NAND 플래시 메모리에서 페이지 크기에 따른 B+ 트리의 성능 평가[1)]

유현석° 전한별 김도윤 박상원
한국외국어대학교 컴퓨터및정보통신공학과
{hsyoo°, hbchun, dykim, swpark}@dislab.hufs.ac.kr

Performance Evaluation of a B+Tree on Various Page Sizes in NAND Flash Memory

Hyun-Seok Yoo°, Han Byul Chun, Do Yun Kim, Sangwon Park
Computer Science & Information Communication Engineering, Hankuk University of Foreign Studies

요 약

휴대용 기기들의 데이터 저장소로 플래시 메모리가 많이 사용되고 있으며 플래시 메모리가 대용량화 되어감에 따라 점차 디스크를 대체할 것이라 예상된다. 따라서 데이터베이스 시스템 역시 저장 매체로 플래시 메모리의 사용이 증가할 것으로 예상되며 이에 따른 효율적인 인덱스가 필요하다. 플래시 메모리 기반의 효율적인 인덱스 구축을 위하여 B+ 트리의 페이지 크기에 따른 성능 평가가 필요하다. 본 논문에서는 B+ 트리와 버퍼 관리자를 구현하고, 플래시 변환 계층의 대표적인 4 가지 알고리즘에 대해 B+ 트리의 페이지 크기에 따른 성능을 비교, 분석하여 플래시 메모리 기반의 인덱스를 구축하기 위한 방향을 제시한다.

1. 서 론

디지털 카메라, MP3 플레이어, 핸드폰 등과 같은 휴대용 기기들의 데이터 저장소로 플래시 메모리를 많이 사용되고 있다. 이것은 플래시 메모리가 전원이 꺼지더라도 저장된 정보가 사라지지 않는 비휘발성의 특성을 가지며, 소비전력이 적고 기존 디스크 저장 장치에 비해 빠른 데이터 접근 성능을 보이고, 크기가 매우 작은 특성이 있기 때문이다. 이러한 플래시 메모리는 사용 분야가 다양해지고 저장 용량도 대용량화 되어가는 추세이다. 플래시 메모리의 용량이 증가함에 따라 많은 분야에서 기존 디스크 저장 장치가 플래시 메모리로 대체될 것으로 예상된다. 특히 데이터베이스 시스템도 저장매체로 플래시 메모리의 사용이 증가할 것이다.

하지만 NAND 플래시는 섹터 단위로 읽기, 쓰기 연산이 수행된다. 이 때 블록에 대한 쓰기 연산은 그 블록을 소거한 다음에 수행되어야 하는 제약이 있다. 또한 플래시 메모리에 이루어지는 연산들(읽기, 쓰기, 소거)의 수행시간은 큰 차이가 있으며 각 연산의 수행시간 비율은 1:8:65이다. 즉, 소거 연산은 블록 단위로 수행되며 읽기, 쓰기 단위인 섹터 단위 보다 수행 시간이 오래 걸린다. 그리고 각 블록은 최대 10만 번까지 소거할 수 있다. 따라서 플래시 메모리를 효율적으로 사용하기 위해 플래시 변환 계층(FTL, Flash-Memory Translation Layer)을 사용한다. 플래시 변환 계층은 플래시 메모리와 파일 시스템 사이에 위치하면서 플래시 메모리를 디스크처럼 블록 디바이스(block device)로 사용하게 하는 계층이다. 이것은 파일 시스템에서 요구하는 블록을 플래시 메모리의 블록으로 매핑(mapping)하는 역할을 한다.

1) 본 논문은 2006년도 한국외국어대학교 학술연구비 지원에의 해서 연구되었음

NAND 플래시 메모리를 저장 장치로 사용하는 데이터베이스 시스템에서 효율적인 인덱스 지원을 위하여 B+ 트리의 페이지 크기에 대한 플래시 메모리에서의 성능 평가가 필요하다. 그 이유는 플래시 메모리의 읽기, 쓰기 단위와 소거 단위가 다르고 그 속도도 상이하기 때문이다. 본 논문에서는 플래시 변환 계층 위에 데이터베이스를 효율적으로 접근하기 위해서 B+ 트리와 버퍼 관리자를 구현하고 각 플래시 변환 계층에 대한 알고리즘에 따른 성능 평가를 하였다. 즉, B+ 트리의 페이지 크기에 따른 플래시 변환 계층 알고리즘의 성능을 비교, 분석하여 플래시 메모리 기반의 인덱스를 구축하기 위한 방향을 제시한다.

2. 관련 연구

플래시 변환 계층 알고리즘에서 사용되는 매핑 방법은 3가지가 있다. 첫 번째는 논리 섹터와 물리 섹터를 매핑하는 섹터(sector) 매핑이 있다. 두 번째는 논리 블록 내부에 있는 섹터의 위치가 물리 블록 내부에 있는 섹터의 위치가 동일하게 블록을 매핑하는 블록(block) 매핑이 있다. 마지막으로 물리 블록을 얻기 위해 블록 매핑을 이용하고, 물리 블록 내에 있는 섹터를 찾기 위해 섹터 매핑을 이용하는 하이브리드(hybrid) 매핑이 있다. 또한, 플래시 변환 계층 알고리즘은 쓰는 방식에 따라 물리적인 위치에 논리적인 섹터와 같은 섹터에 쓰는 in-place 방식과 물리적인 섹터 순서대로 쓰는 out-of-place 방식이 있다. 이처럼 플래시 변환 계층 알고리즘은 매핑 방법이나 물리적인 위치에 쓰는 방식에 따라 다음과 같은 방법들로 분류할 수 있다.

2.1 복사 블록 기법(FMAX)

M-System에서 사용되는 FMAX[1]는 그림 1과 같이 플

래시 메모리의 블록을 데이터 블록과 복사 블록으로 나눈다. 데이터 블록에 있는 섹터 A에 쓰기가 발생하면 in-place로 데이터를 기록한다. 섹터 A에 덮어쓰기가 발생하면 복사 블록(mirroring block)에 있는 섹터 B에 기록한다. 그리고 복사 블록에 in-place로 기록하는 단순 블록 상태(simple block state)에서 다시 덮어쓰기가 발생하면 out-of-place로 섹터 C에 기록하고 복합 블록 상태(compound block state)로 바뀐다. 이때 복사 블록에 빈 섹터가 없으면 복합 블록 상태일 경우 합병(merge) 연산을 실시하고, 단순 블록 상태에서는 교환(switch) 연산을 수행하여 소거 횟수를 줄인다.

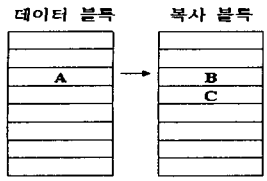

그림 1. 복사 블록 기법

2.2 로그 블록 기법(BAST)

로그 블록 기법(Log block Scheme)[2]은 그림 2와 같이 플래시 메모리의 블록을 데이터 블록과 로그 블록(log block)으로 나눈다. 로그블록은 블록 매핑과 섹터 매핑을 혼합한 하이브리드 매핑을 실시한다. 데이터 블록에 있는 섹터 A에 쓰기가 발생하면 데이터 블록에 in-place로 기록하고, 섹터 A에 덮어 쓰기가 발생하면 로그 블록에 있는 섹터 B에 out-of-place로 기록한다. 다시 섹터 A에 덮어쓰기가 발생하면 섹터 C에 기록한다. 로그 블록에 빈 섹터가 없고 로그' 블록이 순차적이면 교환 연산을 수행 하고, 순차적이지 않으면 합병 연산을 수행 한다. 또한 빈 로그 블록이 없으면 로그 블록 중 희생 블록(victim block)을 선정하여 합병 연산을 한다.

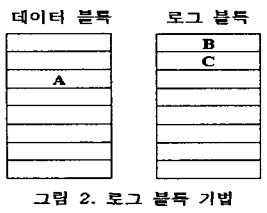

그림 2. 로그 블록 기법

2.3 개선된 로그 블록 기법(FAST)

로그 블록 기법을 개선한 FAST(Full Associative Sector Translation)[3]는 그림 3과 같이 논리적으로 데이터 블록과 임의 쓰기용 로그 블록 그리고 순차 쓰기용 로그 블록으로 나눈다. 로그 블록 기법과 마찬가지로 하이브리드 매핑을 사용하는데 데이터 블록에 있는 섹터 B에 쓰기가 발생하면 데이터 블록에 in-place로 기록한다. 섹터 B에 덮어쓰기가 발생하면 임의 쓰기 로그 블록에 있는 섹터 C에 out-of-place로 기록한다. 만약 데이터 블록 0번째 섹터 A에 대한 덮어쓰기가 발생할 경우 순차 쓰기 로그 블록 E에 in-place로 기록한다. 다시 섹터 A에 덮어 쓰기가 발생할 경우 순차 쓰기용 로그 블록에 있는 섹터 E에 다시 덮어쓰기가 발생하여 합병연산을 하게 되고, 빈 섹터가 없을 경우 교환 연산으로 소거 횟수를 줄인다. 임의 쓰기 로그 블록은 여러 개를 두며 임의 쓰기 로그 블록에 빈 섹터가 없으면 FIFO(First In First Out) 방식으로 합병 연산을 실시한다.

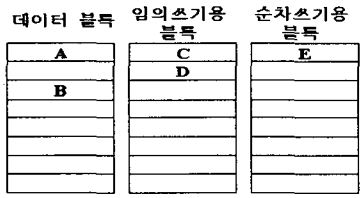

그림 3. 개선된 로그 블록 기법

2.4 여유 영역 기법(MTBS)

여유 영역 기법[4]은 그림 4와 같이 블록 매핑을 사용하고 각 블록은 데이터 영역(data area)과 여유 영역(space area)으로 나눈다. 데이터 영역에 있는 섹터 A에 in-place로 데이터를 기록하다가 섹터 A에 덮어쓰기가 발생하면 여유 영역에 있는 섹터 B에 out-of-place로 기록한다. 그리고 섹터 A에 다시 덮어쓰기가 발생하면 여유 영역에 있는 섹터 C에 out-of-place로 기록한다. 블록의 여유 영역에 빈 섹터가 없으면 합병 연산을 실시한다.

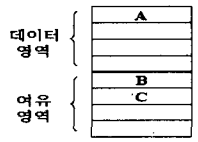

그림 4. 여유 영역 기법

3. 실험환경

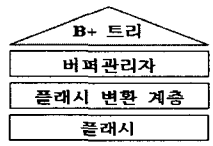

그림 5. 실험환경

플래시 메모리에 대한 I/O를 줄이기 위하여 그림 5에서 보는 바와 같이 플래시 변환 계층과 B+ 트리 사이에 버퍼 관리자를 두었다. 실험에 사용한 B+ 인덱스는 버퍼 관리자에 I/O를 요청한다. 본 실험에서는 임의의 정수 100,000개를 B+ 트리에 삽입하였다. 버퍼 관리자는 LRU로 동작하며 플래시 변환 계층은 2장에서 설명한 4가지 방법을 사용하였다. 섹터의 크기는 512바이트이며 블록의 크기는 16K이다. 버퍼 관리자 및 B+ 트리에서 사용한 페이지의 크기는 각각 섹터 단위인 512B, 입출력 단위인 4K, 그리고 블록 단위인 16K로 하였다.

4. 실험결과

표 1. 알고리즘별 실험결과 (단위 : 횟수)

알고리즘	연산	페이지 크기		
		512B	4K	16K
FMAX	쓰기	120,201	934,035	114,808
	소거	3,514	28,092	3,640
BAST	쓰기	449,280	1,068,925	143,214
	소거	36,649	36,742	3,490
FAST	쓰기	281,214	1,325,746	62,548
	소거	7,099	33,087	1,929
MTBS	쓰기	111,697	648,133	109,949
	소거	3,409	20,218	3,647

3장에서 설명한 실험의 결과는 표 1과 같다. 읽기 연산은 쓰기와 소거 연산에 비해 매우 빠르고, 플래시 메모리의 성능은 쓰기 연산과 소거 연산에 의해 결정되므로 위 결과에서 읽기 연산의 횟수는 제외하였다.

4.1 복사 블록 기법(FMAX)

복사 블록 기법은 쓰기에서 블록의 크기가 16K일 때 가장 좋은 성능을 보이며 소거에서는 512B가 가장 좋은 성능을 보인다. 512B는 섹터의 크기와 동일하며 한 블록에 32개의 페이지를 가지게 된다. 이로 인해 덮어 쓰기가 33번 발생하게 되면 복사 블록에 빈 페이지가 없으므로 합병 연산이 발생한다. 4K의 경우에는 한 블록이 512B를 가진 페이지의 1/4인 8개의 페이지를 가지게 된다. 이로 인해 복사 블록에 9번의 덮어쓰기가 발생하면 합병 연산이 일어나 512B일 때 보다 쓰기와 소거의 횟수가 크게 증가 하게 되어 성능이 좋지 못하다. 16K의 경우에는 블록 단위로 쓰기 연산과 소거 연산이 이루어지기 때문에 플래시 변환 계층 알고리즘에 의한 영향을 받지 않는다.

4.2 로그 블록 기법(BAST)

로그 블록 기법은 쓰기와 소거 연산 둘 다 16K일 때 가장 좋은 성능을 보인다. 512B와 4K에서는 다른 알고리즘과 비교해서 좋지 않은 성능을 보인다. 이는 로그 블록들의 개수가 한정되어 있고 로그 블록에는 동일 블록의 페이지만 기록할 수 있기 때문이다. 그리고 B+ 트리의 삽입 연산은 여러 블록에 작은 크기의 레코드 갱신 연산을 많이 발생시킨다. 이때 로그 블록은 대부분이 비어 있는데도 불구하고 희생 블록으로 선정되어 합병연산이 일어난다. 이것은 한 페이지에 대한 쓰기 연산이 하나의 소거연산을 유발하는 효과가 있다. 16K의 경우에는 블록 단위로 쓰기 연산과 소거 연산이 이루어지기 때문에 순차적인 블록으로 인식하여 합병연산이 아닌 교환연산이 이루어진다. 이로 인해 쓰기와 소거의 횟수를 줄이는 효과를 낳아 가장 좋은 성능을 보인다.

4.3 개선된 로그 블록 기법(FAST)

로그 블록 기법을 개선한 FAST는 쓰기와 소거 연산 둘 다 16K일 때 가장 좋은 성능을 보인다. 또한 다른 페이지 크기에서도 다른 플래시 변환 계층 알고리즘과 비교하여 가장 좋은 성능을 보인다. 512B인 경우 임의 쓰기용 패턴에서 덮어 쓰기가 발생하면 데이터는 임의 쓰기용 로그 블록에 산재되게 된다. 4K일 경우에도 임의 쓰기 패턴의 덮어 쓰기가 발생하면 임의 쓰기용 로그 블록에 기록되는데 4K는 512B보다 페이지의 크기가 8배 크기 때문에 512B보다 4K일 때 합병연산이 자주 발생하게 된다. 따라서 페이지 크기가 512B보다 4K일 때, 쓰기와 소거 연산이 훨씬 많아져 성능이 저하된다. 16K의 경우에는 0번째 페이지에 대한 쓰기 연산일 때 무조건 순차 쓰기로 해석하여 순차 쓰기용 블록에 저장하고 이것은 데이터 블록으로 바뀐다. 그래서 모든 쓰기 연산은 순차 연산이 되어 플래시 변환 계층 알고리즘의 기법을 사용하지 않은 것과 같은 효과가 있다.

4.4 여유 영역 기법(MTBS)

여유 영역 기법은 데이터 영역과 여유 영역에 대해 최적 비율인 1/2로 블록을 구분하고 있다. 복사 블록 기법과 마찬가지로 쓰기에서는 16K가 가장 좋은 성능을 보이며 소거에서는 512B가 가장 좋은 성능을 보인다. 512B에서는 한 블록 당 32개의 페이지를 16개의 데이터 영역과 16개의 여유 영역으로 나누어 사용하고 있다. 16번의 덮어 쓰기가 가능하기 때문에 다른 페이지 크기에 비해 소거 연산 횟수에서 가장 좋은 성능을 보인다. 4K의 경우에는 각각 4개의 페이지로 이루어진 데이터 영역과 여유 영역으로 나누어져 4번의 덮어 쓰기가 가능하다. 이 때문에 5번의 덮어쓰기가 발생하면 합병연산이 일어나 512B에 비해 쓰기 연산과 소거 연산에 대한 횟수가 증가하게 된다. 16K의 경우에는 쓰기 연산의 단위가 블록 단위이기 때문에 한 블록에 쓰기 연산을 수행할 수 없다. 이로 인해 쓰기 발생 시 두 블록으로 나누어 쓰기 연산이 수행 된다. 그리고 덮어 쓰기 연산 역시 쓰기 연산이 일어난 두 블록의 여유 영역에 그대로 쓰게 되며 합병 연산 역시 두 블록씩 소거 연산이 일어나게 된다. 이로 인해 덮어 쓰기가 발생하였을 경우 복사 블록 기법과 마찬가지로 두 개의 블록을 사용하기 때문에 쓰기나 소거 연산에 대해 비슷한 성능을 보인다.

5. 결론 및 향후 연구

B+ 트리에서 페이지 크기에 따른 각 알고리즘을 실험한 결과로 페이지가 소거 단위인 16K일 때 개선된 로그 블록 기법에서 가장 좋은 성능을 보이고 있다. 또한 페이지가 읽기, 쓰기 단위인 512B인 경우 복사 블록 기법과 여유 영역 기법에서 좋은 결과를 보이고 있다. 이 실험에서 B+ 트리의 페이지 크기를 플래시 메모리의 소거 단위인 16K로 했을 때 가장 좋은 성능을 보인다. 이것은 플래시의 읽기, 쓰기 단위인 섹터 혹은 섹터의 배수 단위로 하는 것 보다, 소거 단위인 블록으로 B+ 트리의 페이지 크기를 결정하는 것이 더 좋은 성능을 보인다는 것을 알 수 있다. 하지만 이것은 기존 플래시 변환 계층 알고리즘을 사용하지 않은 것과 동일한 방법이 된다. 따라서 작은 크기의 레코드가 임의의 페이지에 많이 기록(small random write)되는 B+ 트리에 적당한 새로운 플래시 변환 계층이 필요하다는 것을 의미한다. 또한 가변적인 레코드를 고려해서 기존 페이지 레이아웃을 변경하여 성능을 증진시키는 방안도 연구되어야 한다.

6. 참고문헌

[1] Petro Estakhri and Berhanu Iman. Moving sequential sectors within a block of information in a flash memory mass storage architecture, United States Patent, no.5,930,815, 1999.

[2] Jesung Kim, Jong Min Kim, Sam H. Noh, Sang Lyul Min, and Yookun Cho. A space-efficient flash translation layer for compact flash systems. IEEE Transactions on Consumer Electronics, 48(2), 2002.

[3]Sang-Won Lee and Dong-Joo Park. FAST:An efficient flash translation layer for flash memory, Submitted for publication, 2005.

[4] Takayuki Shinohara. "Flash memory card with Block memory address arrangement", United States Patent, no.5,905,993, 1999.