매니코어 환경에서 RocksDB의 확장성 연구

김형준 O 박성순 [1] 원유집
한양대학교 글루시스/안양대학교 [1]
imkim0625@hanyang.ac.kr, sspark@gluesys.com, yjwon@hanyang.ac.kr

Scalability study of RocksDB in manycore environment

Hyeongjun Kim O Sungsoon Park [1] Youjip Won
Hanyang University Gluesys/Anyang University [1]

요 약

RocksDB는 페이스북에서 개발한 key-value 엔진으로, 데이터를 효율적으로 관리하기 위해 자료구조로 Log-Structured Merge-tree를 사용하고 있다. 데이터를 저장할 때에 우선 메모리의 Memtable에 저장을 하고, Memtable이 꽉차게 되면 SST file로 변환하고 이를 디스크에 저장을 한다. 데이터를 탐색할 때는 Memtable, SST file순으로 탐색한다.

본 논문에서는 RocksDB에서 데이터 탐색을 할 때에 확장성 여부에 대해서 실험하고 확장성이 발생하지 않는 원인으로 다수의 스레드가 Memtable에 접근할 때 발생하는 lock contention과 물리 페이지에 존재하는 데이터를 다수의 스레드가 접근할 때 발생하는 lock contention으로 분석한다. 그리고 이를 해결하기 위한 방법으로 sloppy counter와 같은 분산된 락 방식과 lock-free 방식을 제안한다.

1. 서 론

인터넷의 발전으로 인하여 소셜 네트워크 서비스가 활성화되고 있고, 전 세계적으로 방대한 양의 데이터가 생성이 되고 있다. 방대한 데이터를 관리하기에는 기존에 관계형 데이터베이스는 여러 한계가 드러났다. 대안으로 NoSQL이라는 새로운 데이터베이스들이 만들어지고 있다.

대표적인 소셜 네트워크 서비스를 제공하는 회사인 Facebook도 RocksDB[1]라는 NoSQL을 만들고 오픈소스로 공개하였다.

또한 컴퓨터의 발전으로 인하여 CPU의 성능과 코어의 수가 증가하고 있다. 이로 인하여 여러 작업들을 빠르고 정확하게 처리하는 것이 가능해졌다. 하지만 코어의 수가 증가하여도 컴퓨터의 성능이 코어의 수만큼 비례하여 증가하는 것은 아니다. 이 문제를 해결한다면 우리는 컴퓨터의 자원을 보다 더 효율적으로 사용할 수 있을 것이다.

본 논문에서는 스레드의 수에 따른 readseq와 readrandom 실험을 통해 RocksDB의 데이터 읽기에 대한 확장성이 있는지에 대해서 실험을 하였다. 그리고 확장성에 대한 원인으로 lock contention 문제를 제기한다.

본 논문의 구성은 다음과 같다. 2장에서는 대표적인 LSM-Tree 구조의 key-value store인 RocksDB의 구조와 데이터 삽입과 읽기의 과정에 대해서 설명한다. 3장에서는 실험 환경과 실험 결과에 대해서 설명하고, 4장에서 결론 및 향후 연구로 마무리 한다.

2. RocksDB

RocksDB는 LevelDB[3]를 기반으로 만들어지고, LSM-Tree (Log-Structured Merge-tree) 구조를 가진 NoSQL의 한 종류인 key-value store이다. RocksDB의 구조는 Memory 영역의 Memtable과 디스크 영역의 SST(Static Sorted Table) file로 나눌 수 있다.

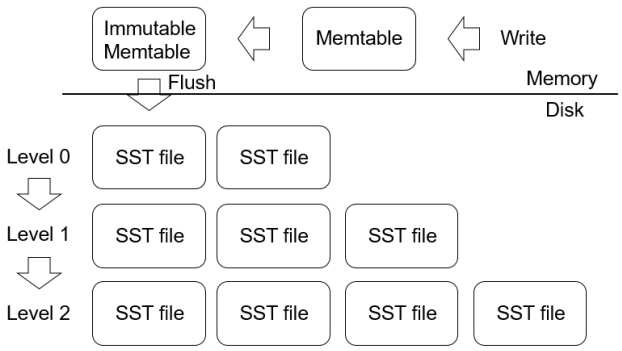

그림 1 RocksDB 구조

데이터 삽입이 발생하게 되면 Memory 영역의 Memtable에 key와 value 값이 저장된다. 데이터 삽입을 통해 Memtable이 가득차게 되면, Memtable은 데이터 변경이 불가능해지는 Immutable Memtable(Read-Only Memtable)로 변경이 되고 데이터 삽입을 위한 새로운 Memtable이 생성된다. 그리고 Immutable은 Level 0의 SST file로 변경되고 정렬이 된 상태로 디스크 영역으로 flush된다. 이후에 데이터 삽입이 발생하게 되어 Level 0의 SST file이 설정된 값 이상으로 증가하게 되면

Level 1의 SST file 중에서 중복된 키 값을 갖는 SST file과 병합 정렬을 통해 합쳐지게 된다. 이러한 과정을 Compaction이라 하고 나머지 하위 레벨에서도 위와 같은 Compaction이 발생한다.

데이터 읽기는 Memtable, Imutable Memtable, SST file 순서로 진행된다. 따라서 최근에 삽입된 데이터에 대해서는 읽기가 빠르지만 낮은 레벨의 데이터에 대해서는 읽기가 느리다는 단점이 있다.

3.1 실험 환경

본 논문에서는 스레드의 수가 증가함에 따라 RocksDB에서 데이터 읽기의 확장성 여부를 알아보기 위해 실험하였다. 벤치마크로는 RocksDB에서 제공하는 DBbench[2]를 사용하였다. DBbench는 RocksDB의 성능을 측정할 수 있는 대표적인 벤치마크이고, 여러 옵션을 통해서 다양한 워크로드에 대한 실험을 진행할 수 있도록 한다. 실험에 앞서 key-value entry의 수가 1000개이고, memtable이 2개이고 SST file이 1개인 DB를 생성하였다. key와 value의 크기는 각각 16byte와 100byte이다. 데이터 읽기만의 확장성 여부를 실험하기 위해 Read-Only로 하였다. 실험 워크로드는 readseq와 readrandom로, 2가지의 데이터 읽기에 대한 실험을 하였다.

표 1 실험 환경

CPU	120 * Intel® Xeon® CPU E7-8870 v2 @ 2.30GHz
RAM	755GB
SSD	256GB Samsung SSD 850
OS	Linux 4.11.6 (Ext4)

3.2 실험 결과

실험 결과 readseq와 readrandom에서 모두 스레드 수가 10개일때까지 성능이 증가하지만 그 이후부터는 성능이 증가하지 않고 오히려 10개일때보다 낮은 성능이 나타났다.

성능이 증가하지 않는 이유로는 lock contention 문제가 있다.[4] 하나의 스레드가 데이터를 읽기 위해 Memtable에 접근할 때, Memtable의 reference counter를 atomic하게 증가시킨다. 그렇기 때문에 다수의 스레드가 Memtable에 접근하게 되면 reference counter에 lock contention이 발생하여 성능이 증가하지 못한다.

또한 Memtable에 데이터가 없는 경우에는 디스크에 저장되어 있는 SST file에 있는 데이터를 메모리에 올리고 read해야 한다. 만약 여러 스레드가 동시에 같은 데이터를 읽으려 한다면, 같은 물리 페이지에 접근하여야 한다. 이 경우에도 물리 페이지에 reference counter에 contention이 발생하여 성능이 저하된다.[6]

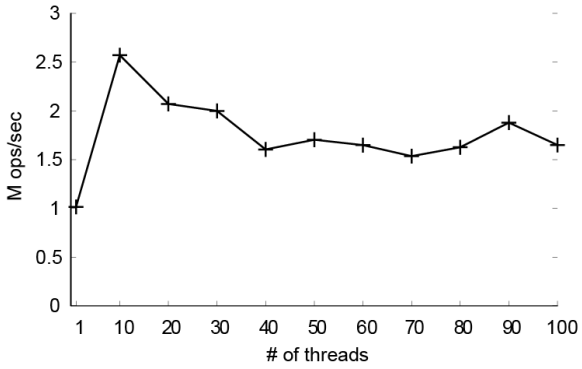

그림 2 readseq 실험결과

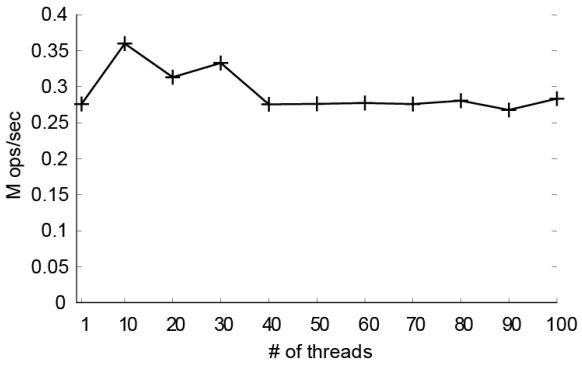

그림 3 readrandom 실험결과

4. 결론 및 향후 연구

본 논문에서는 실험을 통하여 RocksDB에서 스레드의 수가 증가하여도 데이터 읽기에 대한 성능이 증가하지 않는 것을 확인하였고, 그 문제에 대한 원인으로 Memtable에 접근 할 때 발생하는 lock contention 문제와 다수의 스레드가 같은 물리 페이지 상에 존재하는 데이터를 접근할 때 발생하는 lock contention 문제를 제기하였다.

향후에는 RocksDB에서 발생하는 lock contention 문제를 sloppy counter와 같이 하나의 lock을 분산시키는 연구와 lock-free 방식에 대한 연구를 진행하여 DB의 성능을 증가시키고자 한다.

5. 사 사

본 연구는 한국연구재단 기초연구실 지원사업(No. 2017R1A4A1015498), 정보통신기술진흥센터 지원사업 [R7117-16-0232, 32Gbps 데이터 서비스를 위한 익스트림 스토리지 입출력 기술 개발], 그리고 IITP 전문연구실 지원사업(2018-0-00549)의 지원을 받아 수행되었음.

참 고 문 헌

[1] RocksDB, https://rocksdb.org
[2] RocksDB https://github.com/facebook/rocksdb
[3] LevelDB, http://leveldb.org
[4] Silas Boyd-Wickizer, Austin T. Clements, Yandong Mao, Aleksey Pesterev, M. Frans Kaashoek, Robert Morris, and Nickolai Zeldovich. “An Analysis of Linux Scalability to Many Cores” In OSDI 2012: Proceedings of the 9th USENIX conference on Operating Systems Design and Implementation.
[5] C. Min, S. Kashyap, S. Maass, and T. Kim. “Understanding manycore scalability of file systems” In Proceedings of the 2016 USENIX Annual Technical Conference, Denver, CO, June 2016.