플래시 메모리 환경을 위한 이단계 인덱싱 방법

김종대[1], 장지웅[O2], 황규정[1], 김상욱[1]
[1]한양대학교 전자컴퓨터통신학부
{jdkim, lingohkc}@agape.hanyang.ac.kr, wook@hanyang.ac.kr
[2]한국산업기술대학교 게임공학과
jwchang@kpu.ac.kr

A Two-level Indexing Method in Flash Memory Environment

Jong-Dae Kim[1], Ji-Woong Chang[O2], Kyu-Jeong Hwang[1], Sang-Wook Kim[1]
[1]Division of Electrical and Computer Engineering, Hanyang University
[2]Department of Game and Multimedia Engineering, Korea Poly Technic University

1. 서 론

플래시 메모리는 가볍고 비휘발성이며 전력 소비가 적어서 모바일 기기 등에서 데이터 저장 장치로 널리 사용되고 있다[1]. 플래시 메모리의 사용 영역이 보다 확장되고 용량이 증가함에 따라서 플래시 메모리에 저장된 데이터에서 원하는 데이터를 빠르게 검색하기 위한 효율적인 인덱싱 방법의 필요성도 함께 증가하고 있다.

플래시 메모리는 기존의 저장 장치와는 다른 고유한 특성을 지닌다. 첫째, 플래시 메모리의 읽기와 쓰기 연산은 페이지 단위로 수행되며 읽기 연산에 비해 쓰기 연산의 속도가 매우 느리다[1]. 둘째, 덮어쓰기(in-place update) 연산이 지원되지 않는다[1]. 따라서 데이터의 갱신이 발생하면 기존의 데이터를 포함하는 페이지를 무효화하고, 갱신 사항이 반영된 새로운 데이터를 플래시 메모리의 다른 빈 페이지에 기록하는 방법을 사용한다[2]. 무효화된 페이지를 다시 사용하기 위해서는 소거(erase) 연산을 수행하여야 한다. 소거 연산은 다수의 연속된 페이지로 구성된 블록 단위로 이루어지며 먼저 해당 블록에 저장된 유효한 페이지들을 다른 블록으로 복사한 후 소거 연산을 수행한다[1]. 소거 연산은 오버헤드가 매우 큰 연산으로 잦은 쓰기 연산에 의해 발생한다. 이러한 이유로 쓰기 연산을 감소시키는 것은 플래시 메모리 환경에서 성능 향상의 중요한 요소가 된다.

이러한 플래시 메모리의 특성으로 인하여 현재 널리 사용되는 인덱싱 방법인 B-트리는 플래시 메모리 환경에서는 적합하지 않다. 디스크 기반 DBMS에서 주로 사용되는 트리 형태의 인덱스 구조는 트리 구조의 균형을 유지하기 위해서 추가적인 쓰기 연산이 발생하기 때문이다.

본 논문에서는 플래시 메모리 환경에서 효율적으로 동작하는 새로운 인덱싱 방법을 제안한다. 제안하는 인덱싱 방법에서는 플래시 메모리의 인덱스 페이지에 인덱스 엔트리들은 키 값에 의해 정렬된 순서로 저장하고, 메인 메모리에 파일 맵 형태로 존재하는 디렉토리를 이용하여 인덱스 페이지들에 대한 정렬된 순서로 관리한다. 제안하는 인덱싱 방법은 레코드의 삽입, 삭제에 의한 인덱스 구조의 동적인 변경을 최소화하여 인덱스 구조 변경으로 인한 플래시 메모리의 쓰기 연산을 최소화 하였다.

2. 연구 배경

디스크 기반의 DBMS에서 가장 대표적인 인덱스 구조로는 B-트리가 있다. B-트리는 디스크의 입출력이 용이하고 확장성이 뛰어나 디스크 기반 DBMS에서 현재 가장 널리 쓰이고 있다. 플래시 메모리에서 B-트리를 그대로 사용하는 경우, 노드 갱신이 발생하면 동일한 페이지에 덮어쓸 수 없기 때문에 새로운 페이지를 할당하여 해당 노드의 변경된 상태를 저장해야 한다. 이 때 분할 또는 병합 현상이 발생하면 이는 상위 노드에 영향을 미친다. 즉, 상위 노드에 대응되는 플래시 페이지에서도 새로운 페이지의 할당 및 복사 연산이 이루어져야 한다. B-트리에서는 이러한 분할과 병합이 리프 노드로부터 루트 노드에 이르기까지 연쇄적으로 발생할 수 있다. 이와 같이 데이터의 삽입과 삭제로 잦은 덮어쓰기를 유발시키는 B-트리는 인덱스에서 발생하는 다양한 연산을 수행함에 있어서 플래시 메모리의 물리적 특성으로 인하여 디스크 기반에서와 같은 성능을 기대하기 어렵다.

플래시 메모리를 물리적 기반으로 하는 기존의 인덱스 구조로는 B-트리를 플래시 메모리의 특성에 맞게 개선한 BFTL[2]이 있다. BFTL은 서로 다른 인덱스 노드에 속하는 엔트리들을 하나의 플래시 페이지에 가득 채워 저장함으로써 B-트리가 가지는 문제점을 개선하였다. 그러나 이 방법은 논리적인 인덱스 노드와 물리적인 인덱스 페이지 간에 불일치를 유발시키고, 이러한 불일치를 해결하기 위하여 별도의 자료구조를 사용한다. BFTL은 이 추가적인 자료구조로 인하여 메인 메모리의 사용량이 증가하고, 검색 성능이 감소하는 단점이 있다.

3. 이단계 인덱싱 방법

본 장에서는 플래시 메모리의 특성을 고려한 새로운 인덱싱 방법를 제안한다. 제안하는 인덱싱 방법은 균형 잡힌 트리 형태의 인덱스 구조를 지양하고 인덱스 구조의 동적인 변경을 최소화한다.

본 논문에서 제안하는 인덱스는 그림 1에 나타난 바와 같이 크게 인덱스 페이지와 디렉터리로 구성된다.

인덱스 페이지는 인덱스 엔트리들을 저장하는 단위이며 플래시 페이지에 대응되어 저장된다. 인덱스 페이지는 포함하는 인덱스 엔트리의 수에 따라 빈 공간이 존재할 수 있으며, 인덱스 엔트리들은 키 값의 크기에 따라 정렬되어 저장된다.

디렉터리는 인덱스 페이지를 논리적 순서에 따라 직접 접근이 가능하도록 하는 역할을 담당한다. 디렉터리는 파일 맵의 형태로 파일 시스템이 제공할 수 있다. 단, 이 경우 디렉터리는 효율적인 검색을 위하여 중간에 삽입, 삭제가 가능하여야 한다. 제안하는 인덱스에서 삽입 연산은 다음과 같은 절차를 통해 수행된다. 먼저, 삽입된 인덱스 엔

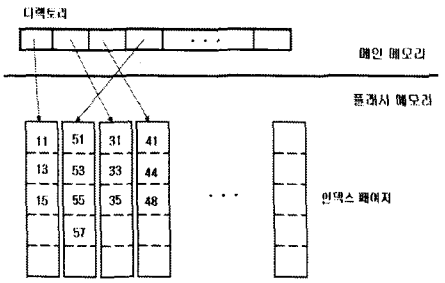

그림 1. 제안하는 인덱스 구조의 예.

트리의 키 값으로 디렉토리를 검색하여 저장될 인덱스 페이지를 찾는다. 만약, 검색된 인덱스 페이지에 충분한 공간이 존재하면 인덱스 엔트리를 저장하고 종료한다. 그러나 공간이 충분치 않을 경우 새로운 페이지를 할당받아 인덱스 엔트리들을 분할하여 저장하고 이를 디렉터리에 반영한다. 인덱스 엔트리의 삭제 연산은 삽입 연산과 동일한 방법으로 수행된다.

제안하는 인덱싱 방법은 다음과 같은 특징을 갖는다. 첫째, 인덱스 엔트리의 삽입, 삭제 연산에 따라 인덱스 구조가 동적으로 변경되지 않으므로 플래시 페이지의 쓰기 연산과 소거 연산의 횟수가 감소한다. 둘째, 인덱스 엔트리들을 효과적으로 클러스터링함으로써 빠른 검색을 지원한다. 셋째, 인덱스 페이지에 여분 공간을 허용함으로써 분할과 병합의 수행 횟수를 감소시킨다.

제안하는 인덱싱 방법에서 삽입과 삭제 연산은 발생 즉시 인덱스에 반영하는 것 보다 모아서 일괄 처리함으로써 보다 성능을 향상시킬 수 있다. 동일한 인덱스 페이지에 대하여 반복적인 삽입, 삭제 연산이 발생하는 경우 이 연산들의 최종 결과를 인덱스에 반영함으로써 쓰기 연산의 횟수를 감소시킬 수 있기 때문이다.

4. 성능 평가

본 장에서는 플래시 메모리 환경에서 B-트리 인덱스와 제안하는 인덱싱 방법의 검색, 삽입, 삭제 성능을 측정한다. 실험의 성능 척도는 플래시 메모리에서 발생한 읽기, 쓰기, 소거 연산의 총 횟수를 측정하고 각 연산의 수행 시간에 비례한 가중치를 곱한 값을 더한 전체 비용을 사용한다.

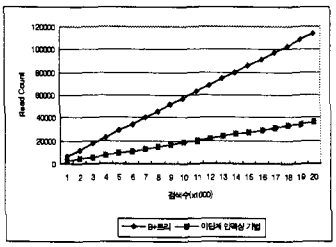

그림 2. 검색 연산 시 수행된 총비용.

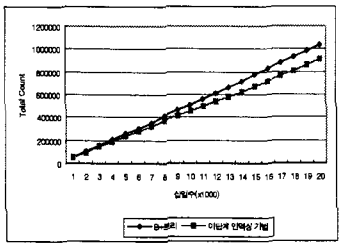

그림 3. 삽입 연산 시 수행된 총비용.

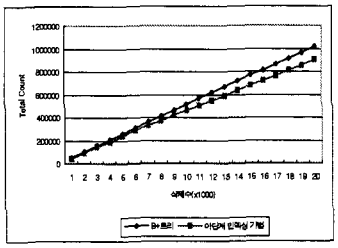

그림 4. 삭제 연산 시 수행된 총비용.

그림 2는 미리 구성된 인덱스를 이용하여 검색 연산의 횟수를 증가시켜가며 측정한 플래시 페이지의 읽기 연산의 횟수를 나타낸 것이다. 그림 2에서 나타난 바와 같이 검색 성능의 경우 제안하는 인덱싱 방법이 B-트리에 비해 약 3배 정도 우수하다. 이는 B-트리의 모든 노드가 플래시 페이지에 존재하는 반면, 제안하는 인덱싱 방법은 메인 메모리에 존재하는 디렉터리를 이용하여 검색을 수행하기 때문이다. 본 실험에서는 미리 구성된 B-트리의 높이가 3이기 때문에 이와 같은 성능의 차이가 나타난다.

그림 3과 4는 삽입/삭제 연산의 수를 증가시켜가며 측정한 전체 비용을 나타낸 것이다. 그림 3과 4에서 나타난 바와 같이 삽입/삭제 성능의 경우 제안하는 인덱싱 방법이 B-트리에 비해 약 14.5% 우수하다. 그 이유는 B-트리의 경우 리프 노드의 분할/병합에 의해서 상위 노드의 추가적인 분할/병합이 발생하는 반면, 제안하는 인덱싱 방법은 이러한 추가적인 비용이 발생하지 않기 때문이다. 삽입/삭제 성능은 B-트리의 높이가 높아질수록 제안하는 인덱싱 방법과의 성능 격차는 더 커진다.

5. 결 론

최근 들어, 플래시 메모리는 모바일 기기의 저장 장치뿐만 아니라 하드 디스크를 대체할 매체로 주목 받고 있다. 플래시 메모리의 용량이 증가함에 따라 플래시 메모리에 저장된 데이터를 빠르게 검색하기 위한 효율적으로 인덱싱 방법이 요구된다. 그러나 플래시 메모리의 물리적인 특성으로 인해 기존의 방법을 그대로 적용하면 심각한 성능 저하를 가져올 수 있다. 본 논문에서는 인덱스 구조의 동적인 변경을 최소화하여 플래시 메모리의 쓰기 연산의 횟수를 감소시키는 새로운 인덱싱 방법을 제안하였고 실험을 통하여 성능을 입증하였다.

감사의 글

본 논문은 제주대학교를 통한 정보통신부 및 정보통신진흥원의 대학 IT연구센터 지원사업(IITA-2005-C1090-0502-0009) 및 2007년도 정부(과학기술부)의 재원으로 한국과학재단(R01-2007-000-11773-0)의 연구비 지원을 받았습니다.

참고 문헌

[1] E. Gal and S. Toledo, "Algorithms and Data Structures for Flash Memories," *ACM Computing Survery*, Vol. 37, No. 2, pp. 138-163, 2005.

[2] C. Wu, L. Chang, and T. Kuo, "An Efficient B-Tree Layer for Flash-Memory Storage Systems," In *Proc. Int'l. Conf. on Real-Time and Embedded Computing Systems and Applications*, RTCSA, Vol. LNCS 2968, pp. 409-430, 2003.