Source Code Analysis

RockDB 개선안

컴퓨터소프트웨어학부

김현정 박정호

CONTENTS

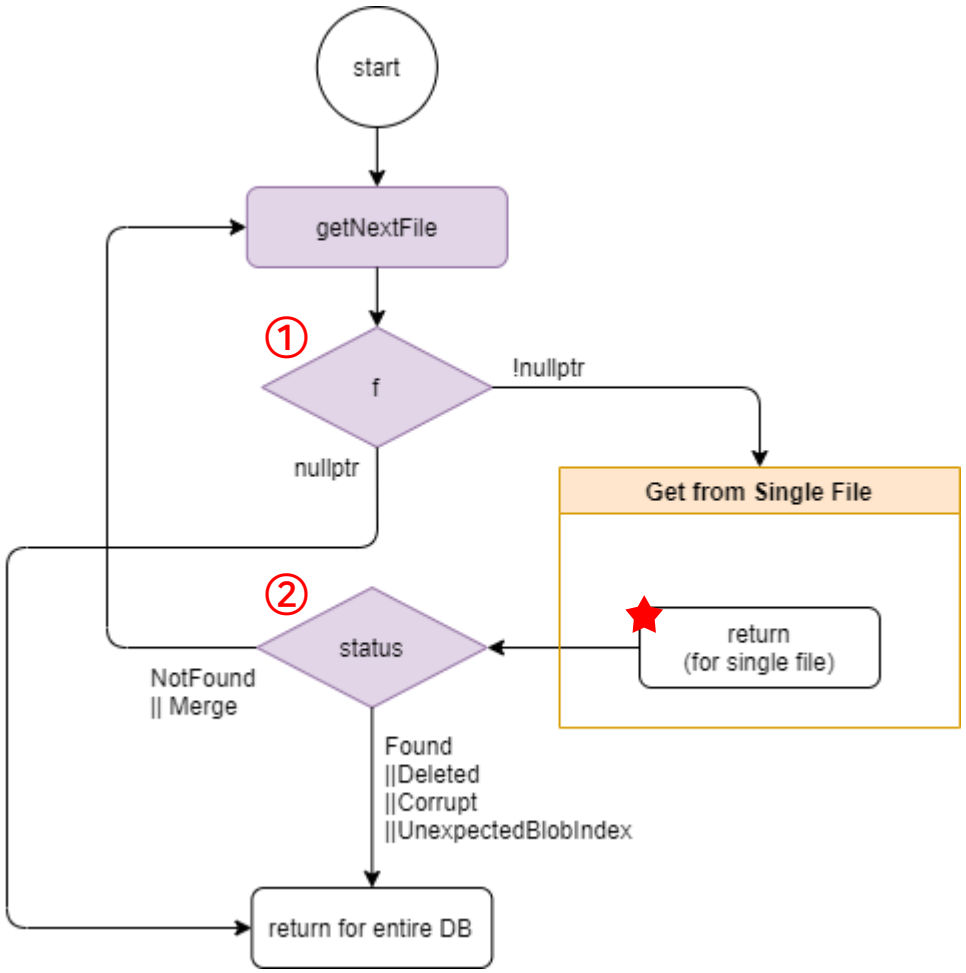
기존 아이디어

구체적 flow 분석

개요와 한계

개선안

구체적 flow 분석 (1)



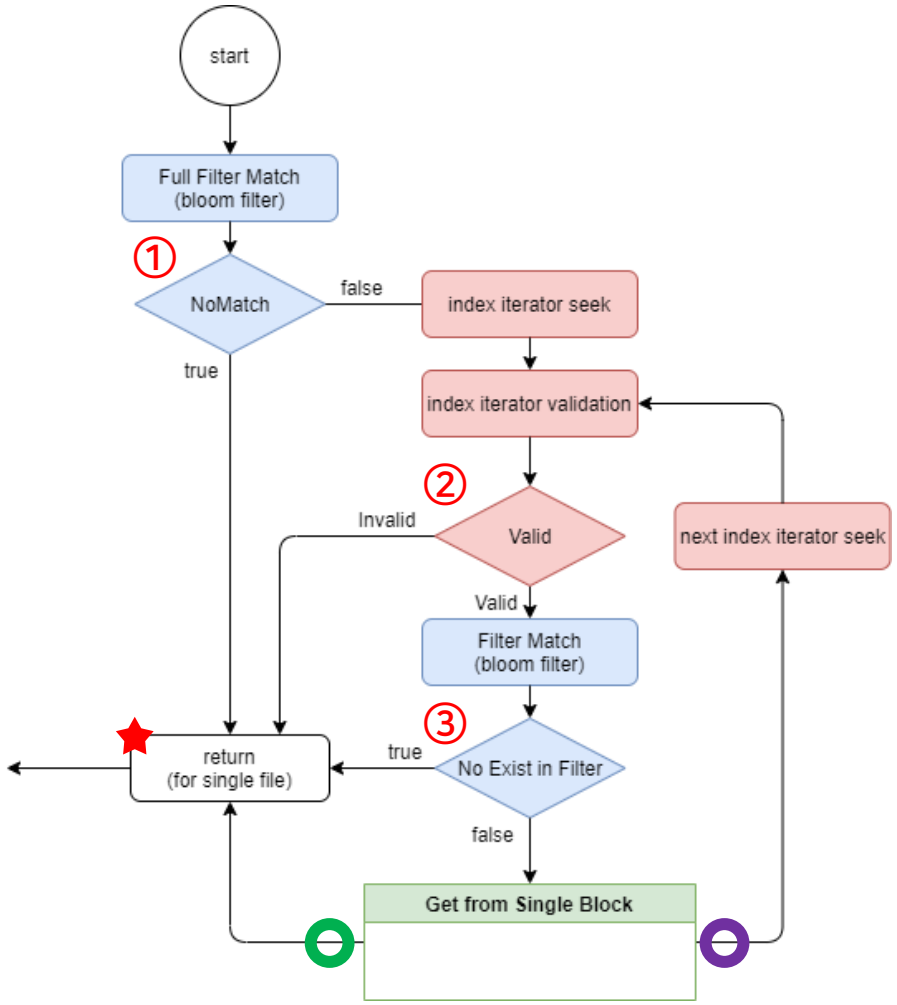
지난 flow chart에서 가장 바깥쪽에 위치한 Loop이다.

FilePicker 객체를 이용해서 파일을 레벨 순으로 찾는 과정인데, 여기서 FilePicker는 특별한 필터링을 수행하지는 않는다. 물론 각 파일의 key range 를 이용한 필터링은 수행되나, 단순히 min, max 값만을 사용하기에 bloom filter 정도의 효과는 없다.

① 은 단순히 전체 파일을 다 순회하는지 확인하는 부분이며, 전체 파일을 순회했을 경우 return 하게 된다.

② 는 파일 하나에 대한 탐색 (주황색 박스) 의 결과에 따른 분기이며, 해당 파일이 merge 중이었거나, 찾고자 하는 key-value 를 찾지 못했다면, 다음 루프로 가게 된다. 이외의 경우는 모두 return 한다.

구체적 flow 분석 (2)



앞의 루프에 포함되었던 주황색 박스에 해당하는 루프이다. 앞 슬라이드에서의 별표 표시된 return 블록을 여기서 확인할 수 있다.

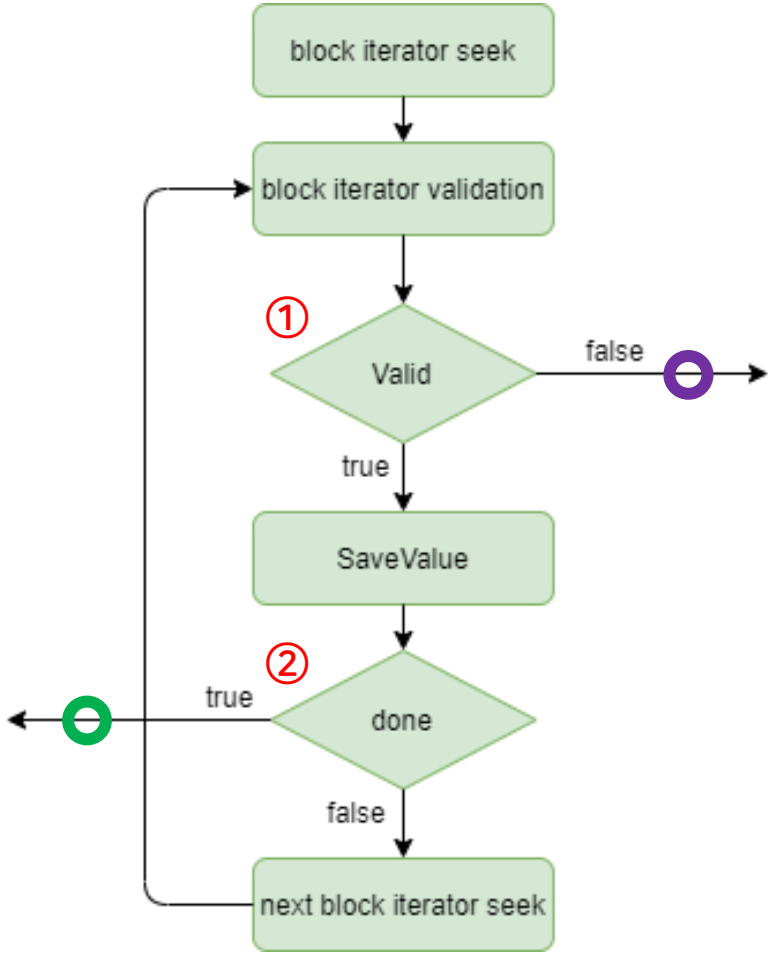
SST File 하나에 대해서 탐색을 수행하는 루프인데, 여기 표시된 과정은 그 중 index block을 순회하는 것만을 설명한다.

① 은 전체 파일에 대해서 bloom filter 를 사용하는 단계이며, 여기서 match되는 것이 없다면 바로 not found를 return 한다.

② 는 index iterator 를 찾았을 때 validation을 하는 단계이며, valid하지 않다면, (즉 index의 순회가 끝났다면) 루프를 break 한다.

③ 은 찾은 data block의 bloom filter 를 사용하는 단계이며, 여기서 match되는 것이 없다면 바로 not found를 return한다.

구체적 flow 분석 (3)



앞의 루프에 포함되었던 초록색 박스에 해당하는 루프이다. 앞 슬라이드에서의 초록색, 보라색 동그라미로 표시된 화살표를 여기서 확인할 수 있다.

Index 를 순회하면서 찾은 Data Block 하나에 대한 순회를 하는 함수이다. 여기서 key-value 와 search key 간의 comparison이 발생한다.

① 은 block iterator 를 찾았을 때 validation 을 하는 단계이며, valid하지 않다면, (즉 해당 block의 순회가 끝났다면) 루프를 break한다.

② 는 SaveValue 함수에서 현재 루프에서 보고 있는 key-value에 대한 key comparison 과 visibility check를 한 뒤의 결과에 따른 분기이다. True 라면 더 이상 탐색할 필요가 없다는 의미로, 루프를 break 한다.

개요와 한계

기존의 FileDescriptor가 가지고 있는 earliest_seqno, latest_seqno를 사용, Version::Get에서의 Loop에서 이를 체크해서 의미 없는 파일을 거르고자 했다. 현재 찾고자 하는 seqno보다 earliest_seqno가 크다면 해당 파일을 탐색하지 않고 넘어가도록 하는 것이 기본 아이디어였다.

하지만 이 아이디어는 FileDescriptor에서 관리하는 seqno의 range가 너무 클 경우 실효성이 없고, seqno만으로는 원하는 key가 있는지 판단할 수 없기 때문에, 다른 key를 가진 row의 seqno에 영향을 받는다.

정리하자면 다음과 같다.

- FileDescriptor의 seq range를 가지고 필터링하자
- BUT, Min, Max라는 값은 너무 정보가 부족하다.
- BUT, 찾고자 하는 key 이외의 row가 seq range에 영향을 준다.

개선안

다음 File 을 구하는 단계에서 Seq Range를 이용한 필터링을 수행할 경우, 찾고자 하는 key 이외의 row의 영향도 받게 된다. 이로 인해 seq range 필터링의 효용성이 매우 떨어지게 된다.

그렇다면 bloom filter check를 통해서 한번 필터링된 파일에 대해서 seq range 필터링을 사용하자. 한번 필터링된 파일들에 대해서 seq range 를 체크한다면 다른 key로 인한 영향을 받을 확률이 줄어들 것이다.

그럼에도 Min Max만으로는 상당히 정보가 부족하다는 사실에는 변함이 없다. 이 방식으로는 한 레벨 안에서의 탐색량을 줄이는 것은 어려울 것이다. 그렇지만 탐색하는 레벨의 수 자체는 줄일 수 있을 것이라 생각한다. LSM Tree의 특성 상, 레벨 K에 있는 파일들의 MIN(seq)보다 레벨 K+1에 있는 파일들의 MAX(seq)가 더 작을 것이라는 건 보장(같은 key를 가지는 파일만을 비교할 때)될 것이기 때문이다.

새로운 아이디어

개요

질문

개요

Radix tree 혹은 binary tree 구조를 사용해서 전체 파일의 sequence number에 대한 내용을 저장하는 아이디어를 생각했으나, 너무 공간이 클 것으로 예상이 되어 기각했다.

적당한 대표값 N개를 사용해서 false positive 가 있더라도 어느정도 필터링되는 구조를 짜도 될 것 같다는 생각을 했으나, 그 대표값을 어떻게 설계할지를 아직 고민중에 있다.

질문

1. Rocksdb내부에서 사용하는 ParsedInternal key가 user key와 함께 seq_num와 기타 정보를 담고 있는데 encode와 decode하는 과정을 거쳐야한다. 이게 seq를 쓰는데 걸림돌이 될까 안될까.

참고 사항

참고할 만한 정보

참고할 만한 사항

1. Cache & Block cache
    1. Cache는 uncompressed data
    2. Block cache는 compressed data가 들어간다.

THANK YOU!

감사합니다!