RocksDB 에서 SST 파일에 따른 WAF 감소에 관한 연구

조민수*, 최원기*, 박상현*†
*연세대학교 컴퓨터과학과
{minsoo0104, cwk1412, sanghyun}@yonsei.ac.kr

A Study on WAF reduction and SST file size on RocksDB

Minsoo Cho*, Wongi Choi*, Sang Hyun Park*†
*Dept of Computer Science, Yonsei University

요 약

RocksDB 는 Facebook 에서 LevelDB 를 기반으로 개발한 임베디드 key-value 스토리지 엔진이다. Log structured tree(LSM-tree) 를 기본구조로 사용하는 RocksDB 는 데이터를 레벨단위로 저장한다. 지속적인 데이터 입력으로 인하여 레벨의 크기를 초과하게 되면 하위 레벨의 SST 파일과 병합을 통해 하위레벨로 내려 보낸다. 이 과정에서 디바이스의 부가적인 쓰기가 발생한다. 본 논문에서는 RocksDB 의 디스크영역에 있는 SST 파일의 크기가 디바이스의 쓰기 증폭에 미치는 영향을 분석하였다. SST 파일크기변화에 따른 디바이스의 쓰기 증폭과 성능변화를 측정하고 비교하였다. 실험결과를 통해 SST 의 크기가 작을수록 쓰기 증폭이 줄었지만 디바이스의 쓰기와 읽기 성능이 감소하는 것을 확인하였다. 결과적으로 쓰기 증폭을 줄이고 성능을 최대화 하기 위해서는 이 둘의 트레이드오프를 파악하고 분석하여 시스템에 맞는 최적의 SST 파일 크기를 찾아야한다.

* 본 연구는 과학기술정보통신부 및 정보통신기술진흥센터의 SW 컴퓨팅산업원천기술개발사업(SW 스타랩)의 연구결과로 수행되었음 (IITP-2017-0-00477).

† 교신 저자 : sanghyun@yonsei.ac.kr

1. 서론

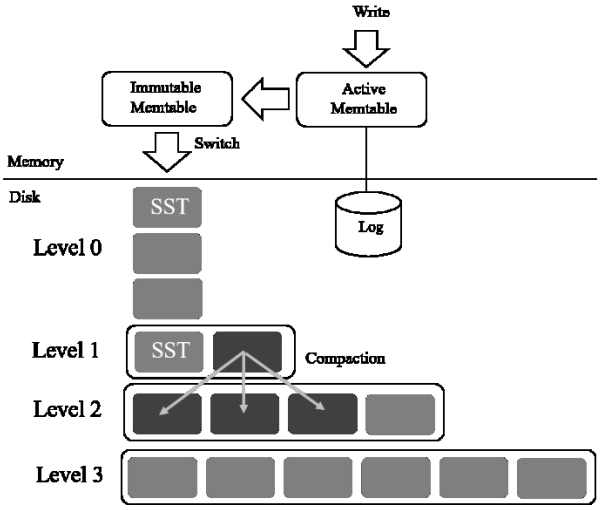
DBMS 는 모든 산업영역에서 데이터를 관리하고 저장하기 위한 필수 소프트웨어로 사용되고 있다. 최초로 개발된 네트워크 DBMS 에 이어 계층 DBMS, 관계형 DBMS, 객체지향 DBMS 와 같은 다양한 종류의 DBMS 가 계속해서 개발되고 있다. 최근에는 빅데이터라는 IT 의 새로운 물결이 DBMS 의 변화를 일으켰다. 그전까지는 2 차원 테이블 형태로 데이터를 관리하는 관계형 데이터 모델이 가장 많이 사용되었지만, 비정형 데이터의 비중이 커져 새로운 데이터베이스 시스템인 NoSQL 이 등장하게 된다. NoSQL 은 기존의 RDBMS 보다 덜 제한적인 데이터 모델을 사용하면서 대규모 데이터 처리에 적합하다. NoSQL 에는 다양한 데이터 구조를 사용할 수 있는데, 본 논문은 여러 데이터 구조 중에서도 가장 단순한 형태인 Key Value 저장방식을 사용하는 RocksDB 에 대해 설명한다.

RocksDB 는 구글의 LevelDB[2]를 기반으로 개발된 Facebook 의 오픈소스 프로젝트이다. RocksDB 는 Log Structured Merge Tree(LSM-trees)[3] 구조를 기반으로 한다. LSM-tree 구조의 경우, 메모리 버퍼인 memtable 이 차게 되면 디스크로 파일들을 내려 보내 디스크에 저장한다. 이때 디스크에 들어온 파일들을 순차적으로 저장하기 위해 하위 레벨에 있는 파일과 병합 정렬을 하게 되는데 이 과정에서 쓰기 증폭(Write Amplification[4])이 발생하게 된다. SSD 와 같은 플래시 메모리는 메모리의 특성상 쓰기 횟수에 제한이 있기 때문에 불필요한 쓰기에 더욱 취약하다. 실제 이와 관련하여 플래시 메모리의 쓰기 증폭을 줄이기 위한 많은 연구가 진행 중에 있다. 본 논문에서는 RocksDB 의 디스크를 구성하는 SST(Sorted Static Table)파일의 크기에 따른 쓰기 증폭과 측정하여 SST 파일의 크기가 디바이스에 미치는 영향을 분석한다. 더 나아가, 쓰기와 읽기 성능을 측정하여 SST 파일의 크기 변경이 쓰기 성능 뿐 아니라 읽기 성능에 어떠한 영향을 미치는지 분석하였다.

본 논문의 구성은 다음과 같다. 2 장에서는 LSM tree 구조인 RocksDB 구조에 대해 설명하고, 3 장에서는 RocksDB 의 데이터 입력과 그로 인한 쓰기 증폭에 대해 설명한다. 4 장에서는 실험 환경을 살펴보고 실험 결과를 분석한다. 5 장에서는 결론을 내리면서 마무리한다.

2. 관련 연구

2.1 RocksDB



[그림 1] RocksDB Architecture

RocksDB 는 LSM trees 구조를 사용하여 데이터를 저장한다. LSM-Tree 의 구조를 사용하는 RocksDB 의 구조는 [그림 1]과 같다. 메모리 영역, Log 영역 그리고 디스크 영역으로 나뉜다. 메모리 영역에는 memtable, Log 영역에는 Log 파일, 그리고 디스크영역에는 SST 파일이 존재한다.

Memtable 은 인메모리 쓰기 버퍼로 새로 삽입된 데이터를 임시로 저장한다. 스토리지에 있는 Log 파일에도 데이터에 관한 정보가 순차적으로 기록된다. Memtable 이 가득 차면 memtable 과 Log 파일은 더 이상 삽입될 수 없는 immutable 상태로 변경되고 새로운 memtable 과 log 파일이 할당된다. Immutable 상태로 변경된 memtable 은 디스크로 이동되어 레벨 0 에 있는 SST 파일로 저장된다. SST 파일은 디스크를 구성하고 있는 데이터 파일로 키-값을 저장하고 있으며 그 키 값에 따라 각 레벨에서 정렬된다. 레벨 0 에서 정해진 SST 파일의 개수가 초과되면 레벨 0 과 중복된 키 값을 갖는 레벨 1 의 SST 파일이 선택되어 병합 정렬을 하여 하위레벨로 내려 보낸다. 이러한 과정을 컴팩션이라고 한다. 나머지 하위 레벨에서도 마찬가지로 각 레벨의 크기가 초과되면 컴팩션이 발생한다. 하위 레벨에서의 컴팩션은 다음절인 Write Amplification 에서 보도록 한다.

2.2 Write Amplification

컴팩션 과정을 살펴보면, 데이터를 스토리지에 쓰기 위해서는 실제 스토리지에 쓰여지는 데이터의 양보다 많은 쓰기 작업이 발생한다. 이와 같은 부가적인 쓰기 발생을 쓰기 증폭(Write Amplification)이라고 한다. 실제 쓰기 증폭은 스토리지에 쓰여진 데이터의 양을 데이터베이스에 쓰여진 바이트로 나눈 값으로 아래의 수식(1)과 같이 나타낼 수 있다.

$$\text{Write Amplification Factor} = \frac{\text{Bytes written to Storage}}{\text{Bytes written to Database}} \quad (1)$$

쓰기 증폭이 증가하면 처리해야할 쓰기의 수가 증가하여 디바이스 입출력 성능이 감소한다. 또한, SSD 와 같은 플래시 메모리는 쓰기 횟수가 제한적이기 때문에 쓰기 증폭이 발생함에 따라 SSD 수명이 단축된다. 따라서 RocksDB 의 컴팩션 과정에서 쓰기 증폭을 최소한으로 만드는 것이 중요하다.

현재 RocksDB 는 Leveled 컴팩션을 디폴트 컴팩션으로 제공하고 있으며 본 논문에서는 Leveled 컴팩션에서의 쓰기 증폭을 다룬다. Leveled 컴팩션에서 디스크의 파일은 레벨 0 부터 레벨 N 으로 구성되어 있다. 레벨 N 을 구성하고 있는 모든 SST 파일의 크기가 레벨 N 의 크기를 초과하면 레벨 N 에서 하나의 SST 파일이 선택된다. 선택된 SST 파일은 자신의 키와 키의 범위가 중복된 키를 가지고 있는 레벨 N+1 의 SST 파일과 병합된다. 이 과정에서 쓰기 증폭이 발생하는데, 이는 레벨 N 에 있는 하나의 SST 파일을 하위 레벨의 파일과 병합하기 위해 레벨 N+1 에 있는 여러 파일에 대해 다시 쓰기작업을 해야 하기 때문이다.

예를 들어, 파일의 크기가 100MB 일 때 레벨 N 의 키가 레벨 N+1 에 있는 5 개의 파일과 중복되면, 약 500MB 의 데이터가 다시 쓰여져야 하므로 쓰기 증폭은 5 가 된다.

3. 실험 결과 및 분석

3.1 SST 파일 크기에 따른 평가

RocksDB 의 디스크에 있는 SST 파일의 크기가 디바이스의 쓰기 증폭에 중요한 요소로 미칠 것으로 예상하였다. 실제 SST 파일의 크기가 디바이스의 쓰기 증폭과 쓰기, 읽기 성능에 미치는 영향을 알아보기 위해 SSD 디바이스에 SST 파일의 크기를 Kilo 바이트 단위에서 Mega 바이트 단위까지 다양하게 변경하여 실험하였다.

3.2 실험 환경

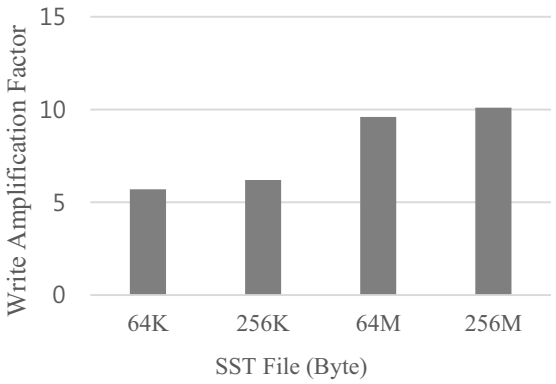
[표 1] 실험 환경

OS	CentOS 7.3.1611(x86_64)
CPU	Intel(R) Core(TM) i7-6700K CPU @ 4.00GHz
RAM	64GB
SSD	SAMSUNG 850 PRO 256GB

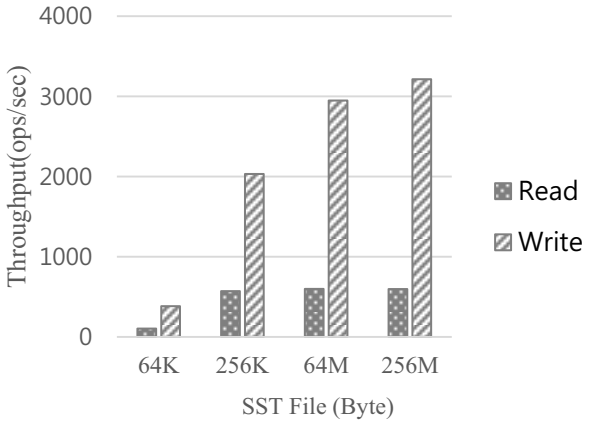
SSD 의 Write Amplification 과 읽기, 쓰기 성능을 측정하기 위해서 DBbench 벤치마크를 사용하였다. DBbench 는 RocksDB 의 성능을 측정하는데 사용하는 대표적인 벤치마크 툴로, 옵션을 통해 다양한 워크로드를 생성하고 동작 시킬 수 있다. 본 실험에서는 키-값을 각각 8192 바이트, 32 바이트로 설정하고 400 만개의 키-값을 입력 받아 약 12GB 의 데이터로 측정하였다. DBbench 의 옵션 중 target 파일 사이즈 옵션을 바꾸어 SST 파일의 크기를 설정하였다. 실험환경은 위의 [표 1]과 같다.

또한, 이 실험에서는 일반적으로 시스템에서 파일을 입출력 할 때 사용되는 Buffered I/O 방식이 아닌 Direct I/O 방식으로 동작하게 설정하였다. Buffered I/O 방식은 OS 자체의 페이지 버퍼 캐시를 사용하여 읽기와 쓰기 작업에 대한 파일을 버퍼 캐시에 일시적으로 저장시키는 방법이다. 버퍼 캐시를 사용하면 빈번하게 발생하는 I/O 를 한번에 처리할 수 있기 때문에 효율적으로 파일의 읽기, 쓰기 작업을 할 수 있다. 그러나 Buffered I/O 방식이 모든 시스템에서 좋은 성능을 내지 않는다. 이 실험에서는 RocksDB 의 SST 파일 크기가 스토리지의 성능에 끼치는 영향을 직관적으로 판단하기 위해 페이지 버퍼 캐시를 사용하지 않고 파일 시스템에서 바로 파일을 입출력 하는 Direct I/O 방식으로 실험하였다.

3.3 실험 결과 및 분석



[그림 1] DBbench WAF 실험 결과



[그림 2] DBbench 성능 결과

SSD 에서 SST 파일 크기 변화에 따른 쓰기 증폭을 측정한 결과는 [그림 1]과 같다. 측정 결과를 보면 SST 파일의 크기가 감소함에 따라 쓰기 증폭이 감소하는 것을 확인 할 수 있다. SST 파일의 크기가 64M, 256M 바이트 일때보다 64K, 256K 일 때 크게 감소하

였다. SST 파일의 크기가 작을수록 레벨에 존재하는 파일의 개수가 증가한다. 파일의 크기가 작을수록 파일의 키 범위도 줄어든다. 컴팩션이 발생할 때, 하위 레벨에서 키 범위가 중복되는 파일의 개수가 확률적으로 적어지므로 SST 파일의 크기가 작을수록 쓰기 증폭이 감소한다. 따라서 [그림 1]의 실험 결과와 같이 SST 파일의 크기가 작아지면 쓰기 증폭이 덜 발생한다.

SSD 에서 SST 파일 크기 변화에 따른 쓰기와 읽기 성능을 측정한 결과는 [그림 2]과 같다. 쓰기 증폭과 마찬가지로 읽기와 쓰기의 성능 또한 SST 의 파일 크기가 감소함에 따라 더 낮은 성능을 보였다. 64K 바이트 크기의 SST 파일은 다른 크기의 SST 파일보다 쓰기와 읽기 성능이 현저하게 떨어지는 것을 확인 할 수 있다. SST 파일의 크기가 작아지면 각 레벨당 존재하는 파일의 개수가 증가하게 된다. 파일 개수의 증가로 인해 디스크 입출력의 효율성이 떨어지게 되므로 성능이 저하된다. 결론적으로 그림 2 의 결과와 같이 SST 파일의 크기를 감소하면 쓰기와 읽기 성능이 떨어지게 된다.

[그림 1]과 [그림 2]의 결과를 종합적으로 보면, SST 파일의 크기가 64 K 바이트 일 때 쓰기 증폭이 가장 적게 발생하지만 쓰기와 읽기 성능은 제일 저하된다. 이것으로 보아 쓰기 증폭을 줄이기 위해 파일의 크기를 최대한으로 감소시키는 것 보다 256K 바이트와 같이 쓰기 증폭이 적당히 작으면서 어느 정도의 쓰기와 읽기 성능을 낼 수 있는 최적의 SST 파일 크기를 찾는 것이 중요할 것이라고 분석되었다.

4. 결론

본 논문에서는 RocksDB 의 쓰기 증폭을 줄이기 위한 방법으로 SST 파일 크기를 변경하여 실험하였다. 그 결과, SST 파일 크기의 감소에 따라 쓰기 증폭이 줄었지만 디바이스의 쓰기와 읽기 성능도 같이 떨어졌다. SST 파일 크기가 작을 때 생기는 쓰기와 읽기 성능 감소 문제를 해결하기 위해서는 쓰기 증폭과 성능 감소의 트레이드오프를 고려하여 시스템에 맞는 최적의 SST 파일 크기를 찾아야 한다. 그러기 위해서는 더 다양한 실험 환경에서 SST 파일의 크기를 변경하여 실험해야 한다. 향후에는 Buffered I/O 방식에서 SST 파일 크기에 따른 쓰기 증폭과 성능을 연구하고자 한다.

참고문헌

[1] RocksDB, https://github.com/facebook/rocksdb/wiki
[2] LevelDB, https://github.com/google/leveldb
[3] O'NEIL, Patrick, et al. The log-structured merge-tree (LSM-tree). *Acta Informatica*, 1996, 33.4: 351-385.
[4] DONG, Siying, et al. Optimizing Space Amplification in RocksDB. In: *CIDR*. 2017.