접근 빈도에 기반을 둔 SSD에서의 트리 인덱스 개선

최원기, 신민철, 박상현
연세대학교 컴퓨터과학과
e-mail : cwk1412@cs.yonsei.ac.kr

Improving tree index on SSD using access frequency

Won-Ki Choi, Min-Cheol Shin, Sang-Hyun Park
Dept of Computer Science, Yonsei University

요 약

플래시 메모리는 입출력 속도가 빠르고 에너지 효율성이 좋지만, out-place update만 가능하며 쓰기 연산을 위한 IO의 지연시간이 읽기 연산보다 현저히 길다는 단점을 가진다. 이를 보완하기 위해 고안된 FlashSSD(Solid State Drive)는 최근 하드 디스크를 대체하는 저장장치로 주목받고 있다. DBMS(Database Management System)의 성능 개선을 위하여 FlashSSD를 활용한 다양한 연구가 진행되었다. 그중 FD-tree[1]를 사용한 인덱스는 좋은 갱신 성능을 보임과 동시에 검색 성능을 보인다. 하지만 FD-tree의 구성 요소 중 하나인 레벨이 하나의 자료구조로만 이루어져 있어 인덱스로서의 비효율성을 가지고 있기 때문에 이를 인덱스의 접근 빈도를 이용하여 개선하고 검색 성능을 높이고자 한다.

1. 서론

플래시 메모리는 입출력 속도가 빠르고 에너지 효율성이 좋다는 장점을 가지고 있다. 또한, 크기가 작고 전원 공급이 중단되어도 데이터가 손실되지 않는 비휘발성인 특징을 가지고 있다. 하지만 플래시 메모리는 out-place update만 가능하고 비용이 큰 erase operation이 존재한다. 그리고 플래시 메모리에는 수명이 존재하여, erase 할 수 있는 횟수가 제한되어 있다. 또한, 읽기 속도에 비해 쓰기 속도가 느리다는 단점이 있다. FlashSSD는 이러한 플래시 메모리의 단점을 보완하기 위해 고안된 장치이다. FlashSSD는 하드 디스크를 대체하거나 메인 메모리와 하드 디스크의 사이에서 캐시로 사용하는 등 다양한 방법으로 활용된다. FlashSSD의 특징을 활용하여 DBMS(Database Management System) 성능을 극대화하기 위한 다양한 연구가 진행되고 있다.

최근 주목할 만한 연구로 FD-tree[1]와 PIO B-tree[2]가 있다. FD-tree는 쓰기 연산을 최적화하기 위해 기존 B+tree에서 자주 발생하던 random write를 줄이고 빠른 속도의 sequential write를 발생시키는 방향으로 기존 B+tree 구조를 개선하였다. PIO B-tree는 FlashSSD의 내부 병렬성을 활용하기 위해 기존 B+tree를 최적화한 인덱스 구조이다. 기존 IO 연산 단위를 바꾸어, Random IO를 모아서 FlashSSD에 동시에 보내는 방식이다. FlashSSD 내부에서 Random IO를 병렬적으로 처리할 수 있기 때문에 효율성을 극대화할 수 있다.

이 중, FD-tree는 head tree와 정렬된 run들로 구성된 트리 기반의 인덱스 구조이다. Head tree는 B+tree 구조를 사용하고 있으며, 각각의 run들은 head tree 아래로 단계를 이루어 구성되어 있다. 하위 레벨이 될수록 run의 크기는 일정한 비율로 증가한다.

FD-tree에서 각 레벨에 있는 run들의 크기가 제한되어 있기 때문에, 특정 레벨에 있는 run 안 인덱스의 개수가 일정한 값을 초과하게 되면, 해당 레벨에 있는 인덱스들이 하위 레벨로 merge 되는 연산이 수행된다. 이 과정에서 merge의 단위가 run 단위이므로, 인덱스 전체가 하위 레벨로 merge 되어 불필요하게 트리의 depth가 증가하는 경우가 생긴다. 이 때문에 검색의 성능이 저하될 수 있다.

본 논문에서는 위에 언급한 단점을 보완하기 위하여 인덱스 전체를 merge 하는 것이 아니라 자주 접근되지 않는 인덱스만을 merge 하는 방법을 제안한다. merge 할 인덱스를 결정하는 방법은 페이지 안의 인덱스들이 데이터에 접근하는 빈도수를 측정하여 상대적으로 낮은 빈도수를 가진 페이지 안의 인덱스들을 선택하는 방법이다.

위 방법 외에도 추가로, FD-tree에서 인덱스가 하위 레벨로만 merge가 되던 방법을 자주 접근되는 인덱스는 상위 레벨로도 merge가 가능하게 수정해 검색 성능을 높이는 방법도 제안한다. 이때 상위 레벨로 merge 할 일부 인덱스를 결정하는 방법은 첫 번째 방법과 반대로 상대적으로 높은 접근 빈도수를 가진 페이지 안의 인덱스들을 선택하는 방법이다.

이 논문의 구조는 다음과 같다. 2장에서는 FD-tree의 구조적 특성과 문제점을 논하고, 3장에서 인덱스의 접근 빈도를 반영하여 개선한 트리의 구조에 관해 설명한다. 제안