, 읽기 성능 향상



Compaction Overview(L0)

- L0 Compaction 기준: 파일의 개수



Compaction Overview(L0)

- **level0_file_num_compaction_trigger=4(default)**

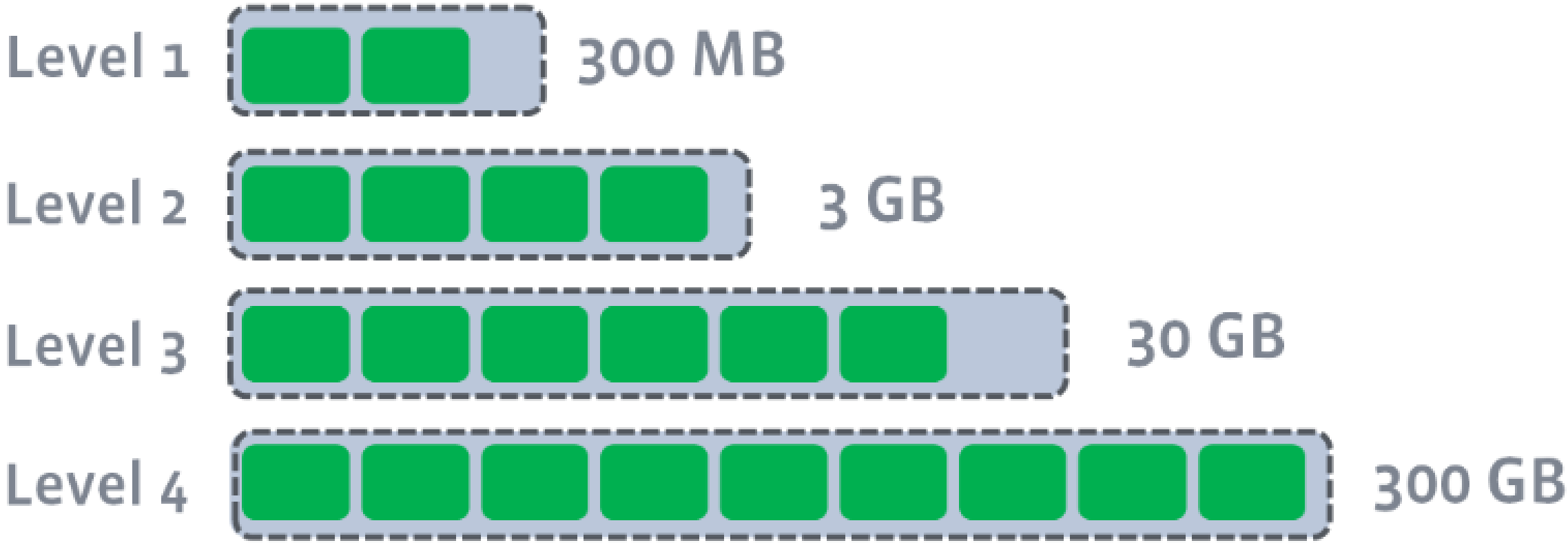
```
** Compaction Stats [default] **
Level    Files   Size     Score
------------------------------
  L0     16/0    525.09 MB   4.0
  L1     11/1      1.06 GB   3.8
  L2     14/1      1.17 GB   0.4
 Sum     41/2      2.74 GB   0.0
 Int      0/0      0.00 KB   0.0
```

<L0 Files> -> 16/0
16: SST 파일 개수, 0: Compaciton 진행 개수

Score: 1.0 이상 -> compact 필요한 상태
(SST 4개: 1.0, SST 16개: 4.0)

Compaction Overview(L1~Ln)

- L1~Ln Compaction 기준: 각 레벨에 쌓인 총 용량



Compaction Overview(L1~Ln)

<db_bench 기준>

- max_bytes_for_level_base=256MB(default) : L1 캐시의 용량
- max_bytes_for_level_multiplier=10(default) : 레벨간 배수

```
** Compaction Stats [default] **
Level    Files   Size     Score
------------------------------
  L0     12/12   405.11 MB   0.0
  L1      7/7    684.89 MB   0.0
  L2      4/0    256.30 MB   0.1
 Sum     23/19     1.31 GB   0.0
 Int      0/0     0.00 KB   0.0
```

Level	최대 용량
L1	256 MB
L2	2.56 GB
L3	25.6 GB

L1의 Score 0.0인 이유 -> 전부 compaction 진행중 (7/7)

Hypothesis(blobdb)

- **Blobdb는 쓰기 증폭이 감소하지만, 각종 오버헤드가 존재**

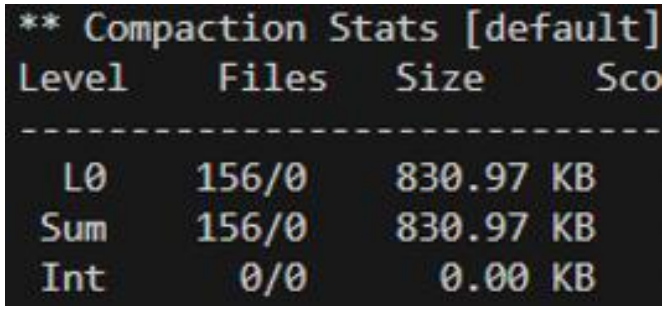
(Blob 파일 기록 및 관리 등)

- **데이터의 크기가 작으면 쓰기 성능이 오히려 떨어지고, 데이터가 크면 쓰기 성능이 상승할 것으로 예상**
- **이중 I/O로 인해, 읽기 성능은 Rocksdb가 더 좋을 것으로 예상**

-> 크기별로, Rocksdb와 Blobdb의 쓰기, 스캔 성능 비교

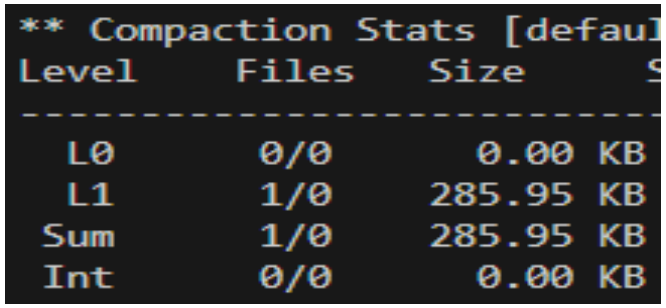
Hypothesis(compaction)

- **Compaction 부하와 Compaction으로 인한 스캔 성능 향상은 항상 비례하는가?**



```
** Compaction Stats [default]
Level    Files   Size     Sco
----------------------------
  L0    156/0    830.97 KB
 Sum    156/0    830.97 KB
 Int      0/0      0.00 KB
```

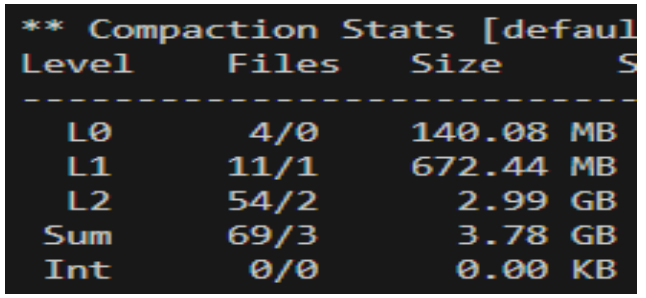
Compaction X



```
** Compaction Stats [defaul
Level    Files   Size     S
----------------------------
  L0      0/0      0.00 KB
  L1      1/0    285.95 KB
 Sum      1/0    285.95 KB
 Int      0/0      0.00 KB
```

Compaction O

<Blobdb>



```
** Compaction Stats [defaul
Level    Files   Size     S
----------------------------
  L0      4/0    140.08 MB
  L1     11/1    672.44 MB
  L2     54/2      2.99 GB
 Sum     69/3      3.78 GB
 Int      0/0      0.00 KB
```

<Rocksdb>

- **파일의 용량이 적어, L1 캐시에 머무르지만, SST 파일이 병합으로 156개->1개로 감소 -> Compaction 부하는 적지만, 어느 정도 이득을 봄**

-> Rocksdb, Blobdb Compaction On/Off 스캔 성능 비교

Hypothesis

- **Blobdb**
  - 쓰기 성능
    - **Blobdb > Rocksdb**
  - 읽기 성능
    - **Blobdb < Rocksdb**

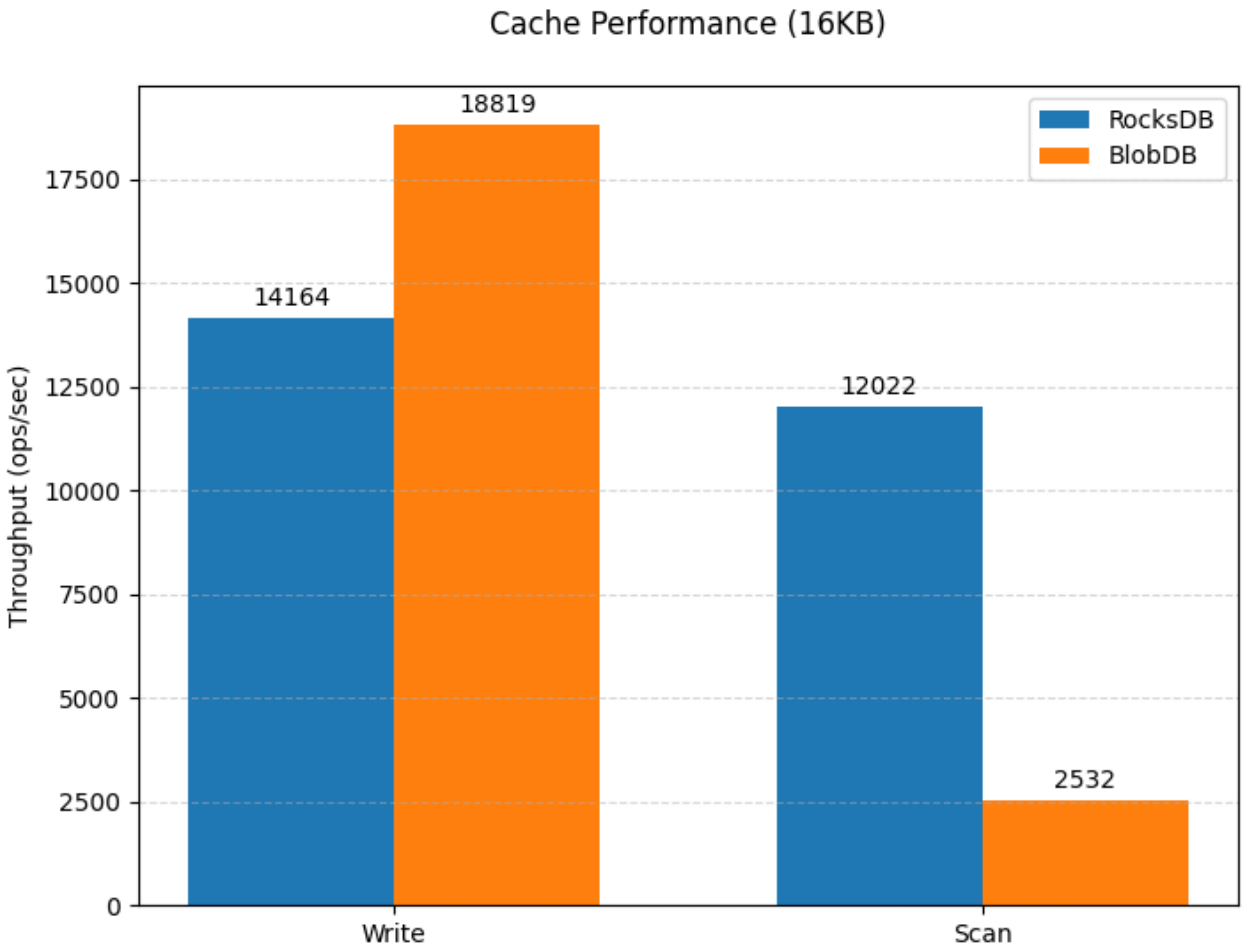
- **Compaction**
  - **Compaction 부하와 스캔 성능 향상은 비례할 것이다. (Rocksdb, Blobdb)**

실험 환경

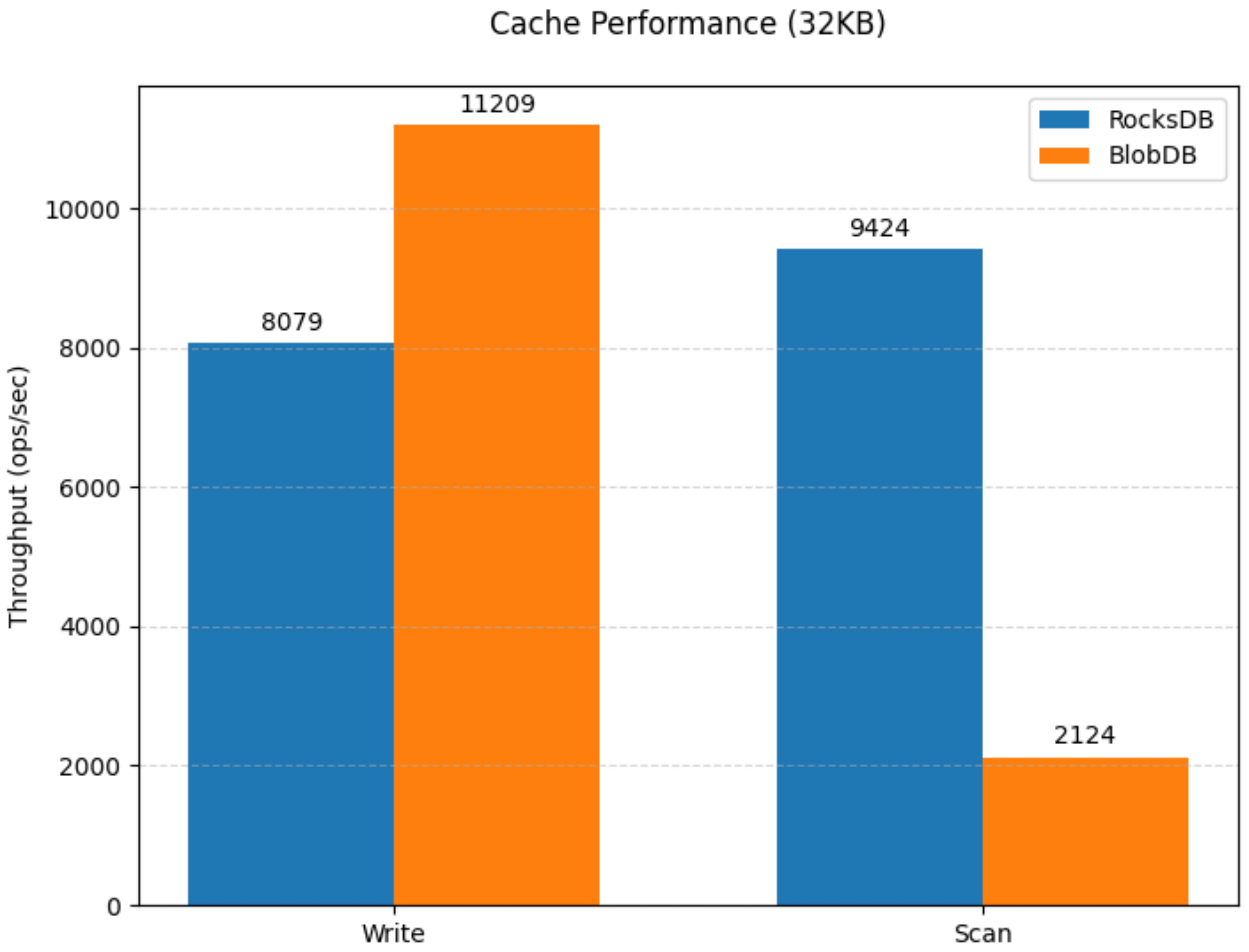
- CPU : Intel® Core™ Ultra 7 Processor 155H
- RAM : 4 GB (4096 MB)
- OS : Ubuntu Linux
- Oracle VM VirtualBox 사용

- TEST SIZES(KB) : 16, 32, 64, 128, 256, 512
- 파일개수 : 25000개
- 20회 반복시행의 평균값

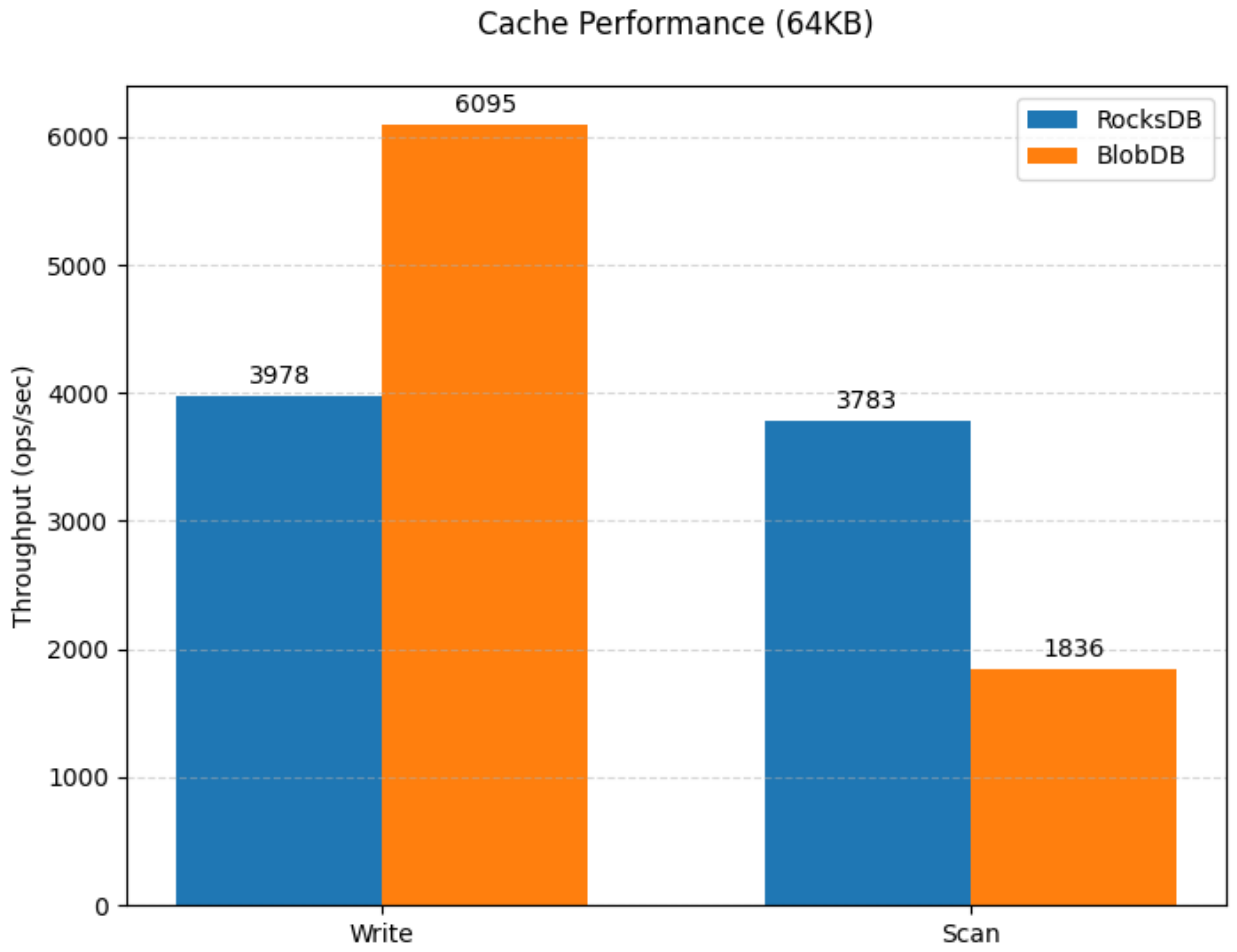
Result & Analysis – 16KB



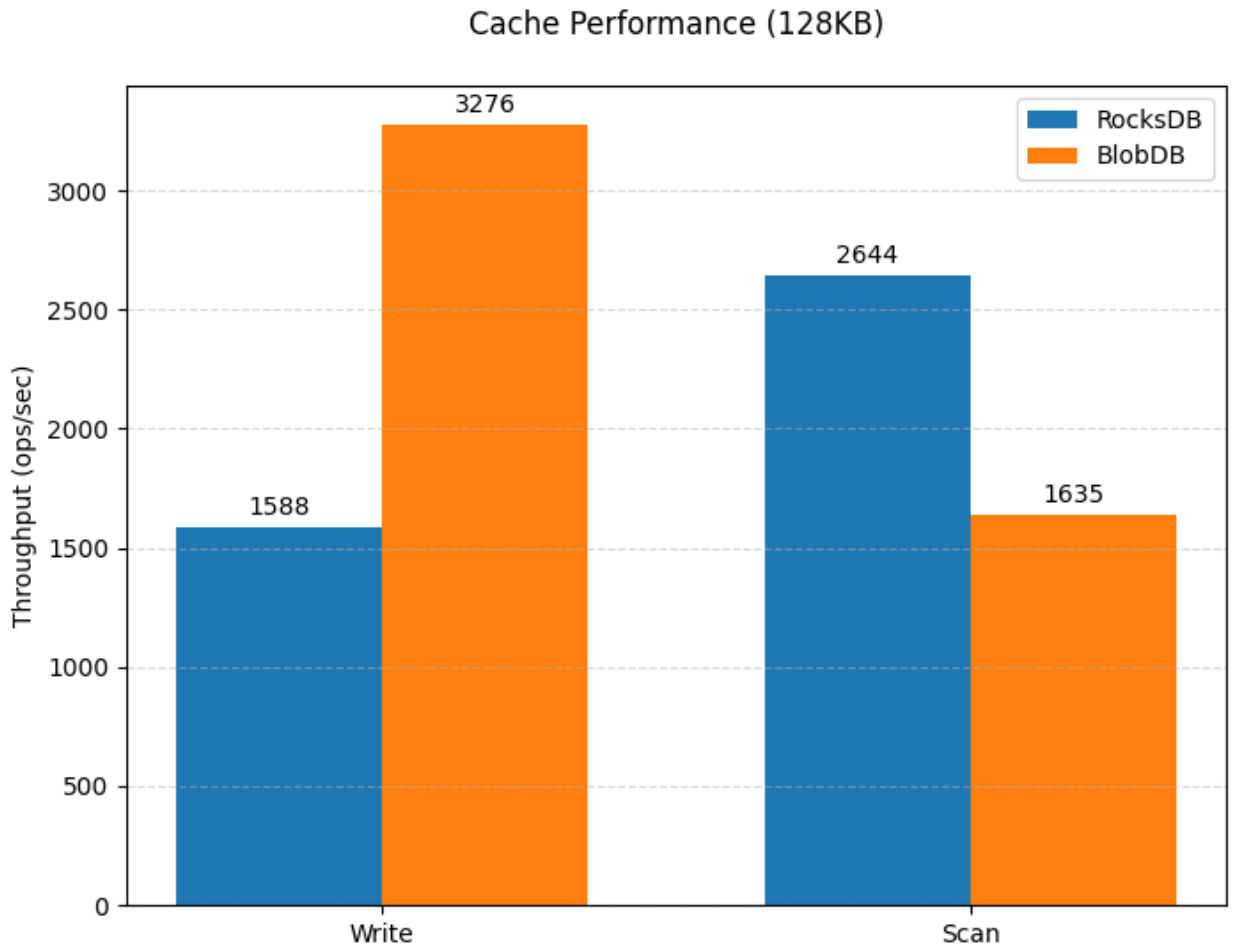
Result & Analysis – 32KB



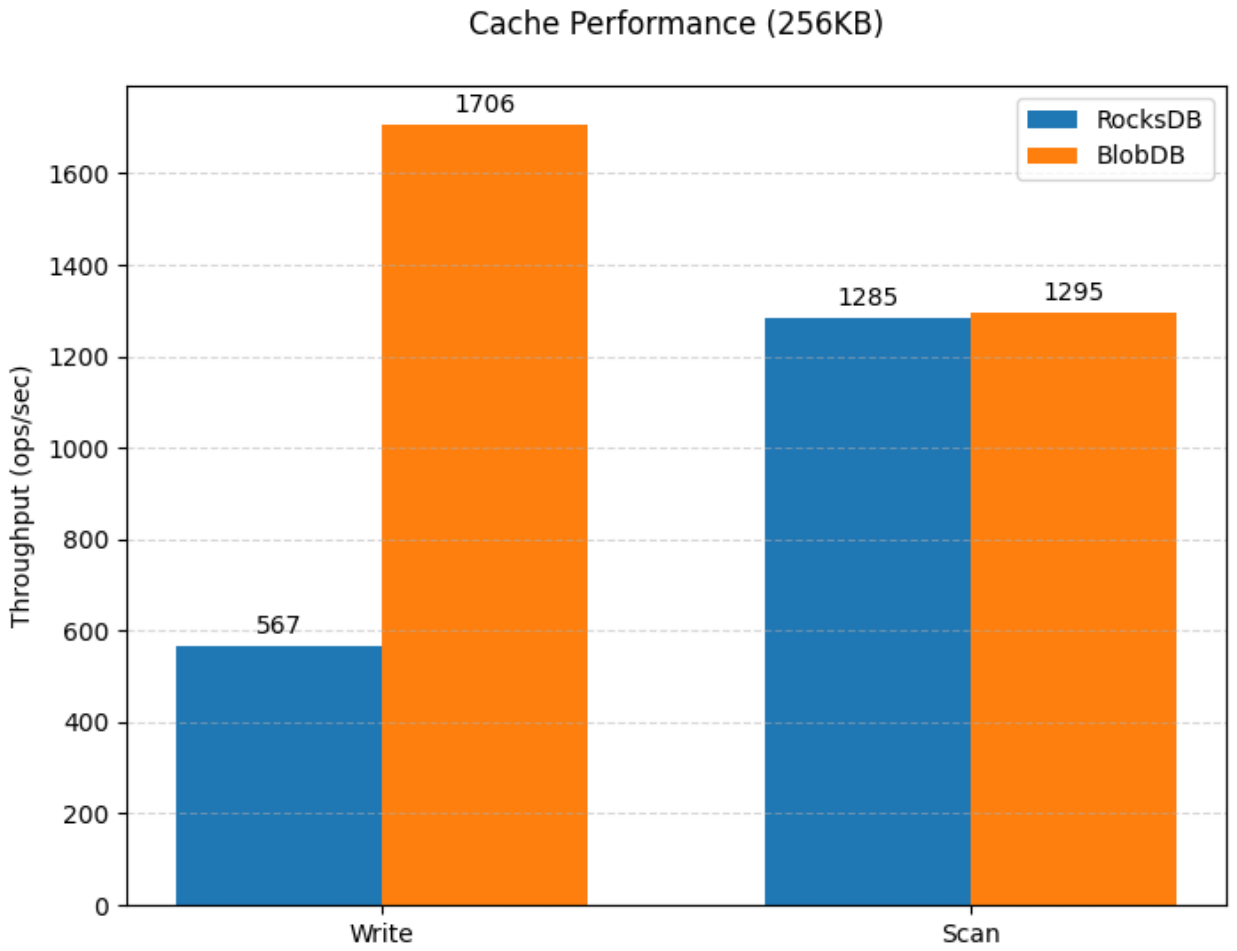
Result & Analysis – 64KB



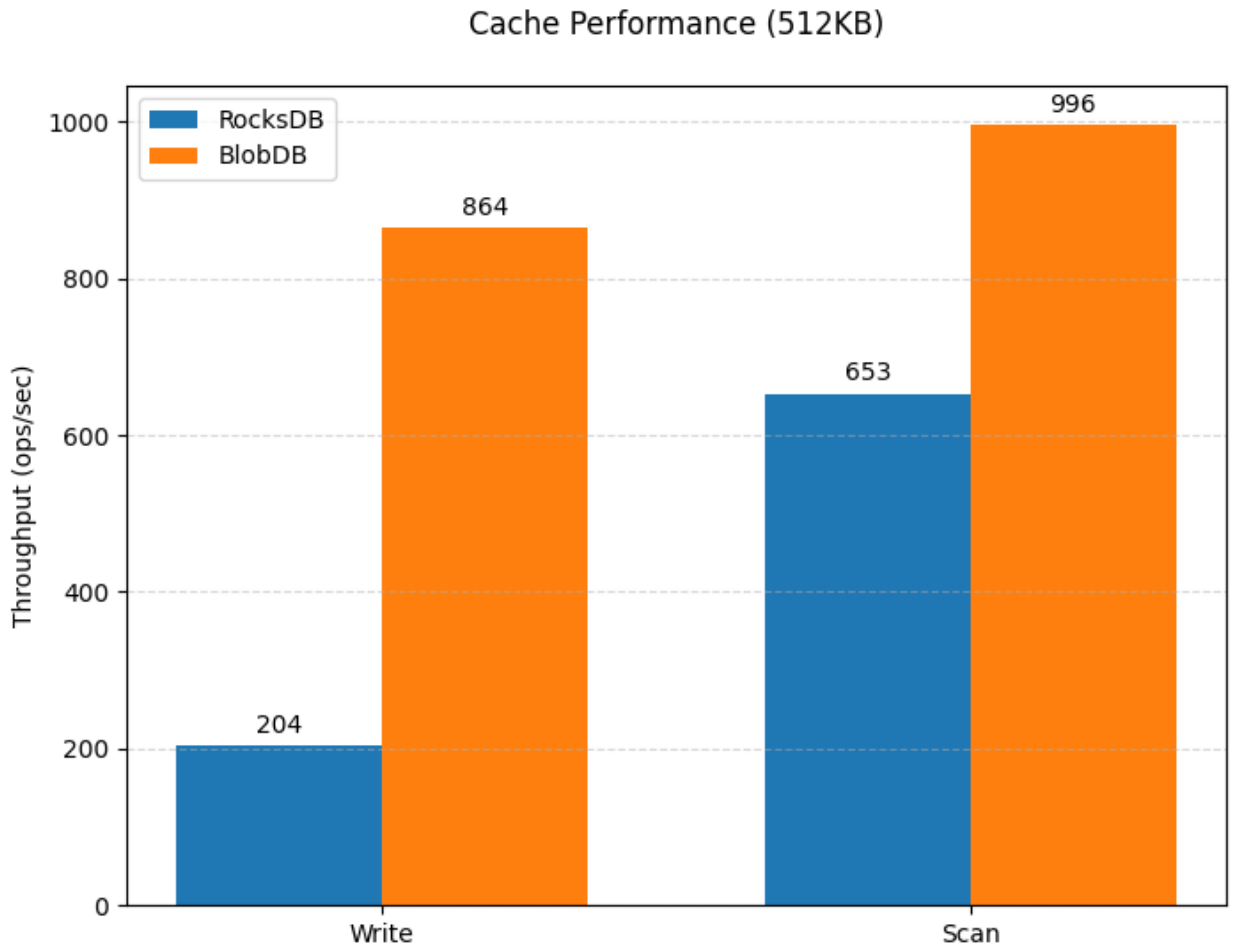
Result & Analysis – 128KB



Result & Analysis – 256KB



Result & Analysis – 512KB

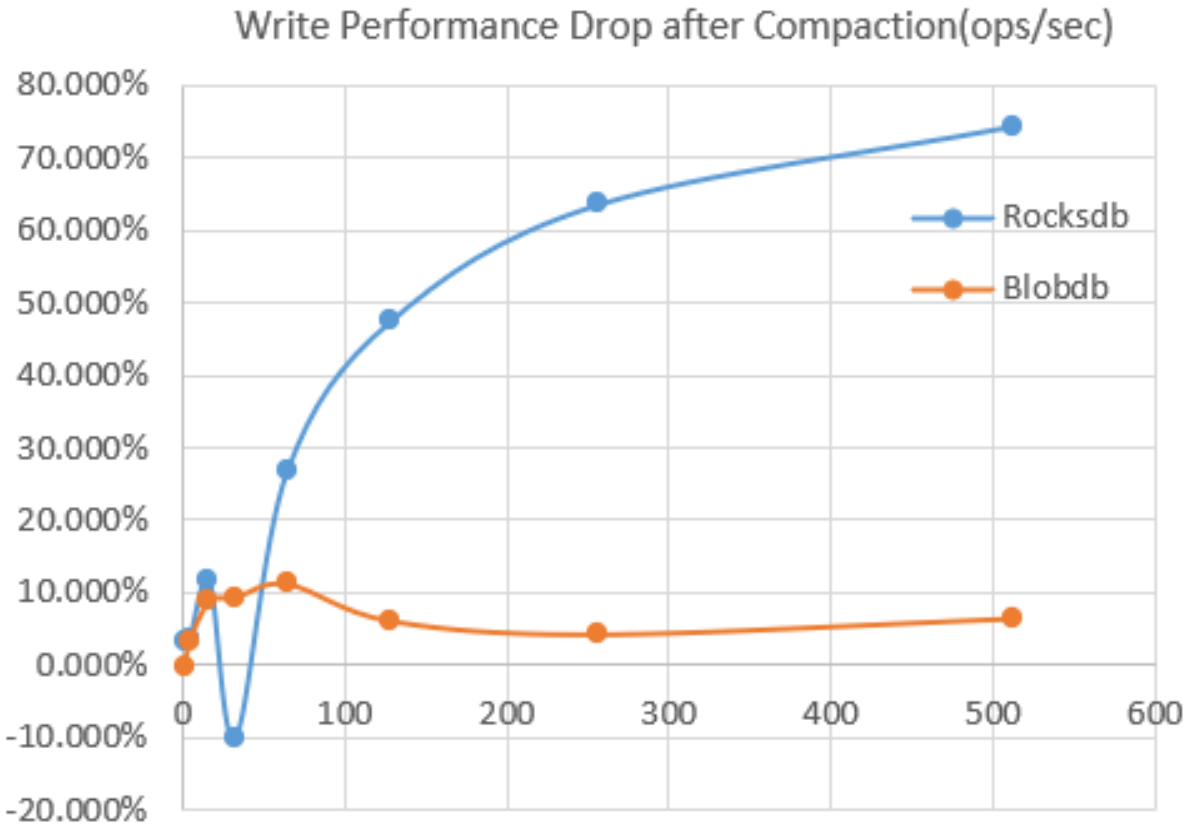


Result & Analysis(Compaction)

- **Compaction off -> on 쓰기 성능 감소량**

- **예상대로 Rocksdb의 쓰기 성능은 크게 감소하고, Blobdb는 적게 감소**

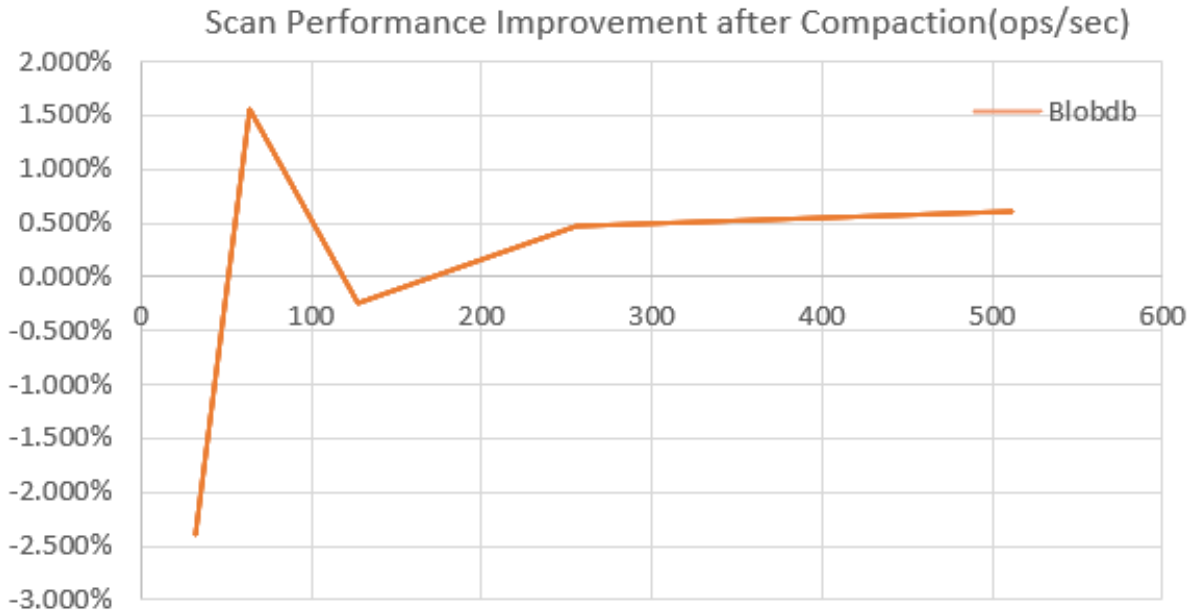
- **Blobdb는 value에 주소를 입력하므로, Compaction 부하 차이가 없음**



Result & Analysis(Compaction)

- **Compaction off -> on**

  blobdb 스캔 성능 증가량



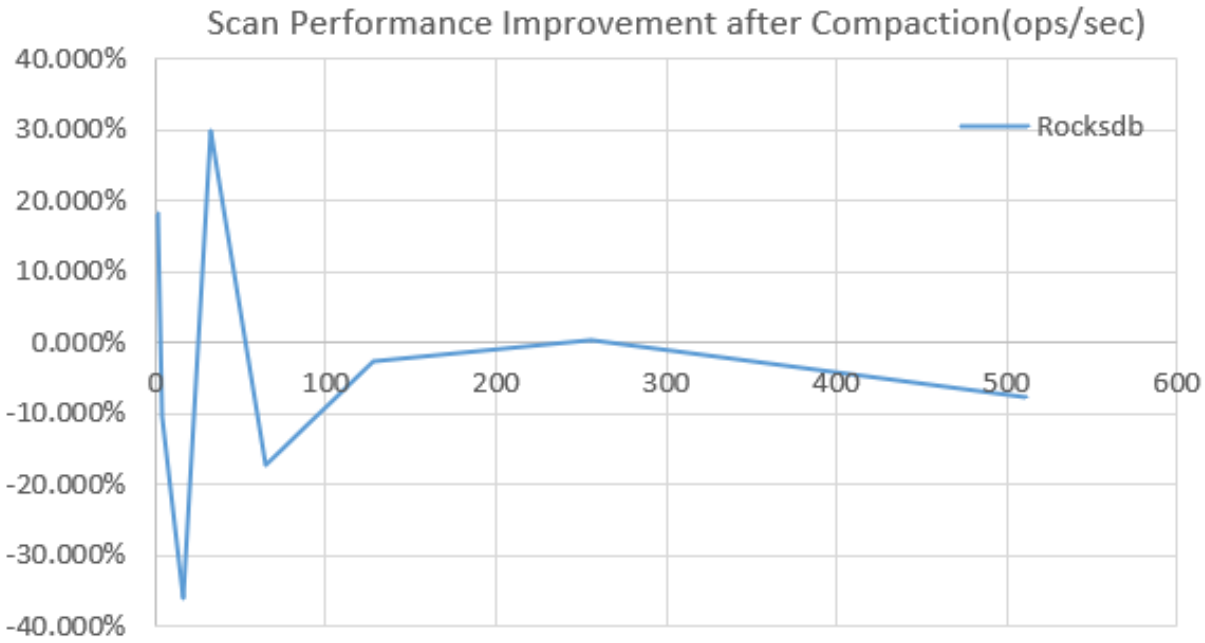
- **LSM-TREE 용량이 적어도 SST 파일 (수백 -> 몇 개) 감소에 대한 의문과 달리 미미한 차이**

- **Compaction 부하가 적으면 스캔 성능 향상이 적음(Blobdb)**

Result & Analysis(Compaction)

- **Compaction off -> on**

  Rocksdb 스캔 성능 증가량



- **스캔 성능이 무조건 증가해야 하는데, 오히려 감소함 (잘못된 결과)**

Discussion

- **Rocksdb 스캔 성능 문제점**

-> 원인: Write Stall

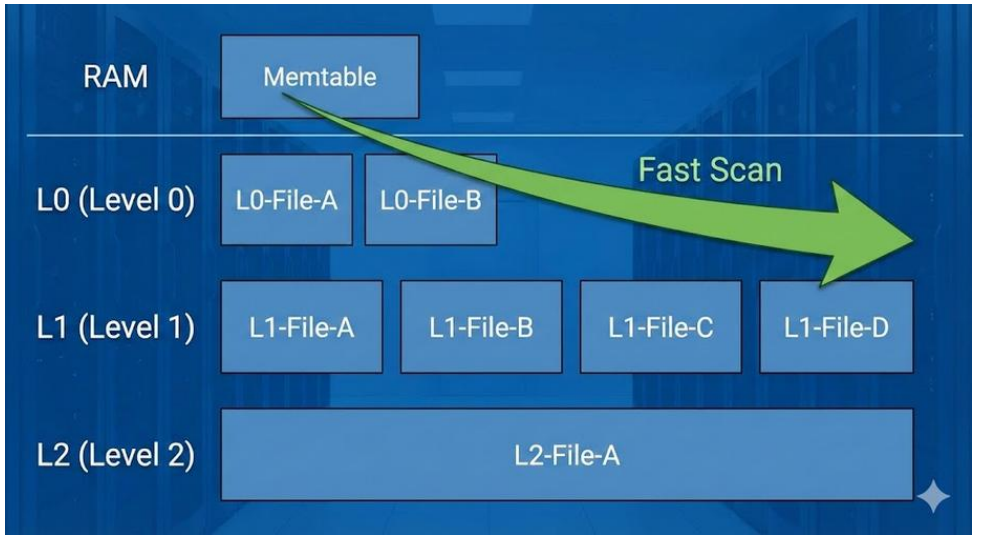
(32KB) Write stall: 24s 임을 확인

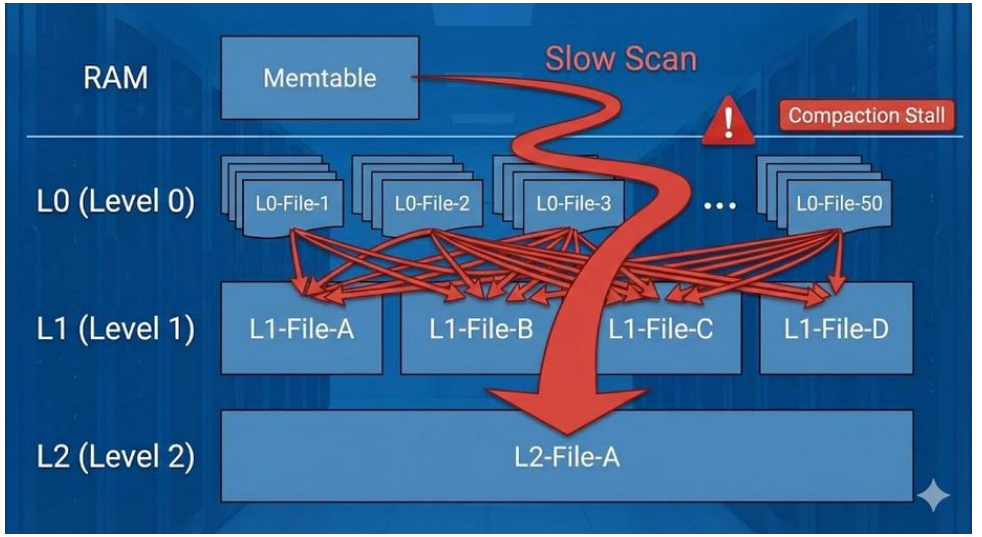
24s -> 11s

(Memtable 개수 1 -> 4)

(Memtable 용량 64MB -> 128MB)

➔ **추후 실험은 Write stall을 고려해서 진행**





Thank you

- 출처
- Home · facebook/rocksdb Wiki · GitHub