Code Review

RockDB 코드 리뷰

컴퓨터소프트웨어학부

김현정 박정호

CONTENTS

실험

실험 세팅

결과 및 분석

실험 세팅

기본적인 세팅은 지난 실험과 완전히 같다.

다만, Get의 각 subroutine 별 latency를 분석하기 위해 perf를 사용했으며, 이는 RocksDB에서 자체적으로 제공하는 perf context를 사용했다.

Get Latency를 크게 4개로 나누었으며, 그 항목은 다음과 같다.

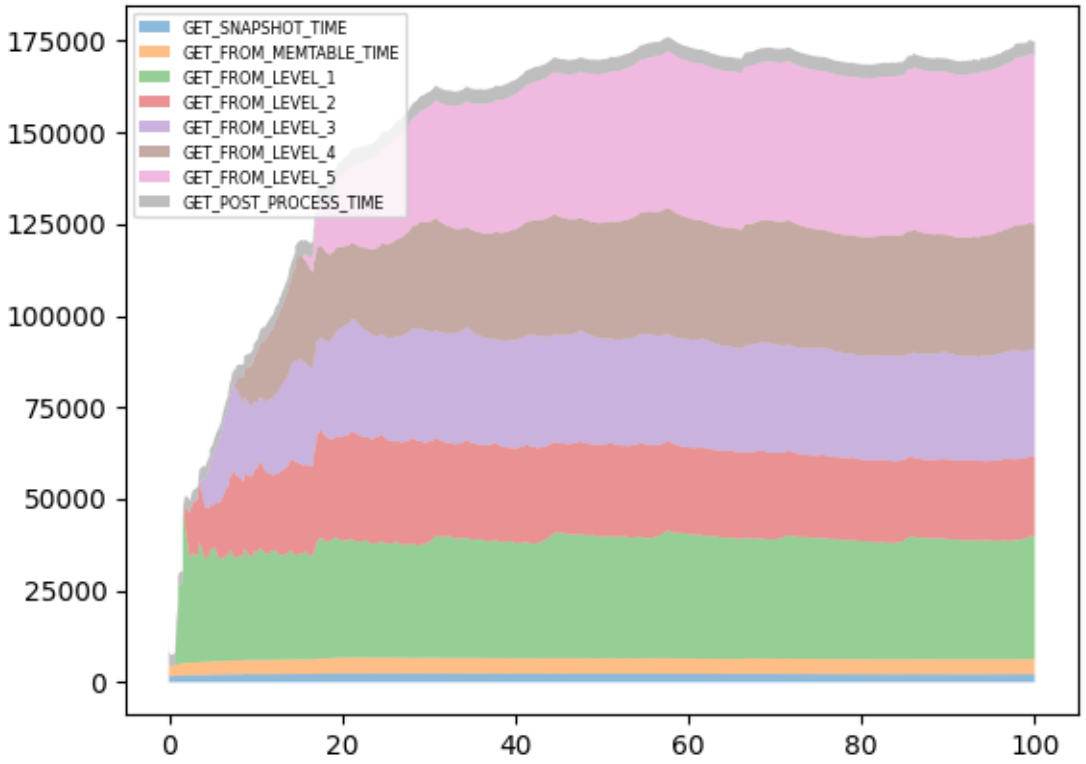
1. GET_SNAPSHOT_TIME : 주어진 snapshot 에서 sequence number 를 구하는 시간
2. GET_FROM_MEMTABLE_TIME : memtable 의 탐색 시간
3. GET_FROM_TABLE_NANOS : 각 레벨의 SST file 에서의 탐색 시간
4. GET_POST_PROCESS_TIME : 모든 탐색이 끝나고, statistics의 기록, status의 결정 등을 하는 시간

결과 및 분석

Snapshot 계산 시간, Memtable 탐색 시간, 탐색 후 처리 시간 등은 사실상 큰 의미를 갖지 않았다.

가장 큰 시간을 차지하는 것은 SST File에서의 탐색 시간이었는데, 시간이 지날 수록 탐색하는 레벨 수가 많아지고, 결국에는 모든 레벨을 다 탐색하게 되는 것을 볼 수 있었다.

각 레벨을 탐색하는 시간은 크게 차이가 나지 않았다.



Snapshot

Sequence Number

Usage

Sequence Number

기본적으로 set_snapshot = true인 상태에서 Transaction을 생성하면, 현재의 last_sequence를 sequence number로 가진 snapshot을 만들게 된다. 여기서 sequence number는 MVCC에서 timestamp처럼 기능하며, transaction의 visible한 범위를 정의하게 된다.

DB의 last_sequence는 처음에는 0으로 초기화, Transaction의 commit이 발생할 때마다 update된다.

Log에서 사용하는 Log Sequence Number도 이 Sequence Number를 사용하는 것으로 확인된다.

Memtable, Immutable Memtable, SST File 각각이 자신의 earliest sequence number와 latest sequence number를 관리하고 있다.

Usage of Snapshot at Get operation

Write Batch에서의 탐색에는 snapshot이 사용되지 않는다.

Write Batch 자체가 transaction의 local write buffer 같은 것이라 snapshot의 의미가 없기 때문이다.

snapshot은 Get에서 sequence number를 구하는데 사용된다.

참고로 isolation snapshot을 사용하지 않는 상황에서 snapshot은 nullptr가 전달된다. 따라서 sequence number는 DB의 last sequence를 가져오게 된다.

Sequence Number는 Timestamp, User Key와 묶여서 LookUpkey로 사용되고, 이 Sequence Number를 사용해서 데이터의 visibility를 확인한다.

Write Conflict Check

기본적으로 write가 발생한 key의 latest sequence number를 구한다.

(GetLatestSequenceForKey 함수에서 이를 구한다.)

이를 이용해서 write conflict를 commit 시에 체크하게 된다.

내부적으론 Memtable, immutable Memtable, SST files 순으로 Get을 수행하면서 진행되며, 아래의 두 조건 중 하나를 만족하면 종료한다.

1. 해당하는 Key가 존재할 경우
2. 해당 level의 earliest sequence number가 lower bound sequence number보다 작을 경우
   (즉 해당 level의 모든 update 는 해당 transaction 이전에 발생했을 경우)

이와 유사한 방법 (각 component의 earliest sequence number가 transaction 의 sequence number 보다 크다면 탐색을 생략) 을 사용하면 어느정도 Get의 탐색 범위를 줄일 수 있을 것 같다.

Get 분석

from Memtable

Version

필요한 추가 분석 사항

from Memtable

여기서 다루는 것은 GET_MEMTABLE_TIME에 해당하는 내용이다.

우선 bloom filter를 이용해서 실제로 table을 탐색할지 여부를 결정한다.

Memtable 은 기본적으로 Skiplist 로 구성되어 있다.
Skiplist 를 순회하며 sequence number 를 이용해 해당 value 가 snapshot에 속하는지 확인한다.

Version

우선 Version 이란, LSM Tree의 file list 를 말한다. Compaction 혹은 memtable flush가 발생할 때마다 새로운 Version 을 생성하게 된다. 또한 Version 은 SuperVersion에서 관리되는 reference count 를 이용해서 관리된다. Reference count 가 0이 된다면 해당 version 에 대한 삭제가 발생한다.

versionSet은 column family 정보를 포함한 모든 version 에 대한 정보를 관리하는 자료구조이다. 각 column family 에 해당하는 version들을 versionSet이 관리한다고 볼 수 있다. 또한 여기에서 current version 이 무엇인지까지도 저장한다.

기본적으로 Get 은 가장 최근의 Version 에서 동작한다. 매 탐색마다 Version 의 reference count 가 1씩 늘어나며, 탐색이 종료되면 reference count를 1 줄인다. 즉, Get 의 동작이 끝나지 않았다면, 탐색 중인 Version 이 더 이상 최신 Version이 아니더라도 탐색이 끝나기 전까지는 Version 을 유지한다.

필요한 추가 분석 사항

Get operation 과정 중 여러 함수에서 merge_context, merge_operand 등의 변수가 사용되는 것을 확인했다. 이것이 어떤 것을 의미하는지 확인할 필요가 있다. Get과 Merge 간의 관계를 파악할 필요가 있다. 현재까지는 Memtable에서의 탐색 중, merge를 수행하는 routine이 있다는 것을 확인했다.

SST Files 에 대한 탐색 중, FilePicker의 GetNextFile 함수에서 file 의 메타데이터 중 sequence number 를 사용하는 것을 확인했다. 다만 이것은 단순히 코드의 무결성을 점검하기 위한 assert 문이었고, 그마저도 level 0에서만 적용되는 것을 볼 수 있었는데, 이에 대한 분석이 필요하다. 크게 보자면, GetNextFile의 논리 전체를 파악할 필요가 있다.

Get 이 Current Version 위에서 동작한다는 것은 공식 사이트에서 확인할 수 있었으나, 이것이 snapshot isolation 을 사용하는 경우에도 그런 것인지는 확인하지 못했다. 따라서, 현재 실험 환경에서 Version을 어느 것을 가져오는지 파악할 필요가 있다.

THANK YOU!

감사합니다!