Week#2 SST file size of RocksDB

Sangeun Chae

2018314760

# INTRODUCTION

RocksDB는 LSM tree의 형태로 데이터를 저장한다. RocksDB는 크게 메모리 상에 저장되는 memtable과 디스크(SSD)에 저장되는 SSTable, 그리고 Log 영억으로 이루어져 있다. SSTable은 memtable이 가득차서 더 이상 변경될 수 없는 immutable상태로 변환된 데이터가 디스크로 이동하여 저장되는 형태이다. 이 과정에서 레벨 0에서 정해진 SSTable의 파일 크기가 특정 임계 값을 넘어가게 되면, 레벨1로 중복되는 KV의 값을 병합(Compaction)하는 과정을 거친다. 이 과정에서, 데이터가 실제 스토리지에 쓰여지는 양에 비해, 더 많은 쓰기 작업이 발생하게 되고, 이를 write amplification이라 한다. 따라서 이번 랩에서는, SST file의 크기에 따른 write amplification의 값을 측정해보고, 원인을 분석할 예정이다.

# METHODS

2MB, 8MB 16MB의 크기 순서대로 SST file의 크기를 설정하고, 메모리 버퍼인 memtable의 크기도 SST file의 크기와 동일하게 설정하여, 3번의 DB Bench를 수행한다. 각각의 DB Bench를 통해 생성된 각각의 LOG file을 저장하고, 각각의 LOG file을 비교 대조한다.

# Performance Evaluation

## Experimental Setup

|  |  |
| --- | --- |
| **Type** | **Specification** |
| OS | Ubuntu 20.04.3 LTS |
| CPU | AMD Ryzen 7 5800X 8-Core Processor (VMware support 4 Core) |
| Memory | 4GB |
| Kernel | Linux ubuntu 5.11.0.34 -generic |
| Disk | VMware Virtual 80GB |

Table 1: System setup

|  |  |
| --- | --- |
| **Type** | **Configuration** |
| Bench Type | Read random write random |
| Direct flush compaction | True |
| Direct read | True |
| Duration | 3600s |
| Max byte for level base | 33554432 Bytes |
| Max bytes for level multiply | 5 |
| SST file size | 2M, 8M, 16M each |
| Memory buffer size | 2M, 8M, 16M each |

Table 2: Benchmark setup

## Experimental Results

**3.1 Results**

Figure 1: Write Amplification per SST file size

**[Figure 1]**은 SST file size별로 모든 level에서 일어나는 write amplification이 일어나는 양의 합계를 그래프로 나타낸 것이다. SST file의 크기가 2M일 때 9.6GB, 8M일 때 3.6GB, 마지막으로 16M 일 때 2.5GB 만큼 write amplification이 일어난다.

Figure 2: Read and Write Quantity in Disk

**[Figure 2]** 은 SST file size별로 모든 level에서 일어나는 read와 write의 양의 합계를 그래프로 나타낸 것이다. SST file의 크기가 2M일 때 read와 write가 각각 2.2GB, 2.2GB, 8M일 때 read와 write의 크기가 각각 0.7GB, 0.6GB, 마지막으로 16M일 때 read와 write의 크기가 각각 0.5GB, 0.4GB 만큼 일어난다.

**3.2 Analysis**

**[Figure 1]**의 결과를 분석하면, SST file size가 증가할수록, write amplification되는 양이 줄어든다. 하지만 compaction과정에서는, SST file의 크기가 증가할수록, 쓰여지는 데이터의 크기가 증가한다. 그 이유는, overlap된 키가 존재하는 모든 SST file에 대한 쓰기 작업이 이루어져야 하기 때문이다. 따라서 SST file size이 비례하여 write amplification이 증가한다. 아래의 그림은 해당 과정의 예시이다.

텍스트, 시계이(가) 표시된 사진

자동 생성된 설명

Figure 3: Compaction (Write Amplification)

**[Figure 3]** 은 compaction이 일어나는 과정을 도식이다. Level 2에서 50에서 100 사이의 key를 저장하고 있는 SST file이 level 3에 compaction되는 과정에서, 50에서 100의 키를 저장하고 있는 level3의 모든 SST file에 대해서 쓰기 작업이 이루어진다. 따라서 SST file size에 비례하여 write amplification이 증가하는 것이다.

하지만 이번 랩의 실험결과는 위의 원리와 상반되는 결과를 보여준다. 그 이유는 memory buffer size가 write amplification 감소의 원인으로 작용하기 때문이다. 해당 실험에서는, memory buffer의 크기를 SST file size와 동일하게 설정했다. 따라서, SST file의 size가 클수록, memory buffer의 크기도 커지고, memory buffer hit ratio가 증가하게 된다. 이에 대한 결과는 **[Figure 2]**에서 확인할 수 있다. SST file의 크기가 증가할 수록, Disk의 Read와 Write의 양이 줄어듦을 알 수 있다. 이는 Memory buffer의 hit ratio가 높다는 것과 직결한다.

Figure 4: Compaction Time per SST file size

Figure 5: Compaction Count per SST file size

Memory buffer에서 hit되는 데이터가 많아 질수록, Memory buffer flush가 적게 일어나며, 이는 Compaction 횟수의 감소에 직결된다. 이에 대한 근거로, **[Figure 4], [Figure 5]** 가 있다. [Figure 4]를 보면 SST file size가 증가할수록, Compaction 과정에 수행되는 시간이 감소한다. 또한, **[Figure 5]**을 통해 알 수 있듯이, SST file size가 증가할수록, Compaction이 수행되는 횟수가 감소함을 알 수 있다.

결론적으로, write amplification에 영향을 미치는 요소 중, SST file size크기에 비해 memory buffer hit ratio의 영향이 더 컸기 때문에 이러한 결과가 나왔음을 알 수 있다.

# Conclusion

Write amplification은 실제로 스토리지에 쓰이는 데이터에 비해, 쓰기 작업이 많이 발생하기 때문에 생긴다. Write amplification이 일어나는 원인에는 다양한 이유가 있지만, 이번 랩에서는 SST file의 크기가 어떠한 영향을 끼치는지 알아보았다. 하지만, SST file의 크기와 memory buffer의 크기를 동등하게 설정하여 DB Bench를 수행한 결과, SST file의 크기의 증가가 write amplification의 증가에 끼치는 영향에 비해 memory buffer에서 hit하는 데이터의 양의 증가가 write amplification의 감소에 끼치는 영향이 더욱 크게 작용했다. 다시 말해, SST file size에 write amplification의 정도가 비례하는 것에 비해, Memory buffer size 증가에 감소하는 write amplification의 양이 더욱 컸음을 알 수 있다.

# REFERENCES

1. <https://github.com/meeeejin/SWE3033-F2021/blob/main/report-submission-guide.md>