OpenSSD 플랫폼에서의 섹터 매핑 FTL의 구현 및 성능 측정

한규화 안정철 신동군
성균관대학교 성균관대학교 성균관대학교
Hgh6877@gmail.com luckyjc7@skku.edu dongkun@skku.edu

Implementing a Sector-level Mapping FTL Algorithm at OpenSSD

Kyuhwa Han, Jeongcheol Ahn, Dongkun Shin

요 약

SSD에 내장되어 있는 FTL은 매핑 기법에 따라서 성능 및 비용이 크게 달라진다. 기존의 SSD가 대부분 페이지나 슈퍼페이지 단위의 매핑을 사용했지만, 실제 워크로드에서는 그보다 작은 크기의 쓰기 요청이 많아 섹터 단위의 매핑이 요구되고 있다. 본 연구에서는 섹터 매핑 기법을 사용하는 FTL의 성능과 오버헤드에 대해서 살펴보기 위해서, 실제 SSD 제품에서 사용되었던 컨트롤러 기반의 OpenSSD라는 SSD 개발 플랫폼에서 섹터 매핑 FTL을 구현하고 실험을 진행하였다. 효과적인 섹터 매핑의 구현을 위해서 OpenSSD가 제공하는 하드웨어의 기능을 활용하고, 대용량의 매핑 정보를 효율적으로 관리하기 위한 기법들을 제안하고 있다. 실험 결과, 섹터 매핑 기법이 작은 크기의 쓰기 요청에 대해 슈퍼 페이지 매핑 기법보다 월등히 좋은 성능을 보이지만, 매핑 테이블의 오버헤드가 성능에 미치는 영향도 크다는 것을 알 수 있었다.

1. 서 론

플래시 메모리는 저전력과 비휘발성으로 인해 주로 모바일폰이나 MP3 플레이어에 많이 사용되어 왔지만, 최근에는 가격이 내려가고 용량이 커짐에 따라 SSD라는 형태로 기존의 하드디스크 드라이버를 대체하기 시작하고 있다. SSD는 여러 개의 플래시 칩과 DRAM 버퍼, 컨트롤러를 내장하고 있으며, 멀티 채널과 멀티 웨이 구조를 활용하여 여러 칩에 동시에 읽기/쓰기가 가능하기 때문에 일반 플래시 칩보다 높은 데이터 전송 대역폭을 제공한다.

플래시 메모리 칩은 여러 블록으로 구성되어 있으며, 블록은 여러 개의 페이지를 가지고 있고, 페이지 단위로 읽기/쓰기를 해야 하며 블록 단위로 지우기를 해야 한다. 페이지에 데이터를 기록하기 전에 해당 블록을 반드시 지워야 하며, 지우기 횟수의 제한이 있다. 이러한 특성을 다루기 위해서 FTL(Flash Translation Layer)이라는 소프트웨어 레이어가 사용된다. SSD 내부에는 FTL이 내장되어 있어서 호스트 컴퓨터는 일반 블록 장치로 SSD를 접근할 수 있다.

SSD 내부에서 FTL은 파일시스템이 보낸 논리 주소를 플래시 메모리의 물리 주소로 변환해 준다. 최근의 SSD는 주소 변환 기법으로 주로 페이지 매핑 기법을 사용하고 있다. 페이지 매핑은 과거의 블록 매핑에 비해서는 그 성능이 아주 뛰어나지만, 페이지 크기보다 작은 쓰기 요청에 대해서는 Read-Modify-Write 연산이 필요하여 성능이 떨어진다. 이것은 또한 잦은 가비지컬렉션(garbage collection, GC)를 유발시켜 SSD의 수명을 단축시킨다. 특히, SSD가 슈퍼페이지(동시에 쓰기 가능한 여러 페이지들의 묶음) 단위의 매핑을 사용하는 경우에는 그 문제가 더욱 심각해진다.

본 연구는 지식경제부 및 정보통신산업진흥원의 지원사업의 연구결과로 수행되었음

그림 1은 PC 워크로드에 대해서 쓰기 요청의 크기를 분석한 자료로서, SSD의 슈퍼 페이지 사이즈를 8KB라고 가정하고, 페이지 크기에 정합(aligned) 여부를 조사한 그래프이다. 모든 워크로드에서 8KB에 정합하지 않는 쓰기 요청이 50퍼센트 이상 발생함을 알 수 있다. 이것은 보다 작은 크기의 매핑 기법이 필요하다는 것을 보여준다.

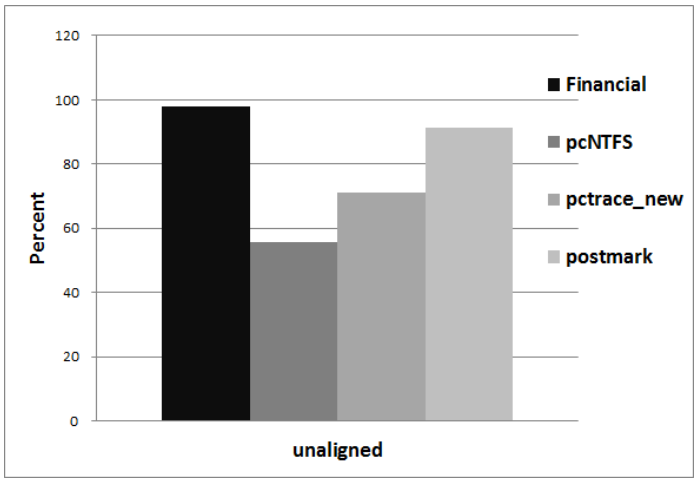

그림 1 PC 워크로드의 unaligned 분석 결과

본 논문에서는 기존의 페이지 매핑 기법을 보완하기 위해서, 512B 섹터 단위의 매핑 기법의 유효성을 검증하고자 한다. 섹터 매핑 기법은 작은 크기의 쓰기 요청에 대한 처리가 빠르며, 공간의 활용도가 높아 지우기 횟수를 줄일 수 있다. 반면 매핑 테이블의 크기가 크기 때문에 저장 공간이 많이 필요하므로, 이러한 오버헤드에 대한 관찰을 바탕으로 효과적인 구조의 FTL 설계가 요구된다.

기존의 많은 연구들[2,3,4]이 새로운 FTL 기법을 제시할 때,

시뮬레이션 환경에서 성능을 검증한 것과는 달리, 본 논문에서는 실제 SSD 개발 환경인 OpenSSD[1]에 섹터 매핑 FTL 기법을 실제로 구현하고 성능 및 오버헤드를 측정하였다. 특히, Sector Log[4]기법은 로그버퍼만을 섹터 매핑으로 관리하는 기법을 제안하고 시뮬레이터에서 실험하였으나, 본 연구에서는 전체 스토리지 영역을 섹터단위로 관리하는 기법을 제안하고 이를 OpenSSD에서 검증하였다. 또한, 섹터 매핑 기법의 구현을 위해서 OpenSSD가 제공하는 하드웨어 장치의 기능을 활용하여 효과적인 섹터 매핑 기법을 제시하고 있다.

본 논문은 다음과 같이 구성된다. 2절에서는 OpenSSD 플랫폼에 대하여 설명한다. 3절에서는 본 논문에서 제안하는 FTL의 구조와 알고리즘을 설명한다. 4절에서는 각각의 FTL 알고리즘을 실제 하드웨어인 OpenSSD[1] 상에서의 읽기/쓰기 성능을 측정한 결과를 보인다. 마지막 절인 5절에서는 결론을 제시한다.

2. OpenSSD 플랫폼

OpenSSD는 INDILINX사의 barefoot 컨트롤러 기반의 SSD 펌웨어 개발 플랫폼이다. 그림 2에서 볼 수 있듯이 OpenSSD는 ARM7 Core와 64MB DRAM, 96KB SRAM 및, 여러 개의 플래시 메모리칩 등으로 구성되어 있다. OpenSSD는 플래시 메모리칩의 수를 조정함으로써 전체 용량을 조절할 수 있는 데, 16bit 4 채널과 8웨이/채널을 활용하여 최대 64개의 칩을 연결할 수 있다.

OpenSSD는 특별한 하드웨어 연산을 제공하는 데, DRAM의 특정 비트를 세팅하는 연산, 클리어 하는 연산, 읽어오는 연산 등이 그것이다. 제안하는 섹터 매핑 FTL은 효과적인 GC를 위해 _mem_search_min_max 라는 연산을 활용했다. 이 연산은 선택한 DRAM 영역에서 설정한 단위를 기준으로 가장 큰 값이나 가장 작은 값의 오프셋을 리턴하는 함수이다.

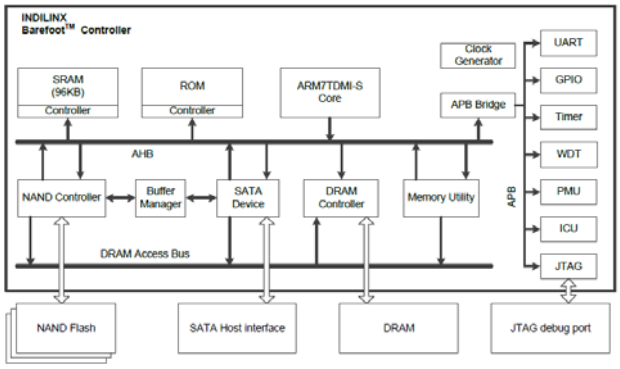

그림 2 OpenSSD 하드웨어 구조

본 연구에서는 섹터 매핑 FTL과의 비교를 위해서, 슈퍼 페이지 매핑 알고리즘인 Greedy FTL을 사용하였다. Greedy FTL은 유효한 페이지 수가 가장 적은 블록을 victim으로 선정하기 때문에 GC를 위하여 매 쓰기마다 해당 블록의 유효한 페이지 수를 체크하고 기록하는 오버헤드가 있다. GC는 쓰기 명령 시 사용할 페이지가 없을 경우 발생하며, 하나의 빈 블록을 GC 블록으로 할당하여, GC 발생 시 GC 블록에 유효한 페이지를 옮긴다.

3. 섹터 매핑 FTL 알고리즘

그림 3은 제안하는 섹터 매핑 방식 FTL의 전체적인 구조를 보여준다. 플래시 메모리는 데이터 영역과 섹터 매핑 테이블 영역으로 구분된다. 섹터 매핑 테이블은 논리 섹터 번호(LSN)와 물리 섹터 번호(PSN)간의 매핑 정보를 제공한다. DRAM에는 호스트로부터 전송된 데이터를 임시 보관하는 병합 버퍼와 매핑 테이블의 일부만을 가지고 있는 CMT(cached mapping table), 그리고 GC의 희생 블록 선정 시 참조하는 유효 섹터 카운트 테이블이 있으며, SRAM에는 병합 버퍼에 있는 데이터의 섹터 단위 매핑 정보가 보관된다.

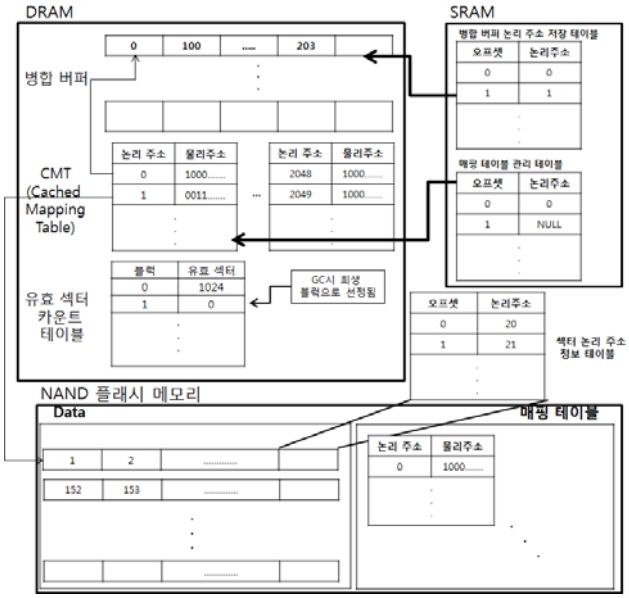

그림 3 섹터 매핑 FTL의 전체 구조

3.1 매핑 테이블

매핑 테이블은 논리 섹터 번호 순으로 정렬된 구조로 유지된다. 모든 논리 섹터 번호에 대하여 매핑 정보를 저장하여야 하기 때문에 테이블의 크기는 매우 크다. 하나의 LSN을 위해서 4B를 할당한다. 예를 들어, 4GB의 SSD를 위해서는 4B×4GB/512B=32MB의 매핑 테이블이 필요하다. 그러므로 큰 용량의 SSD에서는 모든 매핑 정보를 DRAM에 저장하는 것은 불가능해진다. 매핑 테이블을 플래시 메모리에만 저장하게 되면, 매번 데이터 접근시마다 플래시 메모리를 추가로 접근해야 하므로 오버헤드가 매우 크다.

그러므로 본 기법에서는 DFTL[2]과 유사하게 Cached Mapping Table(CMT)을 사용한다. 전체 매핑 테이블은 플래시 메모리에 보관되며, 매핑 테이블의 일부만을 DRAM의 32MB 크기의 CMT에 로딩하여 사용한다. 만약 읽기/쓰기 요청을 처리하기 위해 필요한 매핑 정보가 DRAM에 없을 경우에는 필요한 정보를 플래시 메모리의 매핑 테이블에서 1MB단위로 읽어온다. 매핑 테이블 단위가 1MB인 이유는 SRAM에 저장된 매핑 테이블 관리 테이블의 크기가 SRAM의 크기를 초과하지 않게 하기 위함이다. 매핑 테이블 관리 테이블은 DRAM에 매핑 테이블이 올라가 있는 경우에는 DRAM의 위치 오프셋이, 그렇지 않은 경우에는 NULL값이 저장된다. CMT는 LRU와 유사하지만 좀 더 간단한 알고리즘을 통해 관리된다.

호스트로부터 전송된 데이터는 임시로 병합 버퍼에 기록되므로, 읽기 요청을 처리할 때, 해당 섹터는 병합 버퍼 또는 플래시 메모리에서 찾을 수 있다. 해당 섹터가 병합 버퍼에 있는 경우에는 매핑 테이블에 병합 버퍼 상의 섹터 오프셋이 저장된다. 데이터의 위치를 구분하기 위해서, 매핑

테이블의 PSN값의 MSB(Most Significant Bit)을 사용하는 데, 병합 버퍼에 섹터가 있는 경우에는 MSB를 1로, 플래시 메모리에 있는 경우에는 MSB를 0으로 세팅한다.

3.2 병합 버퍼

본 논문이 제시하는 FTL은 섹터단위의 쓰기 명령을 수행하기 위하여 병합 버퍼를 사용한다. 그림 3에서 볼 수 있듯이 호스트로부터 전송된 데이터들은 32KB 슈퍼 페이지 크기가 될 때까지 병합 버퍼에 저장한다. 병합 버퍼는 32KB 크기의 버퍼를 32개 가지고 있어 크기가 1MB이다. 병합 버퍼에 저장되어 있는 섹터들의 논리 섹터 번호 정보는 SRAM에 저장된다.

63개의 섹터들이 모이면, SRAM에 저장된 이들 섹터들의 논리 주소를 슈퍼 페이지의 마지막 섹터로 붙여서 64개의 섹터로 슈퍼 페이지를 구성한 뒤에 플래시 메모리에 기록한다. 쓰기 명령이 수행되는 동안 호스트로부터 전송된 데이터는 새로운 병합 버퍼를 할당받아 모으게 된다.

3.3 GC 정책

섹터 매핑 기법의 GC 정책은 Greedy FTL에서 사용하고 있는 방식과 동일하다. 희생 블록의 선정은 유효한 섹터의 개수가 가장 적은 블록을 선정하는 방식을 사용한다. 그림 3에서 볼 수 있듯이 블록의 유효한 섹터의 수를 저장하는 유효 섹터 카운트 테이블이 DRAM에 위치한다. 유효 섹터 카운트 테이블은 매 쓰기마다 업데이트되며 관리된다. OpenSSD 상의 하드웨어 연산을 통하여 유효 섹터가 가장 적은 블록을 찾아서 희생 블록으로 사용한다. GC중 유효한 섹터의 복사를 위해서는 페이지 내의 섹터들의 논리 섹터 주소 값이 필요하며, 이것은 해당 페이지의 마지막에 기록되어 있다.

4. 실험 결과

실험에서는 OpenSSD를 사용해 제안하는 섹터 매핑 기법을 Greedy FTL과 비교하였다. 실험은 iometer를 통하여 진행되었으며, 512B 랜덤 쓰기, 4KB 랜덤 쓰기, 128KB 연속 쓰기, 128KB 연속 읽기의 성능을 측정하였다.

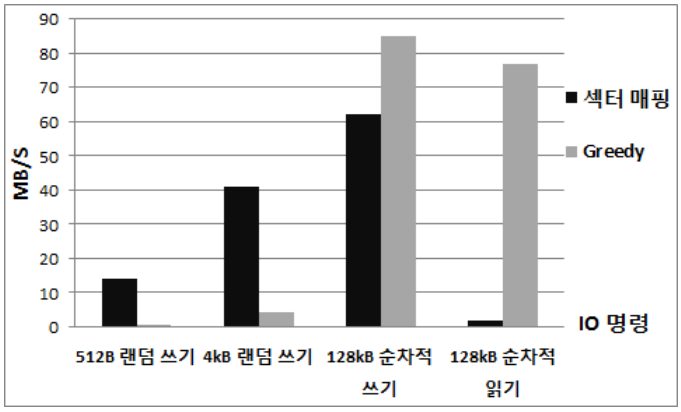

그림 4 각 FTL 별 IO 명령에 대한 성능 평가(MB/S)

그림 4에서 알 수 있듯이 작은 단위 IO에 대한 섹터 매핑 FTL의 성능은 슈퍼 페이지 매핑 FTL의 20배 이상이 된다. 이것은 슈퍼 페이지 매핑 FTL이 작은 단위 IO에 대하여 오버헤드를 가지고 있기 때문이다. 하지만 128KB에 대한 읽기/쓰기 명령에 대해서는 슈퍼 페이지 매핑 기법의 성능이 더 좋게 나온다. 이것은 매핑 정보를 섹터 단위로 관리하는 오버헤드 때문이다. 특히, 현재 구현한 버전에서는 순차 읽기에 대해서도 각 섹터별 매핑 정보를 참조하기 때문에 큰 오버헤드가 존재한다. 이것은 연속된 구간에 대한 효과적인 매핑 기법들[3]을 활용하면 개선될 수 있다.

또한 섹터 매핑 기법의 FTL에서 CMT 관리를 위한 오버헤드를 측정하기 위한 실험도 진행하였다. CMT 관리 오버헤드가 없는 섹터 매핑 FTL의 성능 측정을 위하여 SSD의 용량을 줄이는 방법을 사용하였다. 표 1은 CMT 관리 오버헤드 측정 실험의 결과를 보여준다. 표 1에서 알 수 있듯이 오버헤드가 존재하지 않는 경우가 50배 이상 성능이 좋다.

즉, CMT 관리 오버헤드를 줄이는 기법이 요구되는데, 섹터 매핑 기법과 페이지 매핑 기법을 혼합한 하이브리드 형식의 FTL[3,4]을 통해 전체 매핑 테이블의 크기를 줄이는 방법이 사용될 수 있을 것이다.

표 1 매핑 테이블 오버헤드 측정 결과 (IOPS)

	w/o CMT O/H	w/ CMT O/H
512B 랜덤 쓰기	28370	540

5. 결론

페이지 매핑 기법이 블록 매핑 기법을 사용하는 FTL 보다 작은 단위 IO 처리에서 강한 면을 보여준다. 하지만 페이지 단위보다 작은 단위의 IO 처리에서는 오버헤드가 존재하는 것이 사실이다. 본 논문은 페이지 매핑의 약점이 될 수 있는 작은 단위의 IO 명령에 대한 해결 방안으로 섹터 매핑 기법에 대하여 알아보았다.

섹터 매핑 기법이 작은 단위의 IO 명령에 대하여 페이지 매핑보다 좋은 성능을 보였지만 매핑 테이블의 오버헤드를 더한다면 이 또한 제대로 된 성능을 발휘하지 못한다. 이 문제를 해결하기 위해서는 페이지 매핑 기법과 섹터 매핑 기법의 단점을 서로 보완해 줄 수 있는 FTL이 필요하다. 필자는 페이지 매핑과 섹터 매핑을 동시에 사용하는 하이브리드 매핑 기법을 추가 연구할 것이다. 이를 통하여 본 논문으로 해결하지 못한 작은 단위의 IO 명령의 문제를 해결하고자 한다.

참고문헌

[1]OpenSSD project http://www.openssd-project.org

[2] Gupta, a. et al., “DFTL: A Flash Translation Layer Employing Demand-based Selective Caching of Page-level Address Mappings”, ACM SIGPLAN Notices, Vol. 44 No.3 2009

[3] Park, D. et al., “CFTL: A Convertible Flash Translation Layer Adaptive to Data Access Patterns”, ACM Performance evaluation review, Vol.38 No1, 2010

[4] Seongwook Jin, et al. "Sector Log: Fine-Grained Storage Management for Solid State Drives," 2011 ACM Symposium on Applied Computing (ACM SAC 2011), Mar. 2011.