플래시 기반 저장장치에서 사상 테이블의 캐싱 알고리즘 성능 연구*

양수현, 류연승
명지대학교 컴퓨터공학과
e-mail:ysryu@mju.ac.kr

Study on Performance of Caching Algorithms for Mapping Table in Flash-based Storage Devices

Soo-Hyeon Yang, Yeonseung Ryu
Dept of Computer Engineering, Myongji University

요 약

NAND 플래시 메모리 기반의 저장장치의 내부에는 Flash Translation Layer (FTL)이라는 소프트웨어가 사용되고 있다. FTL은 파일 시스템으로부터 요청되는 논리 주소를 플래시 메모리의 물리 주소로 변환하며 이를 위하여 사상 테이블을 사용한다. 일반적으로 사상 테이블의 빠른 접근을 위하여 사상 테이블은 저장장치 내부의 RAM에 유지한다. 최근 저장 공간의 용량이 커지게 되면서 사상 테이블로 인해 요구되는 RAM의 크기도 커지게 되어 사상 테이블을 플래시 메모리에 저장하고 일부만 RAM에 유지하는 캐싱 기법들이 연구되어 왔다. 본 논문에서는 SAT-c 라는 사상 테이블 캐싱 기법을 제안하고 캐시 교체 알고리즘들의 성능을 비교하였다.

1. 서론

최근 차세대 저장장치로서 NAND 플래시 메모리 기반의 저장장치들이 널리 사용되고 있다. 플래시 메모리 기반 저장장치는 하드 디스크와 비교하여 빠른 연산 속도, 저전력소비 등의 장점을 가지고 있어 향후 하드 디스크를 점차 대체할 것으로 예상되고 있다. 플래시 메모리가 하드 디스크에 비해 연산속도가 매우 빠르지만, 다음과 같은 몇 가지 특별한 하드웨어적 특징을 가지고 있다[1].

첫째, 플래시 메모리는 블록들의 배열로 구성되며, 각 블록은 고정된 개수의 페이지로 구성된다. 플래시 메모리에 대한 기본 명령어는 읽기, 쓰기, 삭제(erase)인데 읽기와 쓰기의 기본 단위는 페이지이며, 삭제는 블록 단위로 수행된다. 둘째, 플래시 메모리는 저장되어 있는 데이터를 바로 변경할 수 없으며 데이터가 저장되어 있는 블록에 대해 삭제 연산을 한 후에 변경할 내용을 쓸 수 있다. 삭제 연산은 읽기와 쓰기 연산에 비해 상대적으로 많은 시간이 소요된다. 셋째, 한 블록에 대한 삭제 연산의 횟수가 제한되어있다 (10,000번 ~ 100,000번). 만약 특정한 블록에 삭제 연산이 집중되면 해당 블록은 쓰이지 못하게 되며 플래시 메모리 장치의 수명에 영향을 미치게 된다. 이와 같은 특정 블록의 마모를 피하기 위해 삭제 연산이 모든 블록에 고르게 수행되는 것이 좋다. 이를 마모도 평준화(wear leveling)라고 부른다.

플래시 메모리 상의 데이터의 변경이 요구될 때 데이터의 원래 위치에 변경할 내용을 저장하는 원위치 변경(in-place update) 방식은 블록의 삭제를 요구하기 때문에 쓰기 연산의 성능을 저하시킨다. 이러한 문제점을 해결하기 위해, 플래시 메모리 기반의 저장장치는 Flash Translation Layer (FTL)이라는 특별한 소프트웨어를 사용한다. 플래시 메모리 저장장치 내부의 컨트롤러는 ROM에 있는 FTL을 실행한다. FTL은 파일 시스템으로부터 요청되는 논리 주소를 플래시 메모리의 물리 주소로 변환하고, 변환된 물리 주소에 데이터의 입출력을 수행한다. FTL은 변경할 데이터를 원 위치가 아닌 다른 위치에 저장하고 주소 사상 테이블(address mapping table)을 두어 관리하는 타 위치 변경(out-of-place update) 방식을 수행한다.

일반적으로 사상 테이블의 빠른 접근을 위하여 사상 테이블은 저장장치 내부의 RAM에 유지한다. 최근 저장 공간의 용량이 커지게 되면서 사상 테이블로 인해 요구되는 RAM의 크기도 커지게 되어 사상 테이블을 플래시 메모리에 저장하고 일부만 RAM에 유지하는 캐싱 기법들이 연구되어 왔다. 본 논문에서는 FTL의 사상 테이블을 플래시 메모리의 지정된 영역에 저장하고 필요할 때마다 사상 테이블을 페이지 단위로 나누어 RAM에 적재하는 SAT-c 기법을 연구하였다. 또한, 캐싱 알고리즘으로서 LRU, LFU, Random 정책을 모의실험을 통해 성능을 비

* 이 논문은 2010년도 정부(교육과학기술부)의 재원으로 한국연구재단의 기초연구사업 지원을 받아 수행된 것임 (2010-0021897)

<표 1> 저장장치 용량에 따른 사상 테이블의 크기

저장장치 용량	블록 개수	PMT 크기	BMT 크기	HMT 크기
32 GB	2^{16}	64 MB	512 KB	512 KB + 약 6 MB
256 GB	2^{19}	512 MB	4 MB	4 MB + 약 50 MB
1 TB	2^{21}	2 GB	16 MB	16 MB + 약 200 MB

교하였다. 2장에서는 FTL의 주소변환 기법과 사상 테이블의 문제점을 설명하고, SAT 기법을 소개한다. 3장에서는 제안하는 SAT-c 기법을 설명하고 4장에서 캐싱 알고리즘의 성능을 평가하고 그 실험 결과를 기술한다. 5장에서 결론과 향후연구를 기술한다.

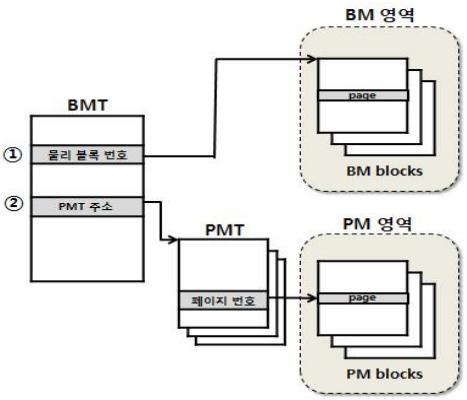
2. 관련 연구

2.1 FTL의 주소 변환 기법

FTL에서 사용되는 주소 변환 기법은 페이지 수준(page-level) 사상, 블록 수준(block-level) 사상, 혼성(hybrid) 사상 기법의 3 종류로 분류된다. <표 1>은 3 가지 사상 기법에 대하여 사상 테이블의 크기를 예를 들어 보여주고 있다. 한 블록의 크기는 512 KB, 한 블록의 페이지 개수는 128 개를 가정하였다. 또한, 사상 테이블의 각 엔트리의 크기는 8 바이트라고 가정하였다.

페이지 수준 사상 기법은 플래시 메모리의 페이지마다 한 엔트리를 가지는 페이지 사상 테이블(PMT: Page Map Table)을 유지하며, 파일 시스템에서 요청되는 논리 섹터 번호를 물리 페이지 번호로 변환한다. FTL이 데이터의 변경 요청을 받으면, 원래 페이지는 무효로 만들고, 새 페이지를 할당하여 요청된 데이터를 기록한다. 페이지 수준 사상 기법은 PMT의 크기가 매우 커질 수 있는 문제점을 가지고 있다. <표 1>에서 볼 수 있는 것처럼 PMT는 수십 MB 이상의 크기가 되므로 SSD의 RAM이 많이 필요하게 되어 사용이 어렵다.

사상 테이블의 크기를 줄이기 위해 블록 수준 사상 기법을 사용할 수 있다. 블록 수준 사상 기법의 사상 테이블은 플래시 메모리의 블록마다 한 엔트리를 갖는다. 따라서, <표 1>에서 보는 것처럼 블록 사상 테이블(BMT)의 크기는 PMT의 크기보다 매우 작다. 블록 수준 사상 기법에서는 논리 주소가 (b_i, p_i)로 구성된다. 이때, b_i는 논리 블록 번호이고 p_i는 블록 내 페이지 번호이다. FTL은 b_i를 사용하여 물리 블록 번호로 사상한다. p_i는 물리 블록 내의 페이지 번호가 된다. 블록 수준 사상 기법은 데이터 변경 연산을 수행할 때 데이터 복사 연산을 해야하는 문제점을 가지고 있다. FTL이 데이터 변경 요청을 받으면 새 블록을 할당하고 새 블록 내의 페이지 p_i에 요청된 데이터를 저장한다. 이때, 이전 블록에 있는 유효 페이지를 새 블록에 복사해야 하는 오버헤드가 요구된다. 또한, 저장장치 용량이 매우 커지면 BMT의 크기도 수십 MB 이상의 크기가 되므로 여전히 사상 테이블의 크기는 문제가 된다.



(그림 1) SAT 기법의 사상 테이블 구조

사상 테이블의 크기 문제와 데이터 복사의 문제 등을 해결하기 위해 혼성 사상 기법이 연구되어 왔다[2, 3]. 혼성 사상 기법은 BMT를 유지하면서, 변경 블록들에 대해서 페이지 수준의 사상으로 관리하는 로그 블록을 할당하고 로그 블록에 변경된 데이터를 저장한다. 즉, 블록 수준 사상과 페이지 수준 사상이 혼용되는 방식이다. 혼성 사상 기법에서는 데이터의 변경 시에 유효 페이지를 복사해야 하는 오버헤드가 없다. 그러나, 저장장치의 용량이 매우 커지면, 사상 테이블의 크기가 커지는 문제점을 여전히 가지고 있다. 예를 들어, 전체 블록의 10%가 변경되는 경우에 혼성 사상 테이블(HMT)의 크기를 <표 1>에서 보이고 있다. 기본적으로 BMT를 유지하면서 추가로 각 변경 블록마다 로그 블록을 위한 하나의 PMT를 만든다고 가정하였다. 로그 블록을 위한 PMT의 크기는 1 KB이다. 왜냐하면, 블록의 페이지 수가 128 개이고 각 페이지마다 한 엔트리를 가지는데 엔트리의 크기가 8 바이트이므로 128 * 8 = 1024이기 때문이다. 저장장치 용량이 1 TB인 경우 HMT의 크기는 200 MB 이상이 되어 많은 RAM이 필요하다. 또 다른 문제점으로는, 로그 블록이 다 채워지면 원래의 데이터 블록과 로그 블록의 유효 데이터를 새 데이터 블록에 복사하는 합병(merge) 연산을 수행해야 한다. 합병 연산은 많은 복사 연산을 수행해야 하는 문제점이 있다. 이를 해결하기 위하여 합병 연산을 수행하지 않는 SAT 기법이 연구되었다.

2.2 SAT(Switchable Address Translation)

SAT은 혼성 사상 기법으로서 변경이 없는 블록은 블록 사상 테이블로 관리하고 변경된 블록은 페이지 사상 테이블로 관리한다[5]. SAT은 플래시 메모리의 블록을 BM(Block-level management) 영역, PM(Page-level management) 영역, 가용(free) 블록 리스트로 구성한다. SAT은 블록마다 한 엔트리를 갖는 블록 사상 테이블(BMT)을 관리한다. 논리 블록에 처음 데이터가 쓰여질 때 SAT은 가용 블록 리스트에서 새 블록을 할당하여 저

장한다. 이 블록은 BM 블록이 되어 BM 영역에 포함된다. BM 영역의 블록은 BMT로 관리된다. (그림 1)에서 ①은 BM 블록에 사상되는 논리 블록의 엔트리이다. 이 엔트리는 사상된 물리 블록의 번호를 갖는다.

논리 블록의 페이지가 변경되면 SAT은 가용 블록 리스트에서 새 블록을 할당하여 요청된 데이터를 저장한다. 이 블록은 PM 블록이 되어 PM 영역에 포함된다. 이때, 변경되는 BM 블록도 PM 블록으로 전환하여 페이지 사상 테이블(PMT)로 관리한다. PMT는 변경된 논리 블록을 위해 하나씩 동적으로 생성된다. (그림 1)에서 ②는 PM 블록으로 사상되는 논리 블록의 엔트리이다. 이 엔트리는 대응되는 PMT의 주소를 갖는다. 이 논리 블록에 쓰여지는 데이터는 PM 영역의 블록에 저장된다.

기존의 혼성 사상 기법과 달리 한 PM 블록이 다 사용되었을 때마다 합병 연산을 수행하지 않고 가용 블록 리스트에서 PM 블록을 새로 할당하여 사용한다. 따라서, 수행 비용이 큰 합병 연산을 수행하지 않아 FTL 성능이 향상될 수 있다.

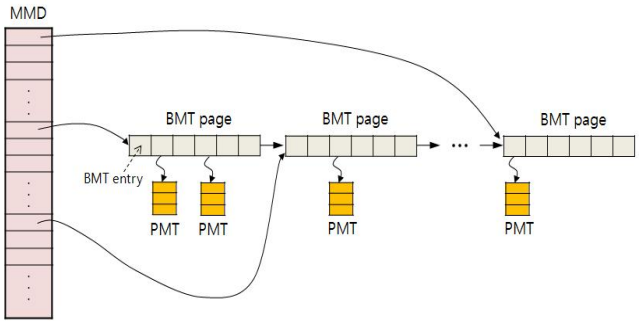
3. SAT-c

3.1 전체 구조

본 논문에서 제안하는 SAT-c 기법은 SAT 기법에 기반을 둔다. 그러나, 일반적인 FTL에도 쉽게 적용이 가능하다. 제안한 기법은 사상 테이블을 플래시 메모리에 저장하며 특정 테이블 엔트리가 참조되어 필요해지면 해당 엔트리가 저장되어 있는 플래시 메모리의 페이지를 RAM에 적재한다. 이처럼 사상 테이블 전부가 RAM에 적재되는 것이 아니므로 사상 테이블의 엔트리 위치를 관리하기 위한 별도의 테이블을 둔다.

SAT-c 기법은 BMT와 PMT들을 플래시 메모리의 특정 영역인 맵 블록(map block)에 저장한다. 맵 블록은 사상 테이블 또는 사상 테이블의 엔트리가 생성되거나 변경될 때마다 그 내용을 로그 방식으로 저장하는 영역이다. SAT-c는 사상 테이블의 한 엔트리가 참조되어 필요해지면 이를 RAM에 적재한다. 엔트리를 RAM에 적재할 때 엔트리가 저장되어 있는 플래시 메모리의 페이지를 RAM에 적재한다. 이것은 플래시 메모리의 읽기/쓰기 단위가 페이지이기 때문이다. 한 페이지 크기가 4 KB이고 BMT의 한 엔트리 크기가 8 바이트라면, 한 페이지에는 512개의 BMT 엔트리가 있게된다. 첫 512개의 BMT 엔트리가 저장된 페이지를 "BMT 페이지 0"이라 하고, 다음 512개의 BMT 엔트리가 저장된 페이지를 "BMT 페이지 1"이라 부른다. 나머지 BMT 엔트리가 저장되는 페이지들도 순서대로 번호를 붙인다. RAM에 적재된 엔트리의 내용이 변경되면 데이터의 신뢰성(reliability)을 위하여 변경된 내용을 즉시 플래시 메모리의 맵 블록에 저장한다.

SAT-c 기법은 BMT 엔트리들의 현재 위치를 관리하기 위해 마스터 디렉토리(master map directiry: MMD)를 RAM에 관리한다. 마스터 디렉토리는 시스템 초기화 과정



(그림 2) SAT-c 기법에서 RAM 상의 사상 테이블 구조

에서 플래시 메모리의 맵 블록을 스캔하여 BMT 엔트리에 대한 정보만을 수집함으로써 만들어진다. 마스터 디렉토리의 엔트리는 각 BMT 페이지의 현재 위치에 대한 정보를 갖는다. 예를 들어, 마스터 디렉토리의 엔트리 값이 0으로 시작한다면 나머지 값은 플래시 메모리의 물리 블록과 그 블록 내의 페이지 번호를 의미한다. 또한, 마스터 디렉토리의 엔트리 값이 1로 시작한다면 나머지 값은 RAM에 적재된 페이지의 메모리 주소를 의미한다. <표 1>에서 저장장치가 1 TB인 경우, BMT는 2^{21}개의 엔트리가 있다. 한 페이지에 512개의 BMT 엔트리가 있다면 마스터 디렉토리는 4096개 (= $2^{21}/2^{9}$)의 엔트리를 갖는다. 마스터 디렉토리의 한 엔트리의 크기가 8 바이트라면 마스터 디렉토리의 크기는 32 KB가 된다. 이 크기는 매우 작기 때문에 마스터 디렉토리는 RAM에 항상 둔다.

PMT의 현재 위치는 BMT에서 관리한다. 2.2절의 SAT 기법에서 설명했듯이 BMT의 엔트리는 대응하는 논리 블록의 변경 여부에 따라 BM 블록 번호 또는 PMT의 주소를 갖는다. PMT의 크기는 일반적으로 플래시 메모리의 페이지 크기보다 작다. 예를 들어, 한 블록의 페이지 수가 128 개이고 PMT 엔트리의 크기가 8 바이트로 가정한다면 1 KB (= 128*8)가 된다. 이때, 플래시 메모리의 한 페이지가 4 KB라면 4 개의 PMT가 저장될 수 있다. 따라서, PMT가 새로 생성되거나 그 내용이 변경되면 PMT 전체를 플래시 메모리의 맵 블록에 저장한다. BMT의 엔트리가 PMT의 주소를 갖는 경우, 이 엔트리가 RAM에 적재되어 있다면, 엔트리는 RAM의 PMT 주소와 맵 블록 내의 PMT 주소(블록 번호, 블록 내 페이지 번호) 값을 갖는다. 그러나, 이 엔트리가 RAM에 없고 플래시 메모리의 맵 블록에만 저장된 상태라면 엔트리는 맵 블록 내의 PMT 주소만 갖는다.

3.2 교체 알고리즘

SAT-c 기법은 BMT 페이지와 PMT를 저장하는 RAM 캐시의 최대 크기를 제한한다. 만약, 새로운 BMT 페이지 또는 PMT가 필요하여 이를 새로 RAM에 적재할 때, RAM 캐시 상의 BMT 페이지와 PMT 들이 캐시 크기를 초과하게 되면 SAT-c 기법은 교체 알고리즘을 수행한다.

교체 알고리즘은 희생 BMT 페이지를 결정하고 이를 RAM에서 제거한다. 만약, 희생 BMT 페이지 내의 BMT 엔트리가 PMT를 갖고 있다면 PMT들도 RAM에서 제거한다.

희생 BMT 페이지의 결정은 다음 세 가지 방법이 있으며, 본 논문에서는 이 세 가지 방법의 성능을 평가하였다. 첫째, 무작위로 결정하는 RAMDOM 알고리즘, 둘째, 가장 오래동안 참조되지 않은 BMT 페이지를 결정하는 LRU(Least Recently used) 알고리즘, 셋째, RAM에 적재된 이후로 가장 적게 참조된 BMT 페이지를 결정하는 LFU(Least Frequently used) 알고리즘이 있다.

4. 성능평가

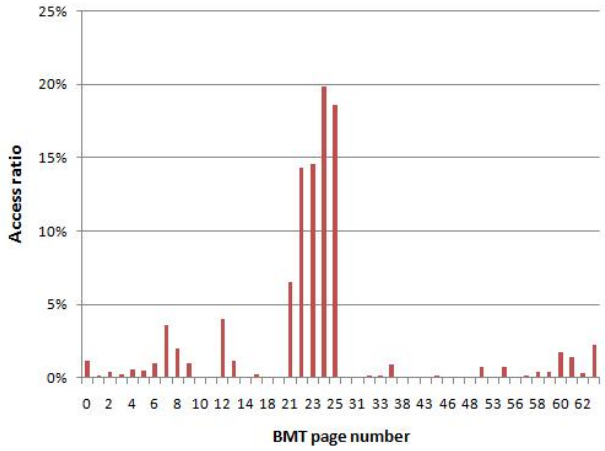
본 논문에서 제안한 SAT-c 기법의 캐시 교체 알고리즘을 평가하기 위하여 트레이스 기반 시뮬레이터를 C 언어로 구현하였다. 구현한 시뮬레이터는 실제 디스크 입출력 트레이스를 기반으로 수행되며 사상 테이블이 사용하는 RAM 크기를 계산한다. 플래시 메모리는 32 GB 크기를 사용하였고 플래시 메모리의 파라미터 값을 <표 2>에서 보여주고 있다.

<표 2> 플래시 메모리 파라미터

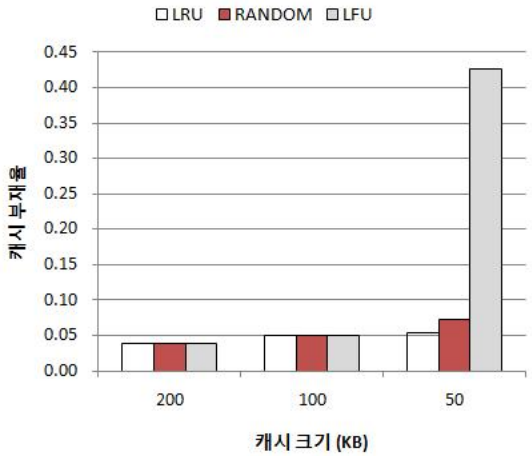
파라미터	값
페이지 크기	4 KB
블록의 페이지 수	128
블록 수	65536

실험에 사용할 워크로드를 위해 윈도우즈 XP가 탑재된 노트북 PC에서 여러 응용 프로그램 (웹 브라우저, 미디어 플레이어, 편집 등)을 수행하면서 디스크 입출력 트레이스를 수집하였다. 트레이스 수집을 위해 diskmon 유틸리티를 사용하였다.

시뮬레이터에서 한 BMT 페이지에 512개의 BMT 엔트리가 저장된다. 총 BMT 페이지 수는 64개이다. (그림 3)은 BMT 페이지에 대한 접근 비율을 보여주고 있다. 일부 BMT 페이지들에 대한 접근이 편중되고 있어 참조의 지역성(locality of reference) 성질이 존재함을 알 수 있다.



(그림 3) BMT 페이지의 참조 율



(그림 4) 교체 알고리즘의 캐시 부재율 비교

(그림 4)는 3.2 절에서 기술한 세 가지 캐시 교체 알고리즘의 캐시 부재율을 비교하여 보여주고 있다. 캐시 크기는 50 KB에서 200 KB까지 변화를 주었다. 캐시 크기가 100 KB 이상으로 클 때는 캐시 공간이 충분하여 세 가지 교체 알고리즘의 성능은 모두 우수하였다. 그러나, 캐시 크기가 50 KB로 작을 때는 LRU 알고리즘이 캐시 부재율이 제일 낮아 성능이 제일 우수하였으며, LFU 알고리즘은 캐시 부재율이 급격하게 높아져 성능이 좋지 않았다.

5. 결론

본 논문에서는 플래시 메모리 기반 저장 장치에서 FTL이 사용하는 주소 사상 테이블의 캐싱 기법을 연구하였다. 모의 실험 결과, 주소 사상 테이블의 접근에 대한 지역성이 존재함을 발견하였으며, 사상 테이블의 교체 알고리즘으로는 LRU 알고리즘이 캐시 부재율이 제일 낮아 성능이 우수함을 알 수 있었다.

참고문헌

[1] Samsung Electronics, “K9GAG08U0M 2G * 8Bit NAND flash memory data sheet,” http://www.samsungelectronics.com.

[2] J. Kim, J. Kim, S. Noh, S. Min, and Y. Cho, “A Space-efficient Flash Translation Layer for Compactflash Systems,” IEEE Transactions on Consumer Electronics, vol. 48, no. 2, May, 2002, pp. 366-375.

[3] S. Lee, D. Park, T. Chung, D. Lee, S. Park, and H. Song, “A log buffer-based flash translation layer using fully-associative sector translation,” ACM Trans. Embedded Computing Systems, vol.6, no.3, Jul. 2007.

[4] Y. Ryu, “Method for controlling flash memory device,” Korea Patent 10-0638638, Sep. 2004.

[5] Y. Ryu, "SAT: Switchable Address Translation for Flash Memory Storage," IEEE Computer Software and Applications Conference (COMPSAC), Jul. 2010.