제 26 회 한국정보처리학회 추계학술발표대회 논문집 제 13 권 제 2 호 (2006. 11)

플래시 메모리 상에서 B+트리 노드 크기 증가에 따른 성능 평가*

최해기*, 박동주*, 강원석**, 이동하**
*숭실대학교 컴퓨터학부
**대구경북과학기술연구원
e-mail : *{hgchoi, djpark}@computing.ssu.ac.kr
**{wskang, dhlee}@dgist.ac.kr

The Effect of the Node Size on the Performance of B+trees on Flash Memory

Haegi Choi*, Dong-Joo Park*, Won-Seok Kang**, Dong Ha Lee**
*Dept. of Computing, Soongsil University
**Dept. of IT, Daegu Gyeongbuk Institute of Science and Technology

요 약

플래시 메모리는 휴대폰과 PDA 와 같은 이동 기기에서 저장 장치로 널리 사용되고 있다. 또한 기가바이트(GB) 단위의 대용량화로 인해 노트북과 개인용 컴퓨터에서 보조기억장치로 사용되고 있다. 요즘에는 대용량의 데이터를 효율적으로 다루기 위한 B+트리와 같은 자료구조를 플래시 메모리 상에서 저비용으로 구현하려는 연구들이 이루어지고 있다. 지금까지의 연구에서는 플래시 메모리에서 B+트리를 구축할 때 노드 크기를 플래시 메모리의 섹터(sector) 크기로 사용해왔다. 본 논문에서는 노드 크기가 플래시 메모리의 섹터 크기보다 더 커졌을 경우, 플래시 메모리에서 구현되는 B+트리의 구축성능과 검색성능 그리고 저장 공간 사용량을 비교 분석한다. 키 삽입 시 정렬 알고리즘과 비정렬 알고리즘을 각각 사용해 구축비용을 측정하였으며 효율적인 노드 검색을 위해 인덱스 노드 헤드 구조를 사용한다. 그리고 이러한 실험결과는 B+트리 노드 크기를 섹터 크기보다 블록 크기로 할당할 때 B+트리 성능의 우수성을 보인다.

1. 서론

플래시 메모리는 소비전력이 적고 하드디스크에 비해 빠른 접근 속도를 가지며 물리적 충격에 강하다. 또한 크기가 작고 무게가 가벼우며 전원이 꺼지더라도 저장되어진 정보는 사라지지 않는 비휘발성의 성질 때문에 휴대 전화기, MP3 플레이어, 디지털 카메라와 같은 휴대용 정보기기들의 보조기억장치로 널리 사용되고 있다. 최근에는 플래시 메모리 집적도의 증가로 인해 대용량화 되어 감에 따라, 휴대용 정보기기들뿐만 아니라 개인용 컴퓨터나 노트북에서도 보조기억장치로 활용되고 있다. 따라서 대용량의 데이터를 효율적으로 접근하기 위한 B+트리와 같은 자료구조를 플래시 메모리상에서 저비용으로 구현하려는 연구들이 이루어지고 있다[1][2].

그런데 플래시 메모리는 하드 디스크와 달리, 섹터에 대해 덮어쓰기(overwrite)가 허용되지 않는다. 그리고 섹터의 데이터를 지우기 위해서는 섹터를 포함하고 있는 블록 전체를 소거(erase)해야 한다.

플래시 메모리에서의 입출력 단위는 섹터(512Byte)이기 때문에, 노드 크기를 섹터 단위로 할당하여 B+트리를 구축하는 연구가 진행되어 왔다[1][2]. 그러나 아직까지 여러 개의 섹터를 하나의 노드에 할당하여 B+트리를 구현할 때 플래시 메모리에서 일어나는 블록 소거횟수나 쓰기연산, 읽기연산의 비용 변화에

* 본 논문은 정통부 및 정보통신연구진흥원의 정보통신선도기반기술개발사업의 연구결과로 수행되었습니다.

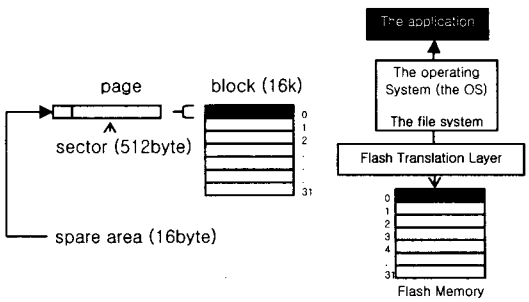
대한 연구가 이루어지지 않았다. 노드에 할당된 섹터의 개수가 B+트리 성능에 미치는 영향을 측정함으로써, 플래시 메모리 상에서 B+트리 구현 시 보다 적합한 노드 크기를 측정하고 플래시 메모리에서 B+트리를 저비용으로 구현할 수 있는 연구가 필요하다. 따라서 본 논문에서는 노드에 할당되는 섹터의 개수를 1 개(512Byte)에서부터 32 개(16KB)까지 확장시키면서 노드 크기가 B+트리 성능에 미치는 영향을 비교분석한다.

본 논문의 구성은 다음과 같다. 2 장에서는 플래시 메모리의 특성과 FTL 그리고 플래시 메모리 상에서 B+트리를 구현한 연구를 간단히 설명한다. 3 장에서는 본 연구에서 구현한 노드 인덱스 헤드 구조를 설명하고 4 장에서는 실험 결과를 비교분석한다. 마지막으로 5 장에서 결론을 맺는다.

2. 관련연구

2.1 플래시 메모리의 특성과 FTL

데이터 저장 용도로 많이 사용되는 NAND 형 플래시 메모리는 다수 개의 블록(block)으로 구성된다. (그림 1) - (a)와 같이 하나의 블록은 32 개의 페이지로 이루어지며, 하나의 페이지는 512 바이트의 데이터를 저장하는 영역(sector area)과 16 바이트의 부가 정보(ECC, LSN)를 저장하는 영역(spare area)으로 구성된다. 플래시 메모리는 데이터 입출력 단위와 데이터 소거 단위가 각각 다르다. 플래시 메모리의 데이터 입출력 단위는 페이지(page)이며 데이터 소거 단위는 블록이다.



(a) 플래시 메모리 구성 (b)FTL

(그림 1) 플래시 메모리 구성과 FTL

플래시 메모리와 파일 시스템 사이에는 FTL(Flash Translation Layer)이 존재한다. FTL 은 플래시 메모리가 처리할 수 없는 덮어쓰기를 효율적으로 처리하기 위한 기법인데, 현재 연구되어진 FTL 중 가장 성능이 우수한 기법은 FAST(Fully Associative Sector Translation)이다[3].

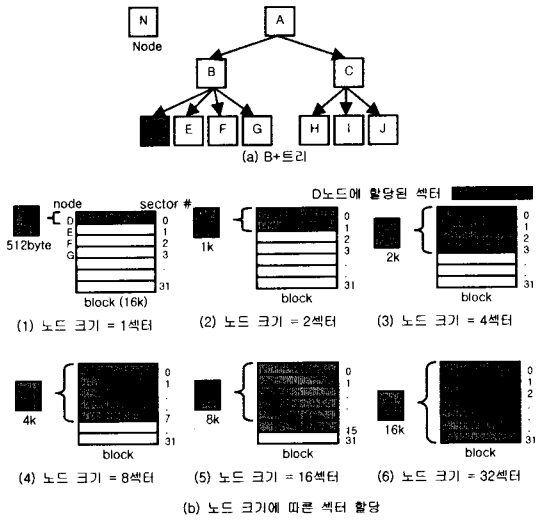
2.2 플래시 메모리상에서의 B+트리 연구

BOF 와 BFTL 은 플래시 메모리의 물리적 특성을 고려하여 B+트리를 저비용으로 구축하기 위한 기법이다[1][2]. 이 기법들은 플래시 메모리에서 비용소모가 가장 큰 블록 소거를 줄이기 위해서 유보 버퍼(Reservation buffer)를 사용하고 있다. BFTL 은 노드 관리를 위해 노드 변환 테이블(Node Translation Table)을 유지한다.

3. B+트리 노드 확장 설계 및 구현

3.1 B+트리 노드 확장 설계

본 논문은 B+트리의 노드크기를 섹터 1 개에서부터 32 개(블록크기)까지 증가시켜 B+트리에 미치는 성능을 비교분석한다. (그림 2)와 같이 한 노드를 2 개 이상의 섹터에 할당하는 경우 물리적으로 연속되는 섹터를 하나의 노드로 사용한다. (그림 2)에서는 D 노드를 여러 개의 섹터로 할당 하였을 때 D 노드에 할당된 섹터를 나타낸다.



(그림 2) 노드 확장 방법

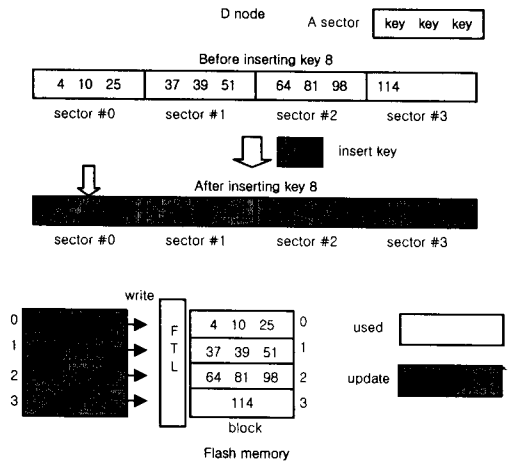
3.2 삽입 알고리즘

본 논문에서는 키 삽입 시 정렬, 비정렬 알고리즘을 각각 사용하여 노드 크기 증가 시 B+트리 성능을 비교분석한다. 정렬 알고리즘은 일반적인 B+트리에서 사용하는 키 삽입 방식이다. 이 알고리즘은 노드안의 키 값들을 항상 정렬된 상태로 유지한다. 이와 달리 비정렬 알고리즘은 노드안의 키 값들을 정렬시키지 않고 키 값을 삽입하는 방식이다.

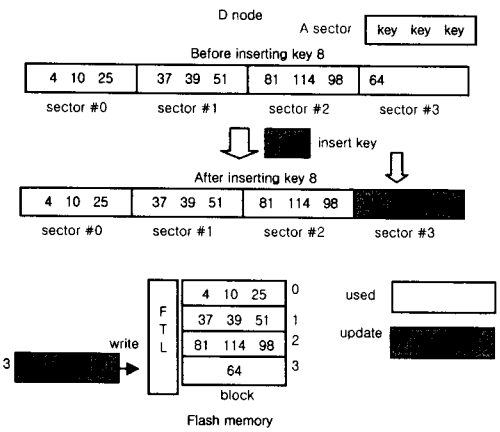
정렬 알고리즘은 B+트리 구축 시 노드가 할당하는 섹터의 수가 증가할수록 추가적인 쓰기 연산을 빈번히 발생시키는 문제점을 가지고 있다. 예를 들어 (그림 3)에서와 같이 노드에 할당된 섹터 개수가 4 개이고 따라서 D 노드는 플래시 메모리의 0 번~3 번 섹터를 사용한다. 0 번 섹터에는 D 노드의 키 값 4, 10, 25 가 저장되어 있고 섹터 1 번에는 키 값 37, 39, 51 이 저장되어 있다. 이렇게 노드안의 키 값은 정렬된

순서로 섹터에 저장된다. 이 때 D 노드에 키 값 '8' 이 삽입되어지면 노드 안의 키 값들을 정렬시키기 위해서 키가 삽입된 노드가 할당하고 있는 4 개의 섹터 정보들을 모두 수정해야한다. 즉 1 개의 키 값 삽입에 대해서 최악의 경우 4 개의 섹터에 대하여 쓰기연산이 발생하므로 노드가 1 개의 섹터를 할당할 때보다 쓰기연산 횟수가 많아진다. 이렇게 쓰기연산이 많아지면 B+트리에 키 삽입 시 플래시 메모리의 쓰기연산 발생을 증가시켜 구축 성능에 악영향을 미친다.

비정렬 알고리즘은 정렬 알고리즘에서 발생하는 비효율적인 쓰기연산을 대부분 방지한다. (그림 4)는 정렬 알고리즘에서 발생되는 추가적인 쓰기연산을 발생시키지 않고 키 값이 삽입되는 비정렬 알고리즘을 보여준다. 노드가 처음 생성될 때 가지고 있는 키 값들은 정렬된 상태로 저장하기 때문에 비정렬 알고리즘에서라도 노드가 저장할 수 있는 최대 키 값의 반 정도는 항상 정렬된 상태를 유지한다.



(그림 3) 정렬 알고리즘 (노드크기 = 섹터 4 개)



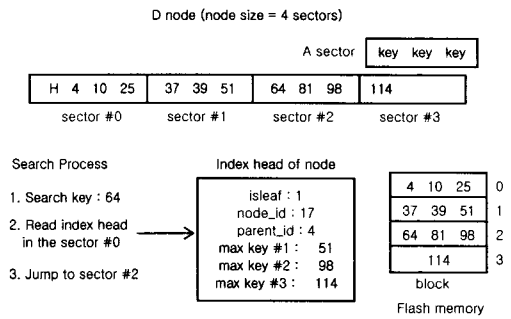
(그림 4) 비정렬 알고리즘

3.3 검색 알고리즘

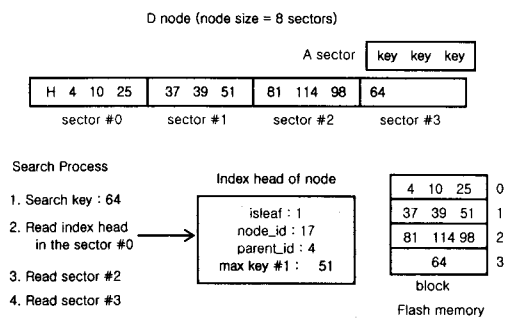
노드에 할당된 섹터의 개수가 증가하면 한 노드 안에 있는 키 값을 검색할 때 그 노드가 할당하고 있는 여러 개의 섹터를 읽어야하므로 검색 성능을 저하시킬 수 있다. 이와 같은 단점을 해결하기 위하여 삽입 알고리즘(정렬, 비정렬)에 따라 노드 헤드를 전 인덱스(full index), 반 인덱스(half index) 구조로 설계한다. 그리고 키 검색 시 노드 헤드 구조에 따라 전 인덱스, 반 인덱스 검색 알고리즘을 사용한다.

전 인덱스 구조 노드 헤드에서는 각 섹터에 저장되어있는 최대 키 값(maximum key value) 정보를 관리한다. (그림 5)는 이러한 전 인덱스 헤드 구조를 사용하여 키 값 65 를 검색할 때에 요구되어지는 읽기 연산 순서를 나타낸다. 이러한 전 인덱스 검색 방법은 노드가 할당하는 섹터의 개수가 32 개가 되더라도 최대 2 번의 읽기 연산만으로 노드 안의 원하는 키 값이 들어있는 섹터를 읽을 수 있다.

반 인덱스 구조 노드 헤드는 정렬된 부분에 대해서는 최대 키 값 정보를 유지하여 인덱스 검색방법을 사용하고, 비정렬된 부분의 키 값은 순차적 검색을 적용한다. (그림 6)은 반 인덱스 헤드 구조를 이용해 노드 안의 키 값 65 를 검색하는 순서를 보여준다. 이와 같은 반 인덱스 검색은 이분 검색 보다 더 향상된 검색성능을 보여준다.



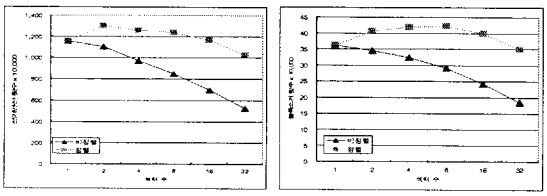
(그림 5) 전 인덱스 검색방법



(그림 6) 반 인덱스 검색 방법

4. 성능 측정 및 비교 평가

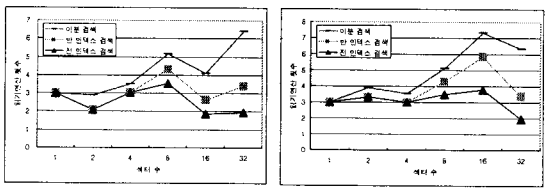
본 논문의 실험은 Red Hat Linux release 9 (Shrike) 운영체제에서 ANSI C 로 구현하였으며 FAST 기법[3]의 FTL 을 사용하였다. 본 장에서는 노드에 할당된 섹터 개수를 증가시킬 때 플래시 메모리에서 발생하는 비용을 측정한다.



쓰기연산 블록소거연산

(그림 7) B+트리 구축 성능

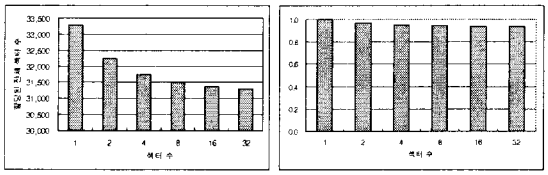
(그림 7)은 B+트리에 500,000 개의 임의(random) 키 삽입 시 노드 크기에 따라 플래시 메모리에서 발생하는 쓰기연산과 블록소거연산 횟수를 나타낸다. 노드 크기가 커질수록 FTL 알고리즘에 더 효율적인 섹터 쓰기연산을 발생시켜 플래시 메모리에서 발생되는 쓰기연산과 블록소거연산을 감소시킨다. 그러나 정렬 알고리즘에서는 키 값을 정렬하기 위하여 부가적인 쓰기연산이 발생하기 때문에 노드 크기가 커질수록 쓰기연산이 증가한다. 그러나 FTL 로 인해 이러한 쓰기연산은 보완되어져 다시 감소하게 된다. 비정렬 알고리즘에서는 노드 크기가 커질수록 쓰기연산과 블록소거연산이 크게 감소한다.



키 개수 = 500,000 키 개수 = 1,000,000

(그림 8) B+트리 검색 성능

(그림 8)은 키 500,000 개와 1,000,000 개가 삽입된 B+트리에 하나의 키를 검색하기 위해서 평균적으로 발생되는 플래시 메모리 읽기 연산 횟수를 비교한다. 노드의 크기가 증가할 때 트리의 레벨이 낮아질 때에는 검색 성능이 향상되지만 레벨이 낮아지지 않을 때는 성능이 감소한다. 이분 검색에서는 트리의 레벨이 낮아질 때 검색성능이 약간 향상되어 노드의 크기가 커질수록 검색성능이 전반적으로 취약해진다. 그러나 반 인덱스, 전 인덱스 검색에서는 트리의 레벨이 낮아질 때 검색 성능이 크게 향상되어 노드크기가 증가할 때 취약해지는 검색 성능을 보완해준다.



(그림 9) 플래시 메모리 사용량

(그림 9)는 노드가 할당하는 섹터의 개수가 증가할 때 플래시 메모리 사용량을 나타낸다. 왼쪽 그래프는 1,000,000 개의 키 값을 B+트리에 삽입했을 때 플래시 메모리에서 사용하는 전체 섹터 개수를 나타낸다. 그리고 오른쪽 그래프는 왼쪽 그래프를 정규화 시킨 것이다. B+트리에서 하나의 노드는 일정한 개수 이상의 키를 가지고 있어야 하기 때문에 노드에서 발생되는 공간낭비가 제한되어진다. 따라서 노드의 크기가 커지더라도 플래시 메모리의 사용량에는 큰 차이가 없는 것을 알 수 있다.

5. 결론

본 논문에서는 플래시 메모리에 구현되는 B+트리에서 노드 크기가 B+트리 성능에 미치는 영향을 비교분석하였다. 이러한 실험은 플래시 메모리에 노드 크기를 섹터 크기로 사용하기 보다는 블록 크기로 사용하는 것이 더 효율적임을 보여주고 있다.

향후연구로는 최근의 플래시 메모리는 대용량화 되면서 페이지의 크기가 2K 바이트로 확장되는 추세이다. 이러한 플래시 메모리에서의 적합한 B+트리 노드 크기에 대한 연구도 필요하다.

참고문헌

[1] Jung Hyun Nam, Dong-Joo Park, "Design and Implementation of the B-Tree on Flash Memory," submitted to Journal of KIPS, 2005.

[2] Chin-Hsien Wu, Li-Pin Chang, Tei-Wei Kuo, "An Efficient B-Tree Layer for Flash-Memory Storage Systems," The 9th International Conference on Real-Time Computing Systems and Applications (RTCSA), 2003.

[3] Dong-Joo Park, Won-Kyung Choi, Sang-Won Lee, "FAST: A Log Buffer based FTL Scheme with Fully Associative Sector Translation," Journal of KIPS, Vol. 12-A, No. 3, June 2005.

[4] J.Kim, J.M. Kim, S.H. Noh, S.L. Min, and Y. Cho, "A Space-Efficient Flash Translation Layer for Compact Flash System," IEEE Transactions on Consumer Electronics, Vol.48, No.2, pp.366-375, 2002.