단국대학교 DANKOOK UNIVERSITY

Zipfian Workload 기반 Time-Aware Tiered Storage 성능비교

과 목 명: 오픈소스SW분석(빅데이터) 1분반
발 표 일: 2025년 06월 05일 (목)
팀　　원: 구선주 (32220207), 임수연 (32193772), 최예림 (32224684)

Contents

Zipfian Workload 기반 Time-Aware Tiered Storage 성능비교

01

서론

- 배경 지식
- 연구 배경

배경 지식

Hot Data vs Cold Data

	Hot Data	Cold Data
접근 빈도	자주 접근됨	거의 접근되지 않음
저장 매체	고성능 저장소 사용 (SSD, 메모리 DB)	비용 효율적인 저장소 사용 (HDD, 테이프, 클라우드 아카이브)
조회 속도	빠른 조회가 필요	느린 조회도 허용
사용 사례	실시간 트랜잭션, 활성 사용자 세션, 실시간 분석	역사 기록, 백업, 오래된 파일 보관
비용	고성능 요구로 인해 일반적으로 비용 높음	느린 속도 허용으로 일반적으로 비용 낮음
데이터 관리	속도 및 효율 최적화가 중요	장기 저장 및 비용 효율성에 초점

배경 지식

RocksDB Storage

RocksDB Hot Instance

(빠른 접근용, SSD 등 고성능 저장소)

- 신규 데이터 저장
- 자주 조회되는 데이터 유지

시간 경과

접근 빈도 감소

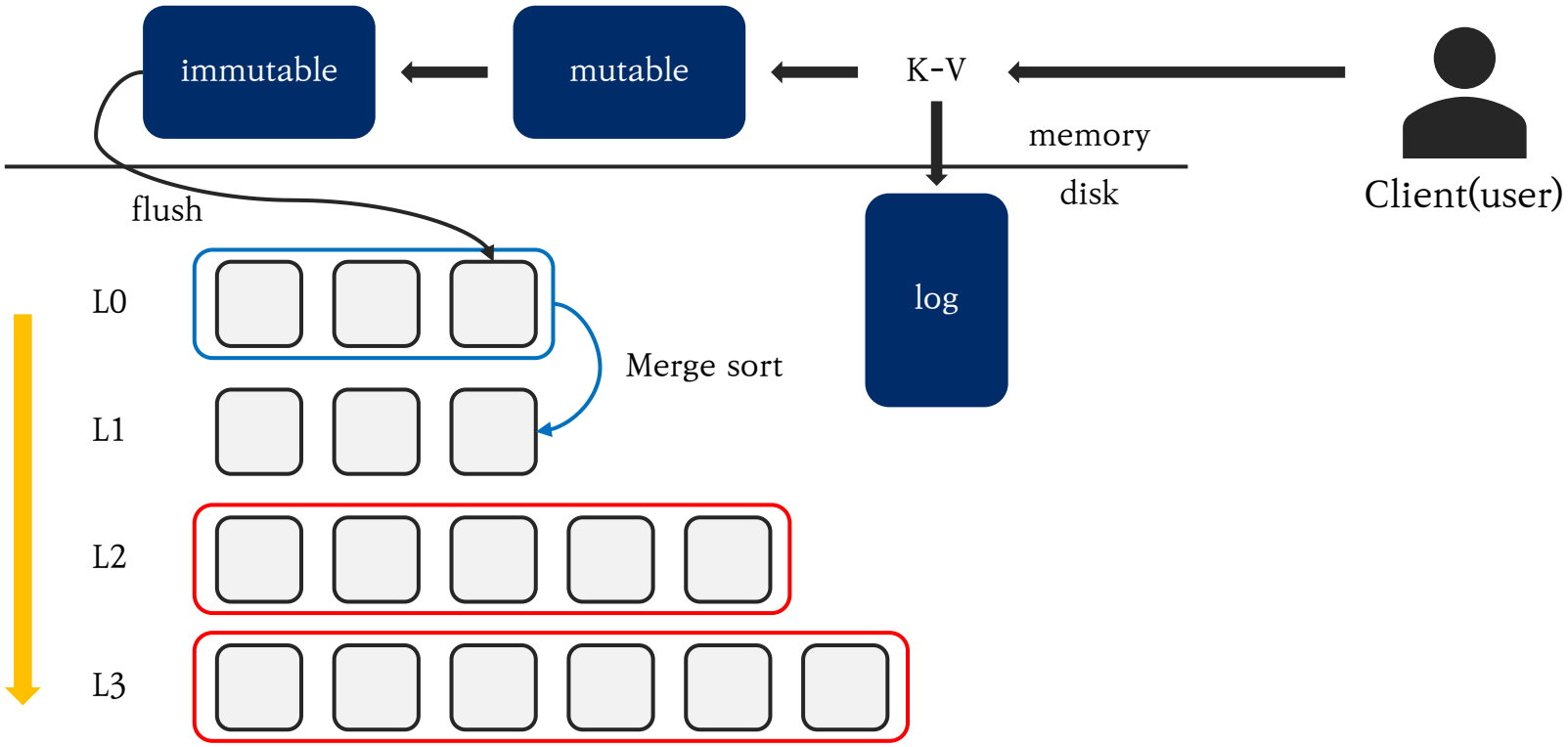
RocksDB Cold Instance

(느리지만 저렴한 저장소, HDD 등)

- 오래된 데이터 이전
- 거의 조회되지 않는 데이터 유지

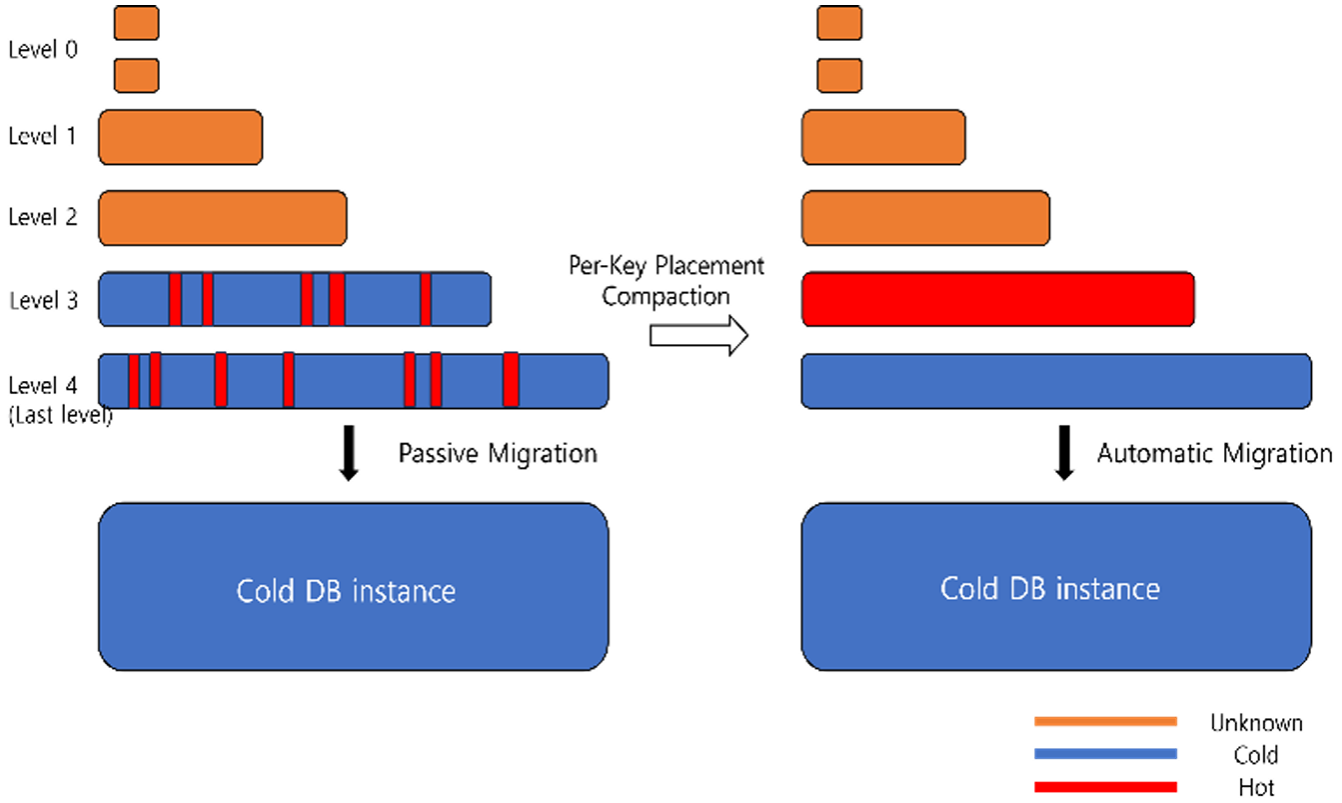
배경 지식

Time-Aware Tiered Storage



배경 지식

Time-Aware Tiered Storage



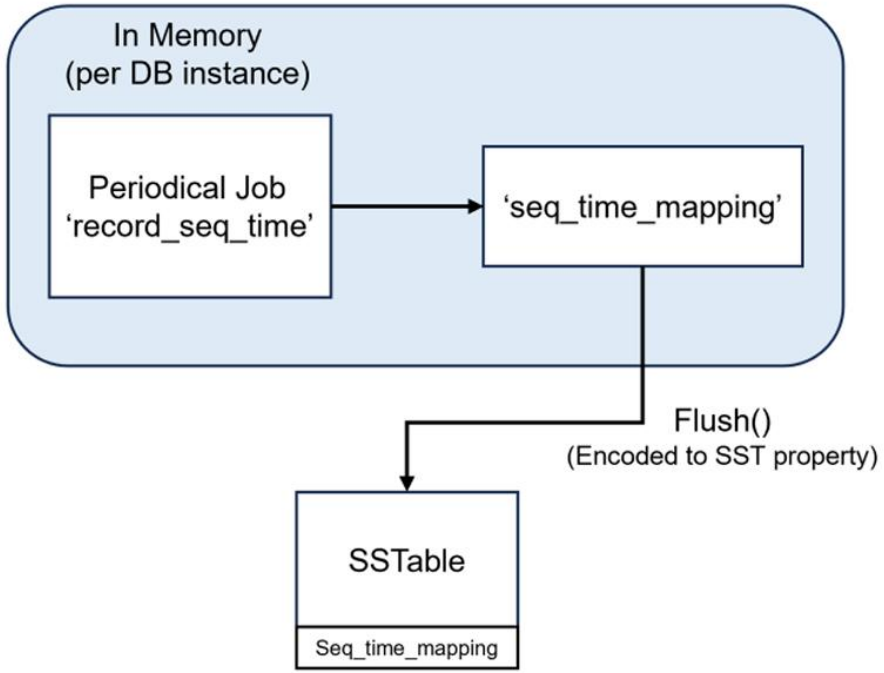
배경 지식

Time-Aware Tiered Storage

Time Tracking	Per-Key Placement Compaction
데이터베이스 내에서 데이터가 쓰여진 시간 추적 → 핫/콜드 구분	Key 별로 데이터의 배치 관리 핫/콜드 데이터 → 다른 스토리지 미디어

배경 지식

Time Tracking

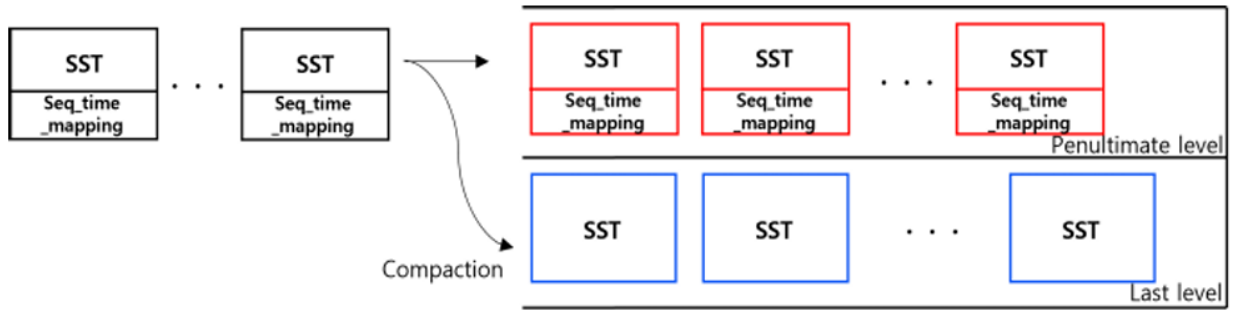


(task interval) = preserve_internal_time_seconds / 100

- Task interval: 최신 데이터 유지 시간 / 100
- seq_time_mapping: 메모리에 보관되는 시퀀스-시간 쌍
- Delta Encoding: SST에 기록 (1KB 미만)
- **정확도 향상**: SST 생성 시간 + 최소 시퀀스 번호로 시점 보정

배경 지식

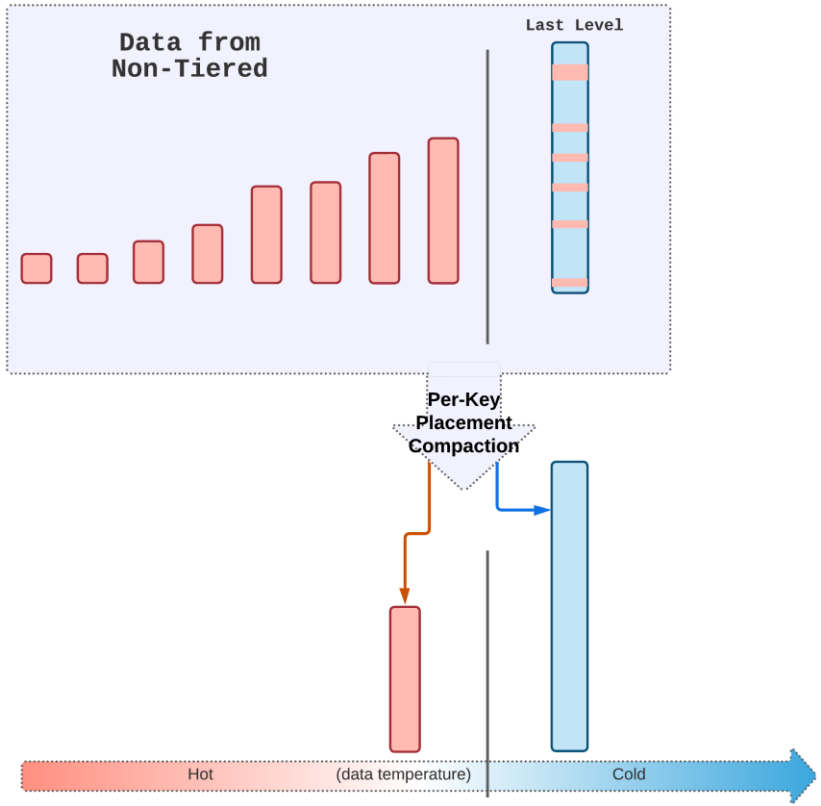
Per-Key Placement Compaction



- Per-Key Placement Compaction: 키 단위로 hot/cold를 구분하여 레벨에 배치
- Penultimate Level: 마지막 바로 이전 레벨 (Hot 데이터 저장)
- Last Level: 마지막 레벨 (Cold 데이터 저장)
- oldest_sequence_number: 기준이 되는 시퀀스 넘버
- last_level_temperature: 저장 매체 결정

배경 지식

Per-Key Placement Compaction



seq-time 정보 → oldest_sequence_number 계산

Hot Data	Cold Data
Oldest보다 최신 데이터	Oldest보다 오래된 데이터
Penultimate Level	Last Level

last_level_temperature = [Hot, Warm, Cold]

Cold SST는 HDD 등으로 배치

배경 지식

Migration

Time-Tracking 비활성화 Migration

- preclude_last_level_data_seconds
- Seq-time 정보가 없음
- Hot 데이터도 Cold로 잘못 분류

↓

성능 저하 가능성

(느린 스토리지로 이동)

Time-Tracking 활성화 Migration

- preserve_internal_time_seconds
- 최근 데이터는 시간 정보가 있음

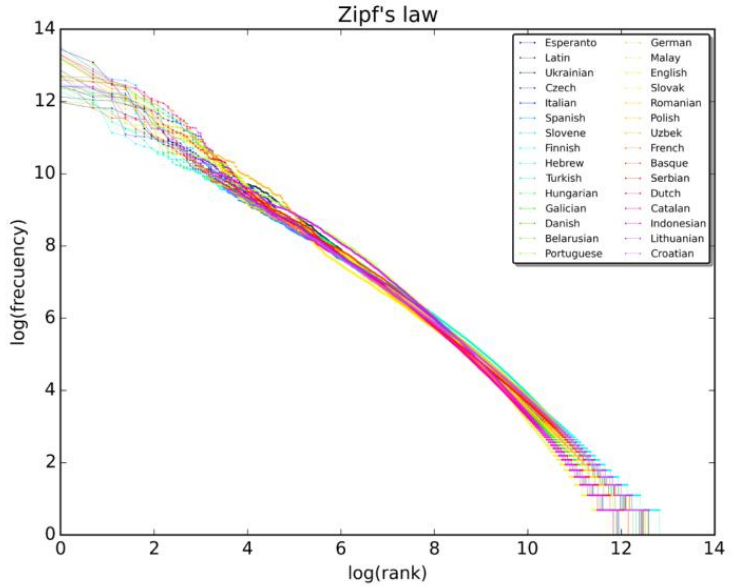
↓

Hot 데이터는 마지막 레벨로 내려가지 않음

Cold 데이터만 Last Level로 이동

연구 배경

Zipfian Workload



- 균등 분포를 따르지 않고 몇몇 키에 집중되는 분포
- 효율적인 compaction 정책 수립에 도움

연구 배경

Zipfian Workload

실험 1	실험 2
Default: Universal Compaction **TATS**: Tiered Storage 활성화 ↓ Tiered Storage가 성능에 미치는 영향	**preserve**: 0%, 20%, 50%, 80%, 100% **preclude**: 0%, 20%, 50%, 80%, 100% ↓ Tiered Storage의 preclude, preserve 값이 RocksDB 쓰기 성능에 미치는 영향

02

본론1

- 가설 설정
- 실험 설계
- 실험 결과

가설 설정

가설 설정

Universal Compaction만 사용한 경우와 비교하여

Time-Aware Tiered Storage (TATS)는 오버헤드가 발생하여 쓰기 성능이 더 낮게 나올 것이다

Case	설명
Default	Zipfian Workload에서 Universal Compaction으로 Write 성능 측정
TATS	Zipfian Workload에서 TATS (default)를 사용했을 때 Write 성능 측정

- Storage를 따로 지정하지 않은 Time-Aware Storage Tiered를 사용했을 때 영향 분석

실험 설계

Zipfian workload

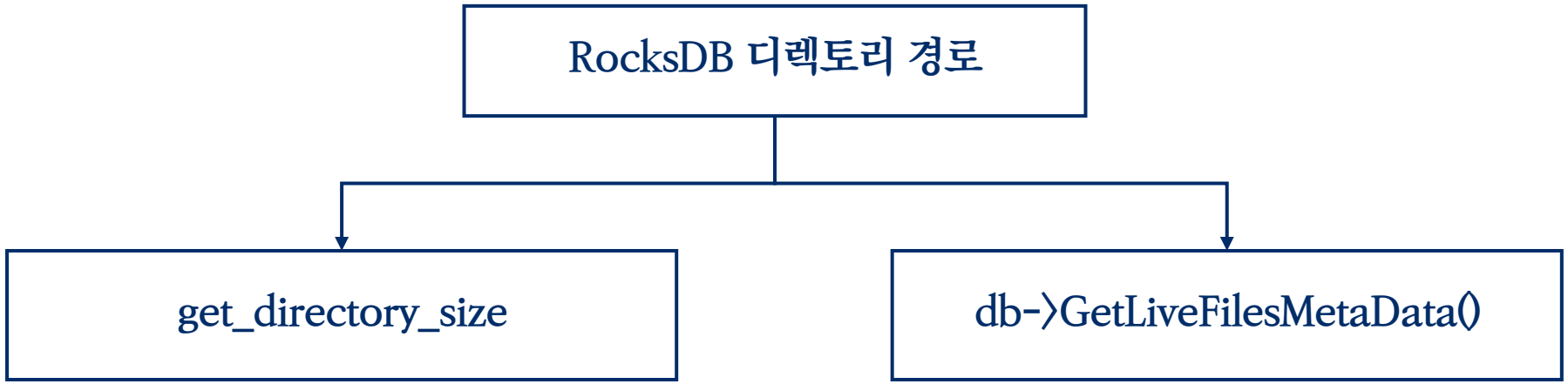
```
public:
    ZipfGenerator(int n, double alpha)
        : generator(42)  // 시드 고정
    {
        std::vector<double> weights(n);
        for (int i = 0; i < n; ++i) {
            weights[i] = 1.0 / pow(i + 1, alpha);
        }
        distribution = std::discrete_distribution<int>(weights.begin(), weights.end());
    }

    int next() {
        return distribution(generator);
    }
};
```

- 일부 hot 데이터에 대한 접근 빈도가 높고, cold 데이터에 대한 접근 빈도가 낮다
- Zipfian workload에서 alpha 값이 크면 hot 데이터에 대한 집중도가 높아진다

실험 설계

Directory 측정



- du -sb <path> 호출
- 디렉토리 바이트 단위 크기 측정

- Live SST 파일 메타데이터 수집
- Level, temperature, size 등 포함

실험 설계

측정 지표

$$\text{last\_level\_byte\_ratio} = \frac{\text{rocksdb.last.level.read.bytes}}{\text{rocksdb.last.level.read.bytes} + \text{rocksdb.non.last.level.read.bytes}}$$

$$\text{Write Amplification Factor} = \frac{\text{rocksdb.flush.write.bytes} + \text{rocksdb.compact.write.bytes} + \text{rocksdb.wal.bytes}}{\text{rocksdb.bytes.written}}$$

실험 결과

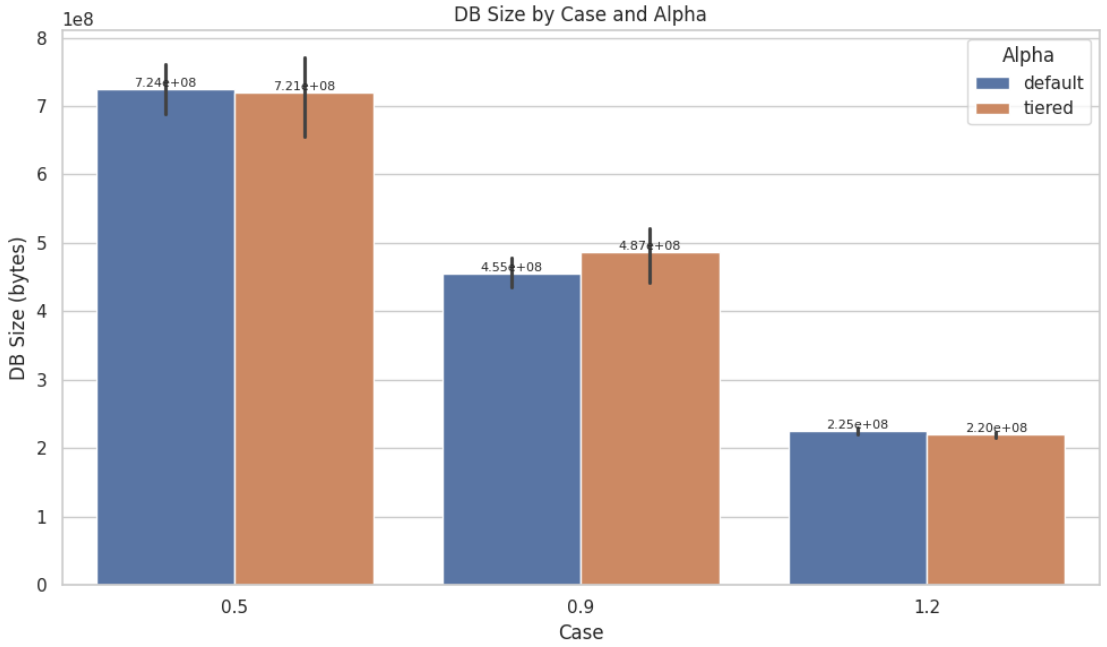
실험 환경

OS	Ubuntu 20.04.6 LTS (64bit)
CPU	16 * Intel(R) Core(TM) i7-10700F CPU @ 2.90GHz
RAM	31.2GiB
SSD	1.0TB
RocksDB	Version 10.2.0

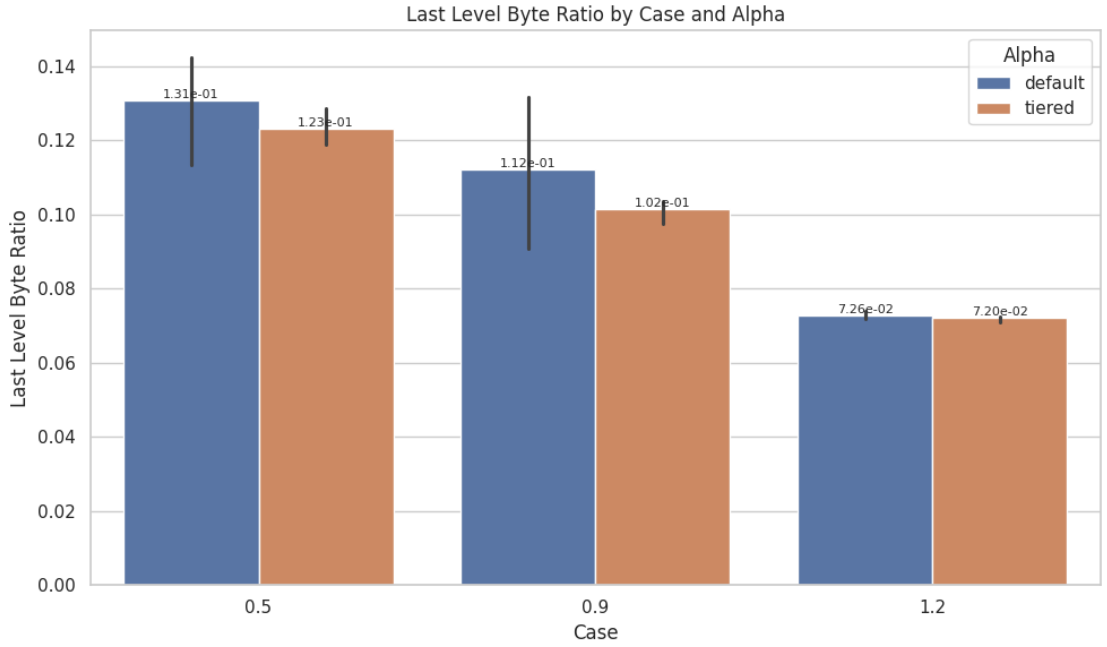
실험 결과

실험1 결과

DB Size



Last Level Byte Ratio

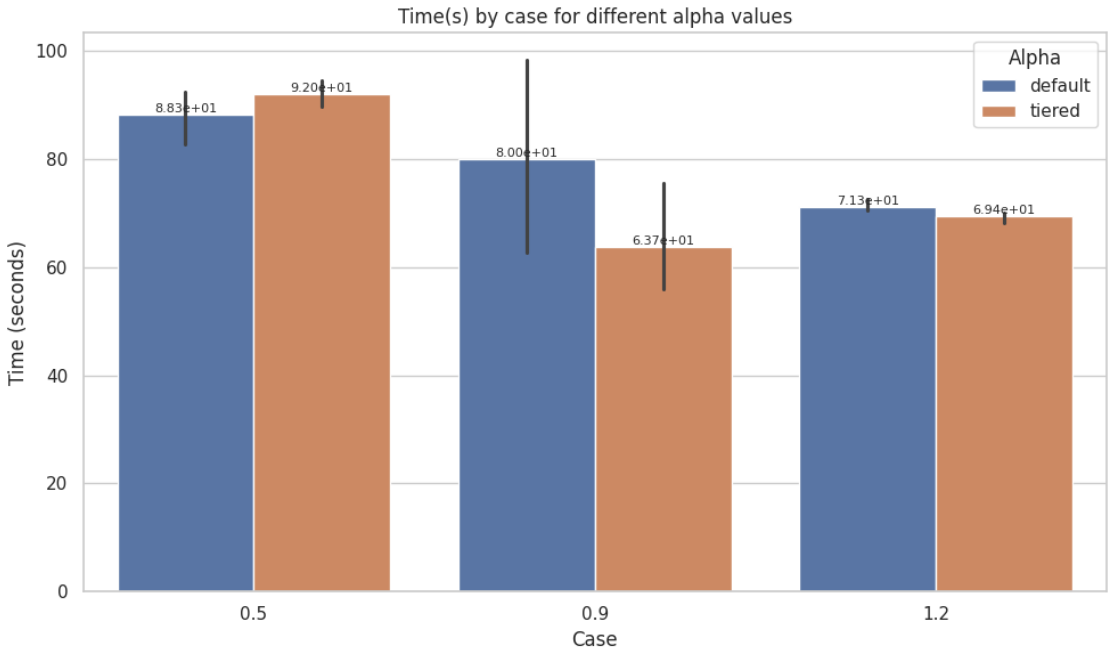


➡ alpha 값이 커지면 성능 향상, default와 tiered 성능 비슷

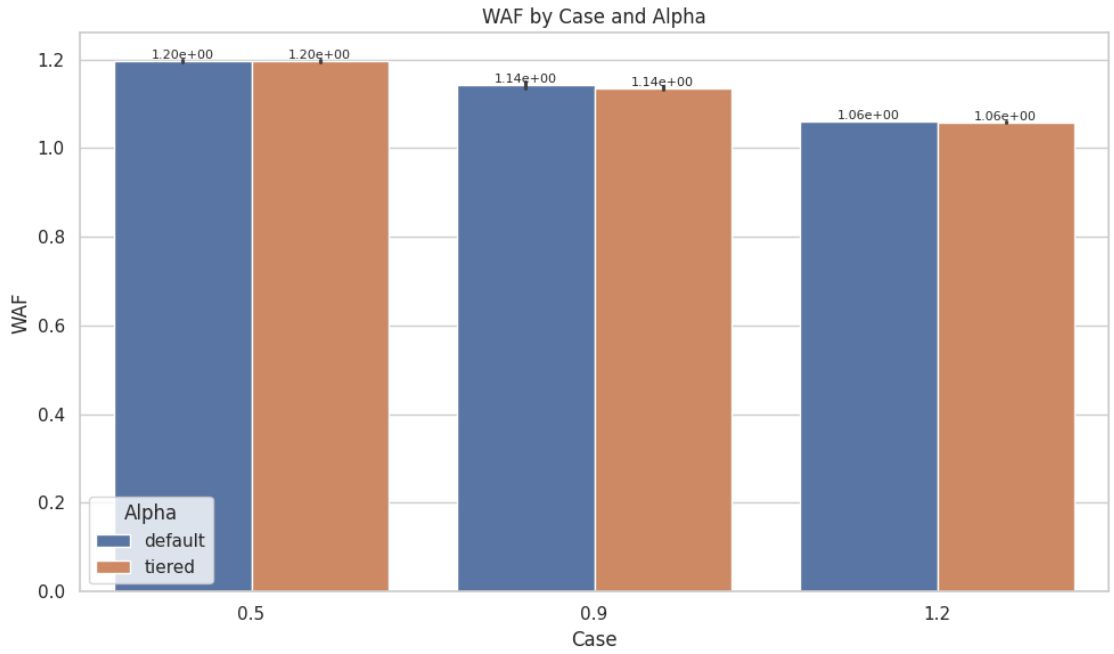
실험 결과

실험1 결과

Time



WAF

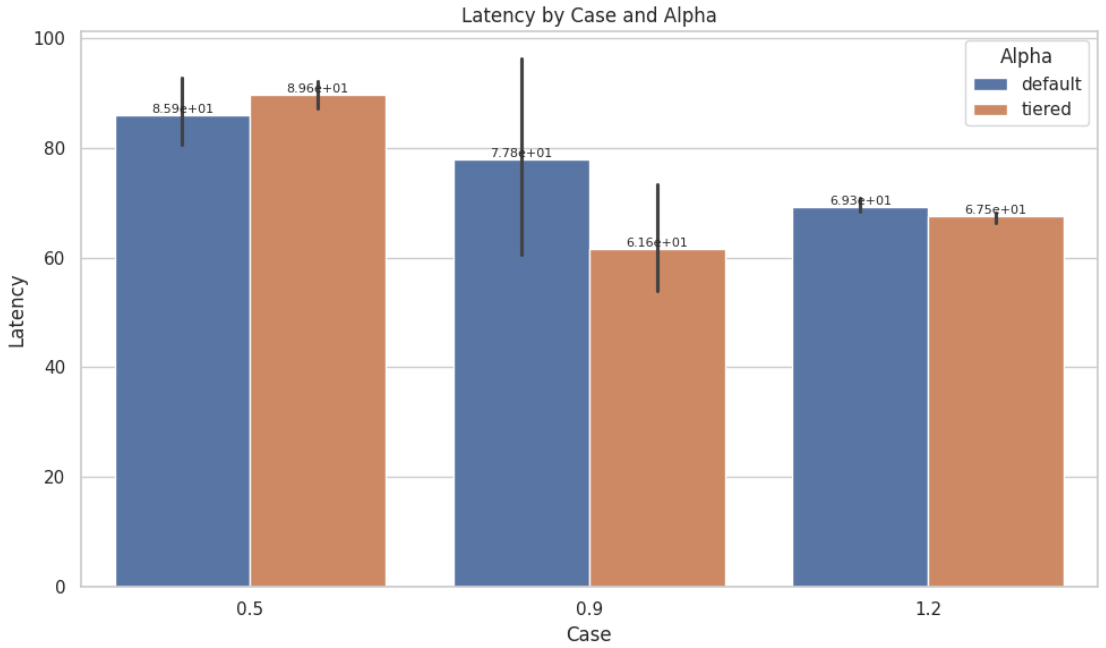


➡ alpha 값이 커지면 성능 향상, default와 tiered 성능 비슷

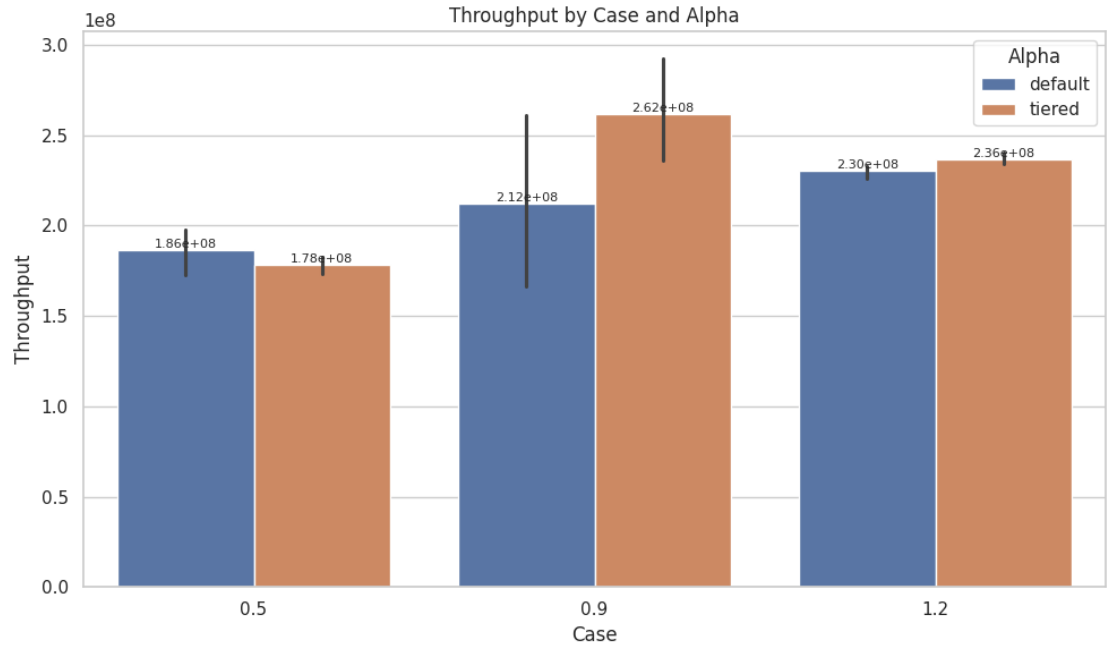
실험 결과

실험1 결과

Latency



Throughput



➡ alpha 값이 커지면 성능 향상, default와 tiered 성능 비슷

03

본론2

- 가설 설정
- 실험 설계
- 실험 결과

가설 설정

가설 설정

Cold 분류 기준에 따라 Time-Aware Tiered Storage의 효과가 달라질 수 있으며,

특히 preclude_last_level_data_seconds의 값이 커지면 WAF의 감소로 쓰기 성능이 향상될 것이다.

조건	기대 결과
Preclude 값이 작을 때 (즉시 compaction)	Level-N compaction 빈번 → WAF 증가, 디스크 사용량 증가, 쓰기 시간 증가
Preclude 값이 클 때 (compaction 지연)	SST 파일이 상위 레벨에 오래 유지 → Compaction 횟수 감소 → WAF 감소, 쓰기 성능 향상 가능

- SST 파일이 마지막 레벨로 내려가기까지의 조건을 시간 기반으로 제어
- 모든 Level-N SST가 동일한 HDD에 저장, preclude 값이 WAF에 미치는 영향 중심 분석

실험 설계

주요 파라미터

변수	설명
α (alpha)	Zipf 분포의 skew 정도 (0.5, 0.9, 1.2)
preserve_internal_time_seconds	Time-Tracking 시 몇 초간 시간 정보를 유지할지 지정
preclude_last_level_data_seconds	설정 시간보다 최근 데이터는 마지막 레벨로 이동되지 않도록 제한
temperature	마지막 레벨에 적용되는 온도 (모든 실험에서 cold로 고정)

실험 설계

주요 파라미터

	0%	20%	50%	80%	100%
0.5	0	20	50	80	120
0.9	0	15	40	65	120
1.2	0	10	25	40	120

	time	temperature
0.5	0, 20, 50, 80, 120	cold
0.9	0, 15, 40, 65, 120	cold
1.2	0, 10, 25, 40, 120	cold

- Level-N에 대해 다른 디바이스를 지정하지 않음 (HDD에 저장)
- Zipfian의 Alpha에 따라 workload access skew가 변화
- preserve_internal_time_seconds과 preclude_last_level_data_seconds는 동일

실험 결과

실험 환경

OS	Ubuntu 20.04.6 LTS (64bit)
CPU	16 * Intel(R) Core(TM) i7-10700F CPU @ 2.90GHz
RAM	31.2GiB
SSD	1.0TB
RocksDB	Version 10.2.0

실험 결과

실험2 결과

SST 파일 개수

Alpha=0.5

case	cold	비고
0	O	평균 4
20	O	일부 3
50	X	
80	X	
100	X	

Alpha=0.9

case	cold	비고
0	O	평균 3
20	O	평균 3
50	O	평균 2
80	X	
100	X	

Alpha=1.2

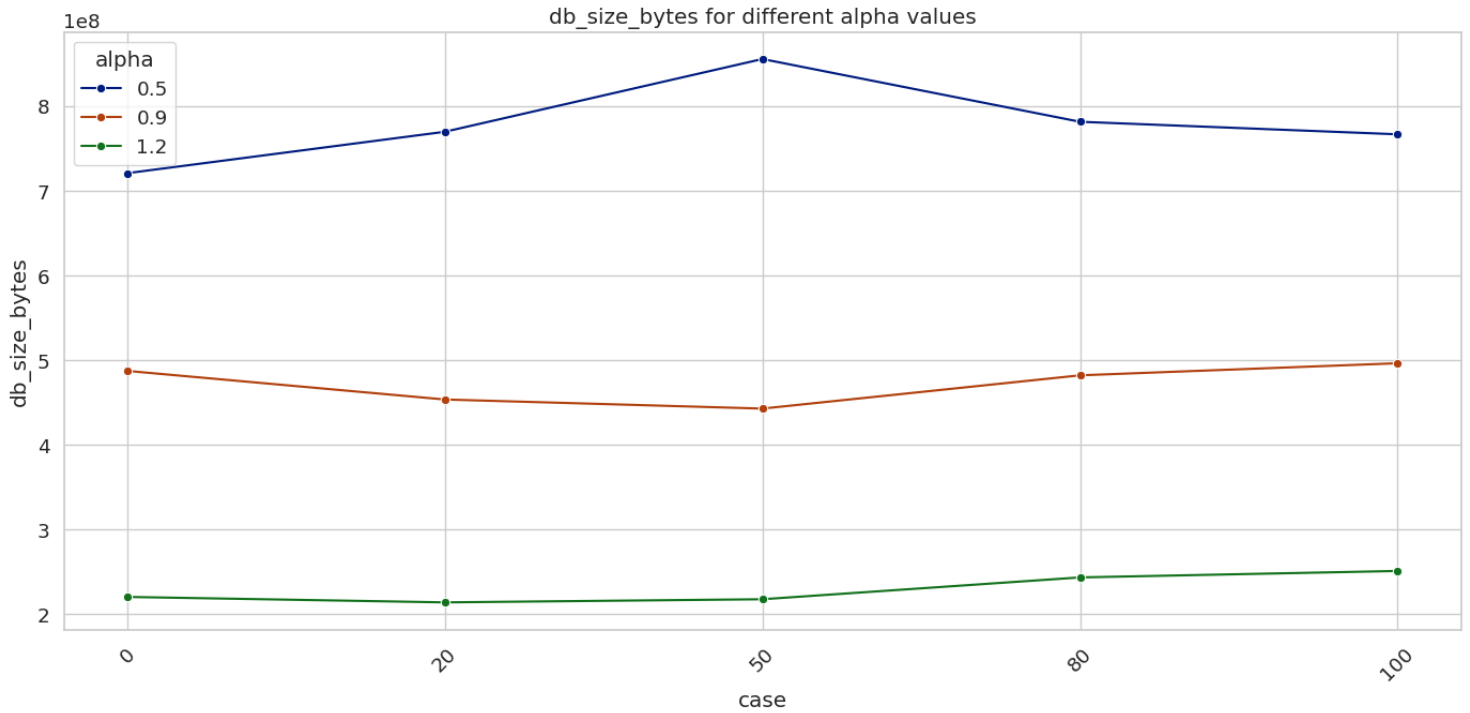
case	cold	비고
0	O	1
20	O	1
50	O	1
80	O	1
100	X	

➡ preclude_last_level_data_seconds가 커질수록 Last Level에 SST가 적거나 없다

실험 결과

실험2 결과

DB Size

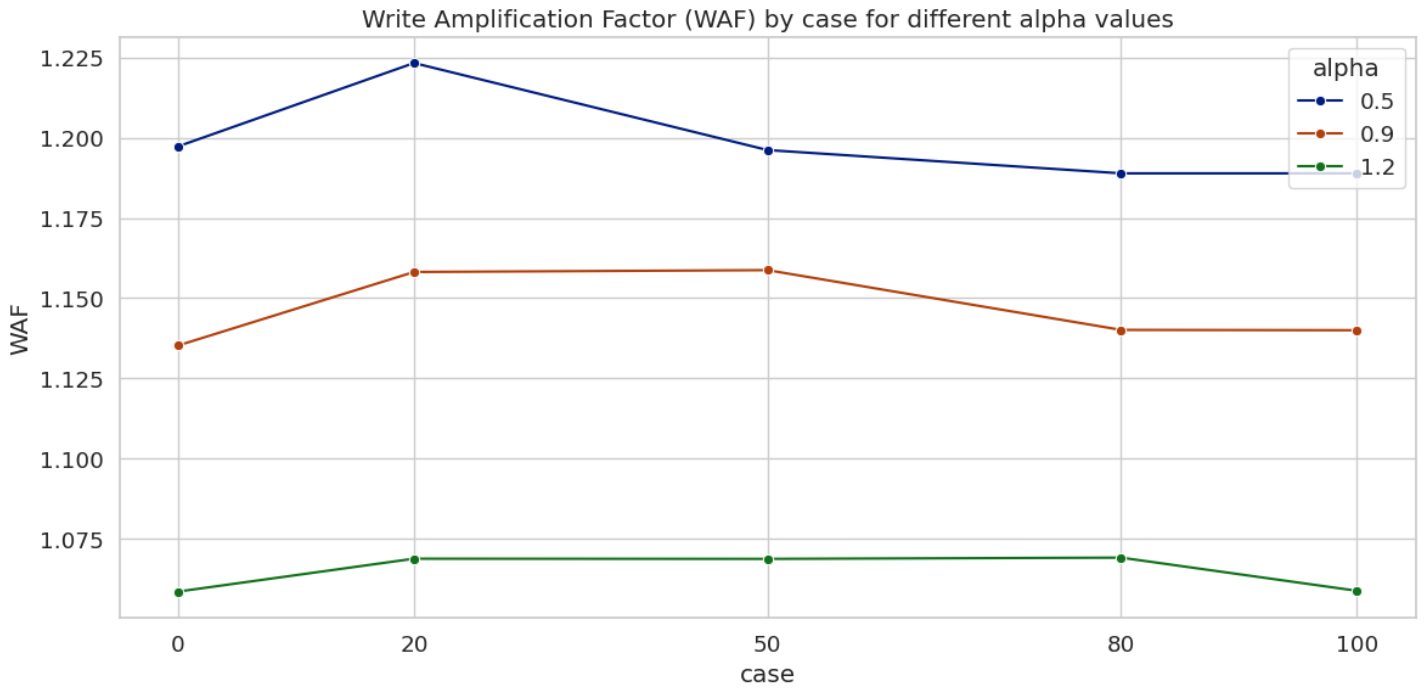


alpha가 낮을수록 데이터베이스 사용량이 많다

실험 결과

실험2 결과

WAF



Alpha가 낮으면 WAF 높음

Case 20: WAF 약간 증가

Case 50 이상: WAF 감소, 안정

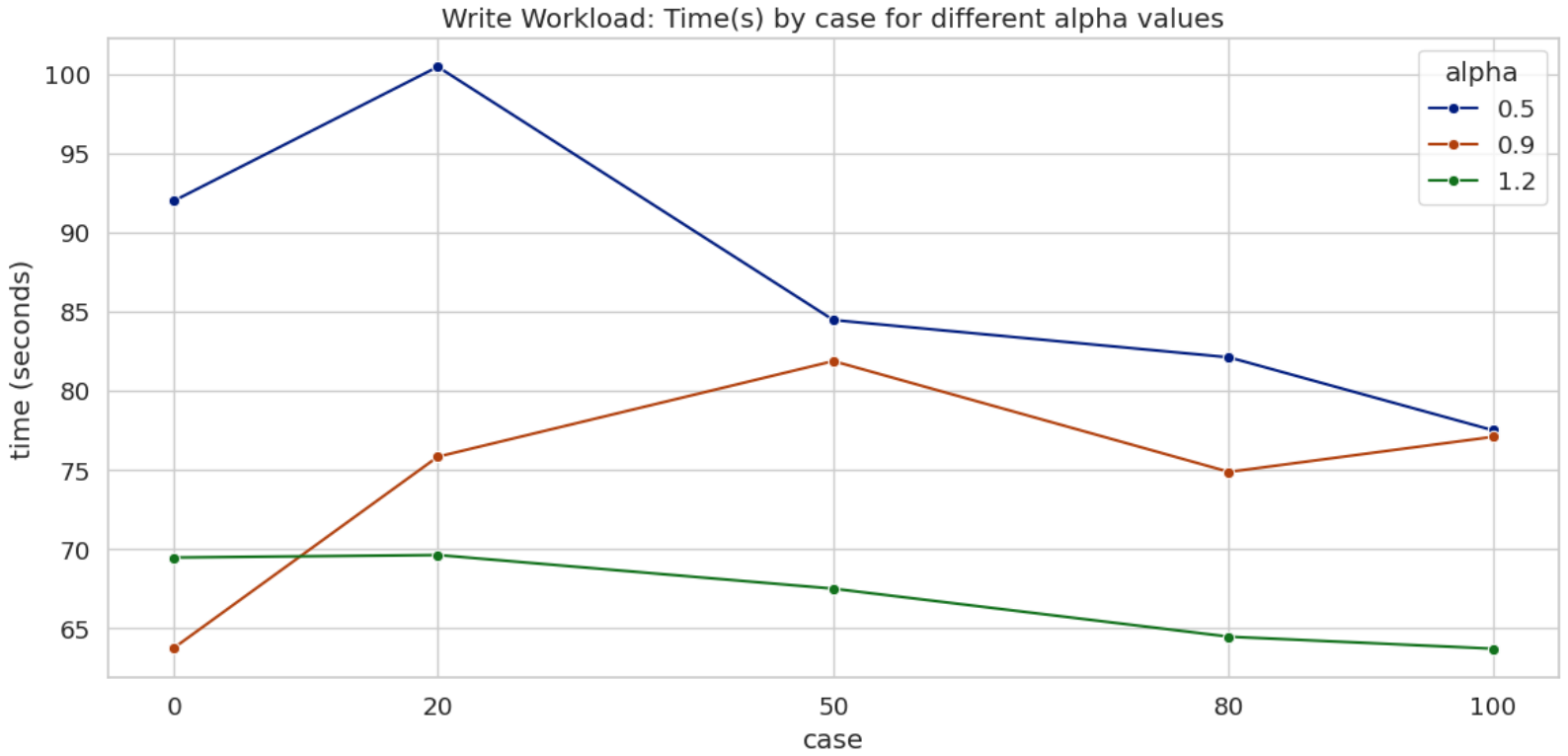
전체적으로 큰 변화 X

Alpha가 높으면 WAF에 큰 영향 X

실험 결과

실험2 결과

Time

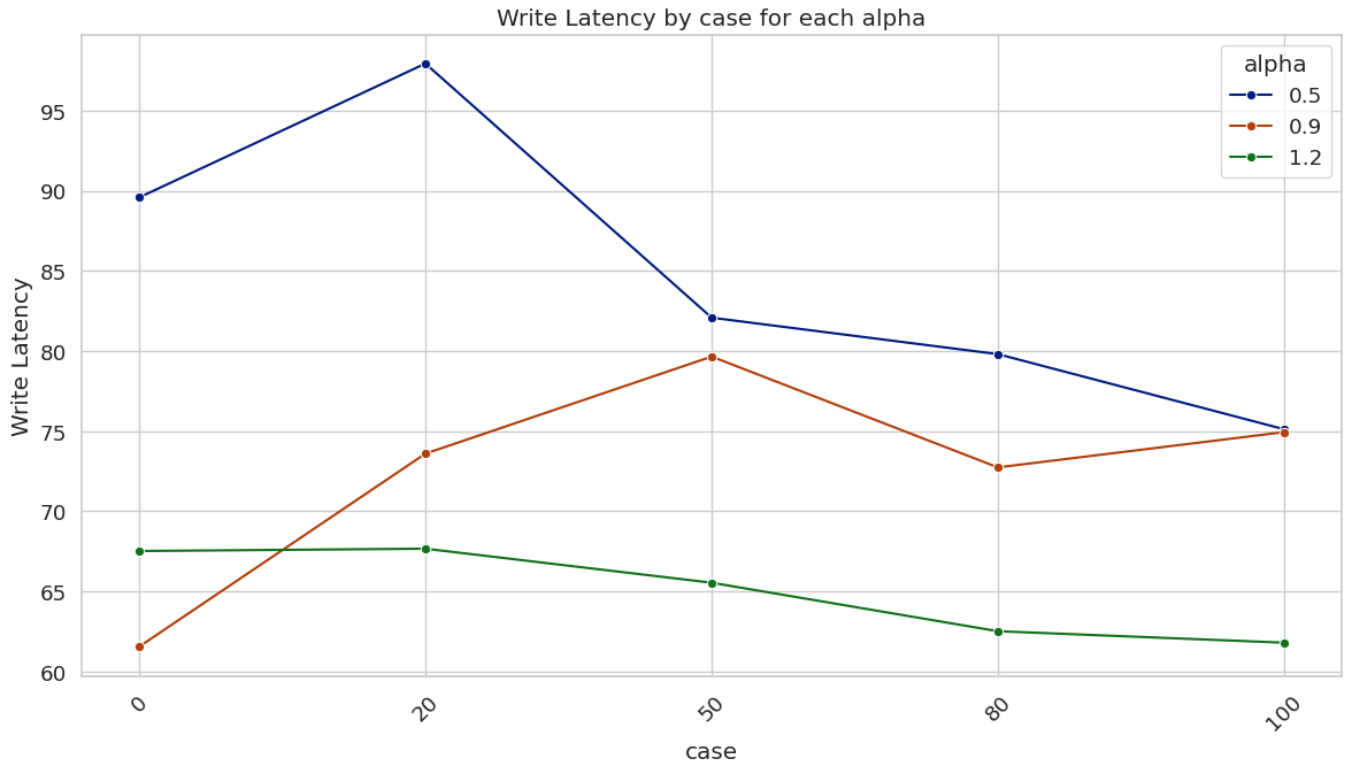


Compaction, stall 증가 ⟷ Stall, flush 지연 감소

실험 결과

실험2 결과

Latency

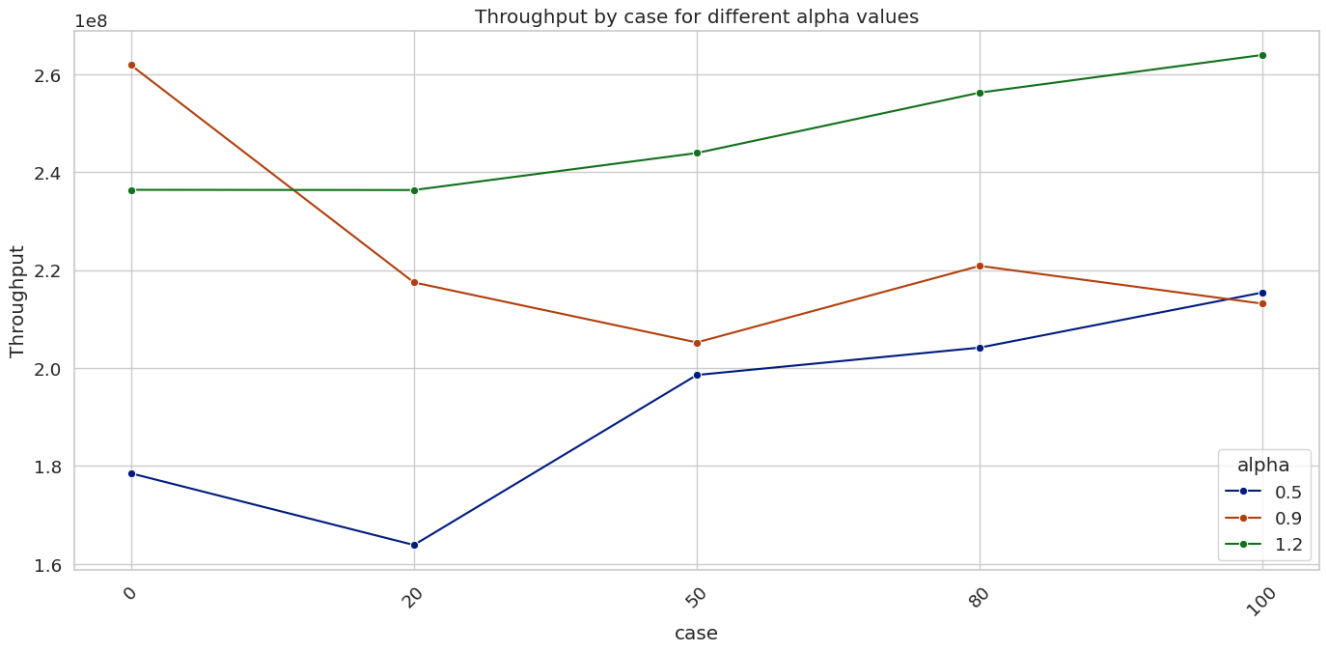


Compaction, stall 증가 ⟷ Stall, flush 지연 감소

실험 결과

실험2 결과

Throughput



Alpha = 0.5, 1.2 : throughput 증가

Alpha = 0.9 : throughput 감소

Preclude의 값이 크면 throughput 증가

중간 skew에서 복잡한 패턴을 가짐

04

결론

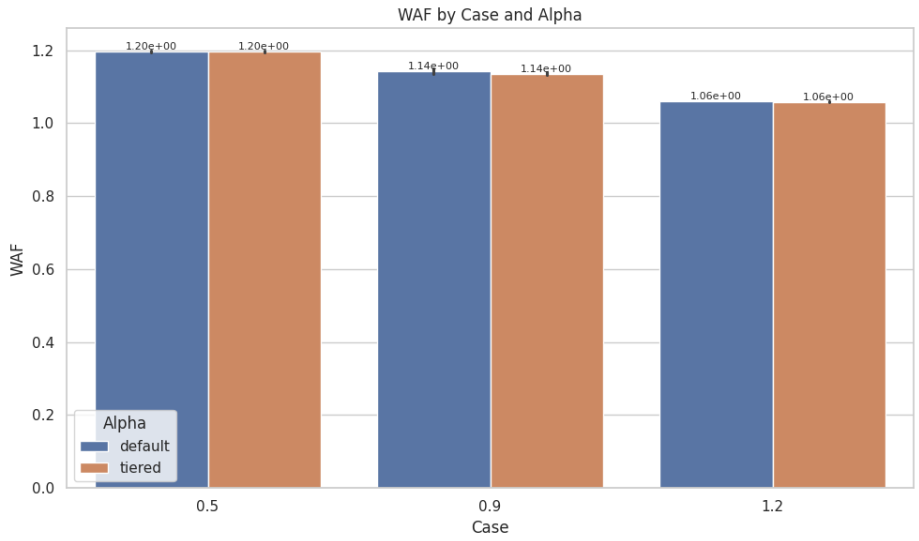
- 고찰
- 향후 연구

고찰

실험1 결과 요약

Universal Compaction만 사용한 경우와 비교하여

Time-Aware Tiered Storage (TATS)는 오버헤드가 발생하여 쓰기 성능이 더 낮게 나올 것이다



Alpha 값이 클수록 성능이 좋다

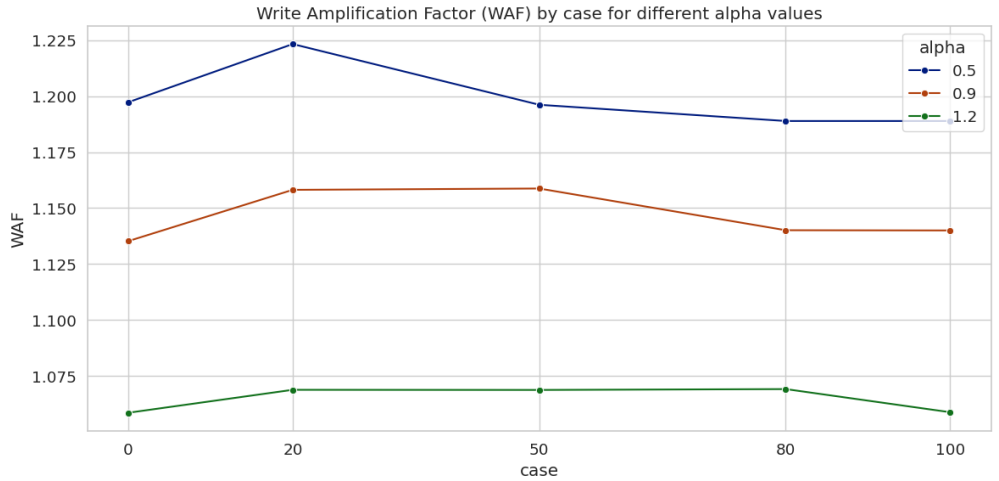
Default보다 TATS의 성능 차이가 없다

고찰

실험2 결과 요약

Cold 분류 기준에 따라 Time-Aware Tiered Storage의 효과가 달라질 수 있으며,

특히 preclude_last_level_data_seconds의 값이 커지면 WAF의 감소로 쓰기 성능이 향상될 것이다.



preclude_last_level_data_seconds 증가

WAF 증가 후 감소, But 영향도가 크지 않음

전반적인 쓰기 성능 향상

Alpha 값이 높을수록 영향도가 적음

고찰

고찰

workload
• Hot 데이터 집중도가 높지 않은 경우 TATS 유의미 • Workload 특성에 따라 preclude time 최적화 가능

실험 한계
• 실험은 단일 디바이스 (HDD)를 기준으로 측정 • Hot/Cold를 설정하여 SSD/HDD 혼용 실험 필요

향후 연구

향후 연구

스토리지 변경 실험

- 본 실험은 단일 디바이스 (HDD)를 기준으로 측정
- Hot/Cold를 설정하여 SSD/HDD 혼용 실험 진행
- 전반적인 I/O 성능 향상 기대

05

참고 문헌

참고문헌

[1] 신수환, 최건희, 유시환, 최종무. (2023). RocksDB의 시간 인식 기반 계층형 스토리지 분석. 2023 한국소프트웨어종합학술대회 논문집, pp.1946-1948.

[2] 2022-11-09-time-aware-tiered-storage.markdown. (2022). https://github.com/facebook/rocksdb/blob/main/docs/_posts/2022-11-09-time-aware-tiered-storage.markdown.

[3] Advanced_options.h. (2025). https://github.com/facebook/rocksdb/blob/main/include/rocksdb/advanced_options.h.

[4] Differences between Hot Data and Cold Data - System Design. (2024). https://www.geeksforgeeks.org/differences-between-hot-data-and-cold-data-system-design/.

[5] iostats_context.h. (2024). https://github.com/facebook/rocksdb/blob/main/include/rocksdb/iostats_context.h.

[6] statistics.h. (2022). https://github.com/facebook/rocksdb/blob/main/include/rocksdb/statistics.h.

[7] Tiered Storage (Experimental). (2023). https://github.com/facebook/rocksdb/wiki/Tiered-Storage-(Experimental).

[8] Time-Aware Tiered Storage in RocksDB. (2022). https://rocksdb.org/blog/2022/11/09/time-aware-tiered-storage.html.

[9] Universal Compaction. (2023). https://github.com/facebook/rocksdb/wiki/universal-compaction.

Thank you

Thank you for listening to my presentation