메모리 제약을 위한 순서 증가 페이지 매핑 FTL 기법

김효진 ○, 신동군
성균관대학교 소프트웨어플랫폼학과
hyojin0313@skku.edu, dongkun@skku.edu

Increasing-Order Page-Level Mapping FTL for Memory-constraint Flash Storage Systems

Hyojin Kim○, Dongkun Shin
Department of Software Platform, Sungkyunkwan University

요 약

SSD는 HDD와 다르게 덮어쓰기가 불가능하고, 읽기/쓰기 단위와 삭제의 단위가 다르다는 특징이 있다. 이러한 특징으로 인해 SSD는 플래시 변환 계층을 필요로 한다. 플래시 변환 계층의 알고리즘은 SSD 성능에 큰 영향을 미친다. 본 연구에서는 파티션이라는 개념을 도입하여 메모리 사이즈를 줄인 IOPM을 제안한다.

1. 서 론

최근 SSD는 low latency, high bandwidth, low power consumption 등의 장점으로 인해 휴대용 단말기나 노트북 시장을 선점하였다. SSD 시장은 지난 5년간 연평균 45.1%로 성장하였다.

SSD에 주로 사용되는 NAND Flash는 덮어쓰기가 불가능하며 페이지 단위로 읽기/쓰기를 하지만 삭제는 블록 단위로 하는 등의 기존의 HDD와 다른 특성을 지니고 있다. 따라서 호스트로부터 전달받은 입출력을 수행하기 위해서는 이를 SSD에서 수행 가능한 형태로 변환하는 플래시 변환 계층(Flash Translation Layer, 이하 FTL)이 필요하다. FTL은 논리 주소와 물리 주소간의 매핑 정보를 저장하며 블록 부족 시 유효 페이지를 복사하고 블락을 삭제하여 재사용할 수있도록 하는 가비지 컬렉션 등의 기능을 수행하게 된다. FTL의 동작방식은 SSD 성능에 큰 영향을 미치게 된다.

페이지 단위 매핑 기법은 논리 주소와 물리 주소의 매핑 정보를 페이지 단위로 관리한다. 따라서 성능은 우수하지만 매핑 테이블의 크기가 크다는 단점이 있다. 이런 단점을 보완 것이 혼합 사상 기법이다. 혼합 사상 기법은 페이지 매핑 정보를 블록 단위로 관리한다. 블록 단위로 매핑을 저장하게 되면 복사가 자주 발생하게 된다. 혼합 사상 기법에서는 이를 보완하고자, 덮어쓰기는 로그 블록에 out-of-place로 쓰여지게 된다. 페이지 단위 매핑 기법에 비해 메모리 공간을 적게 사용한다는 장점이 있지만, 합병로 인한 비용이 발생하게 된다. 페이지 단위 매핑에서 매핑 테이블의 메모리 사용을 줄이기 위해 DFTL[1]이 고안되었다. DFTL은 페이지 단위로 매핑테이블을 관리하는 대신, 캐싱을 통해 메핑테이블의 메모리 사용량을 줄인다. CPM[2]은 여기서 더 나아가 지역성를 활용하여 Garbage Collection(이하 GC)의 비용을 줄인다. 그러나 두 방법 모두 매핑테이블를 캐싱하여 메모리 사용량을 줄였으며 따라서 캐싱으로 인한 비용증가가 있다는 단점이 있다.

본 연구에서는 입출력이 순서가 증가하는 형태로 전달될 가능성이 높다[3]는 점에 착안하여 매핑테이블을 파티션을 통해 관리하는 순서 증가 페이지 단위 매핑 기법 (Increasing Order Page-level Mapping, 이하 IOPM)을 제안하고자 한다.

2. 관련 연구

2.1 DFTL(Demanded-based FTL)

DFTL은 페이지 단위 매핑에서 요구되어지는 메모리 크기를 줄이고자 고안된 기법이다. 매핑 테이블을 플래시 메모리에 저장하며 필요에 의해 캐싱한다. 메모리 크기를 줄일 수 있다는 장점이 있다. 그러나 cache miss가 많이 나거나 cache 크기가 작아지면 성능이 저하된다는 단점이 있다.

2.2 CPM(Clustered Page-Level Mapping)

CPM은 연속된 논리 주소를 클러스터라는 단위로 묶어서 관리한다. 이로 인해 매핑 정보도 클러스터 단위로 가지게 된다. 따라서 DFTL에 비해 매핑테이블을 감소시킬 수있으며, 지역성이 향상되므로 가비지 콜렉션의 비용 또한 줄일 수 있다.

3. IOPM(Increasing Order Page-level Mapping)

이 논문은 2016년도 정부(미래창조과학부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임 (No. 2016R1A2B2008672)

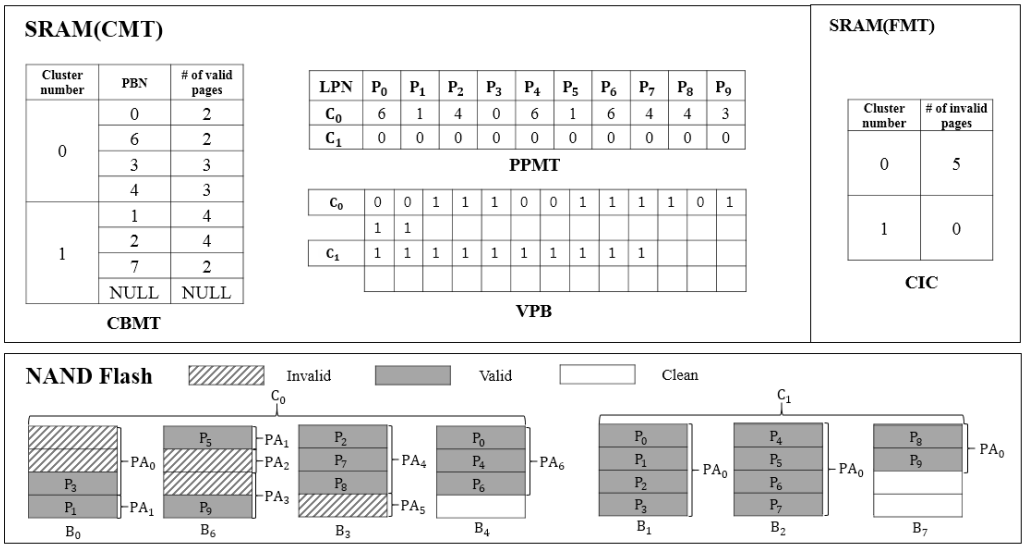

그림 1 IOPM 구조

3.1 Motivation

본 논문에서 제안하고자 하는 IOPM은 지역성을 향상시키고자 CPM에 기본 motivation을 두고 있다. 그러나 앞서 설명한 DFTL이나 CPM은 페이지 단위 매핑에 비해 전체 매핑 테이블 크기를 감소시키지는 못하였다. 캐싱을 통하여 메모리의 사용량을 감소시켰을 뿐이다. 반면 IOPM은 파티션을 통하여 매핑 테이블의 크기를 감소시킨다.

3.2 IOPM Architecture

IOPM은 CPM과 마찬가지로 연속된 논리주소를 cluster라는 단위로 관리한다. 대신 매핑 테이블의 크기를 줄이기 위해 PPMT(Page Partition Mapping Table)와 VPB(Valid Page Bitmap, 이하 VPB)를 이용해 매핑 테이블을 관리한다. PPMT는 논리페이지가 속한 파티션의 정보를 알려준다. VPB는 물리페이지의 유효성을 비트맵으로 관리한다. PPMT와 VPB를 통해 논리페이지와 물리페이지의 매핑 정보를 관리할 수 있다. CBMT(Cluster-Block Mapping Table, 이하 CBMT)은 클러스터에 할당된 블록과 블록 내 유효페이지 정보를 담고 있다. CBMT를 통해 삭제 될 블락을 선정하거나 주소 변환을 할 수 있다. CIC(Cluster-level Invalid page Count, 이하 CIC)는 클러스터 별 무효페이지의 개수를 가지고 있다. 그림 1은 IOPM의 구조를 표현하였다. 모든 매핑 정보는 메모리 위에 적재되어 있다. 구체적인 동작은 3.3과 3.4에서 설명하도록 한다.

3.3 IOPM Address Translation

논리 주소를 물리 주소로 변환하는 과정은 다음과 같다.

1) PPMT를 통해 입출력 요청을 받은 논리 주소가 속한 파티션 정보를 알아낸다

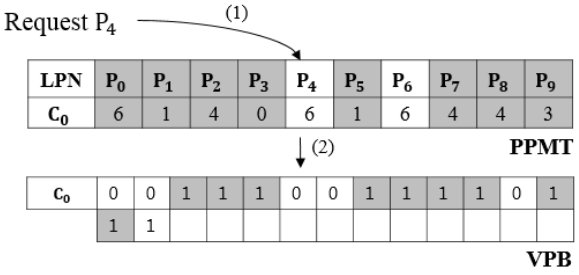

그림 2 IOPM 주소 변환

2) 입출력 요청의 논리 주소 번호가 속한 파티션보다 앞선 파티션에 속한 논리 주소의 개수와 입출력 요청의 논리 주소가 속한 파티션과 같은 파티션에 속하고 요청된 논리 주소보다 작은 논리 주소의 개수를 센다. 그림 2의 PPMT에서 회색으로 칠해진 페이지들이 해당 페이지들이다.
3) VPB에서 2)에서 구한 개수만큼 1을 센다. 그 다음으로 오는 1의 위치가 요청된 논리 주소에 매핑된 물리주소이다. 그림 2의 VPB맵에서 마지막 1이 요청된 P_4의 물리주소이다.

그림 2의 LPN은 논리주소를, C0는 클러스터의 번호를 뜻한다. 그림 2의 PPMT는 각 페이지 P_i가 속하는 파티션 정보를 가지고 있다. VPB는 페이지의 유효성을 나타낸 맵이며, 1이면 해당 물리주소의 페이지는 유효함을, 0이면 무효함을 나타낸다.

3.4 IOPM 동작 설명

3.4.1 Read/Write

호스트로부터 읽기 요청을 전달받은 경우 IOPM은 앞서

3.3에서 설명한 것처럼 논리주소를 물리주소로 변환한다. 해당 물리주소에 위치한 데이터를 읽는다.

쓰기 요청인 경우 논리 주소가 파티션에 속해 있는 지를 확인한다. 속해있다면 덮어쓰기인 경우이므로 논리주소를 물리주소로 변환하고, 해당 물리주소의 데이터를 무효화시킨다. 그 다음 새로운 쓰기 동작을 한다. 새로운 쓰기 동작이란 PPMT내 논리주소의 파티션 정보를 업데이트 시켜주고, 데이터를 쓴 후 VPB내에서 해당 위치를 유효처리 하는 것이다. PPMT 업데이트 시 요청된 논리주소가 마지막으로 요청된 논리주소보다 작으면 파티션 번호를 증가시켜 업데이트해준다. 덮어쓰기가 아닌 경우에는 새로운 쓰기 동작만 진행한다.

3.4.2 GC

GC는 크게 세가지로 분류된다. IntraGC, InterGC, 그리고 PartitionGC이다. IntraGC의 경우 일반적인 FTL과 같은 방식으로 동작한다. 클러스터 내에서 무효 페이지가 가장 많은 블락을 삭제 대상으로 선정한다. 삭제 대상 블락의 유효 페이지들을 현재 쓰여지고 있는 블락에 쓰기를 수행 한 뒤 삭제 대상인 블락을 삭제해준다.

InterGC의 경우 지워진 블락의 부족 등으로 인해 다른 cluster에서 free block을 대신 생성해주는 GC 이다. 이 경우도 IntraGC와 마찬가지로 동작하게 된다. 다만 GC를 위한 블락과 요구 되어지는 쓰여지지 않은 블락을 생성할 때까지 반복하게 된다.

PartitionGC의 경우 파티션의 개수가 메모리공간으로 인해 한정되어 있어서 발생하는 GC 기법이다. 유효페이지가 가장 적은 삭제 대상 파티션 2개를 선정하여, 유효페이지들을 복사하고, 이를 재배열한 뒤 현재 쓰여지고 있는 block에 쓰게 된다. 그리고 나서 삭제 대상 파티션은 삭제된다.

4. 실험

4.1 매핑 테이블 크기 비교

전체 플래시의 크기를 32GB, 페이지의 크기를 8KB, 블락당 페이지의 개수를 128개, 클러스터의 크기를 16MB, 그리고 클러스터당 최대 할당될 수 있는 블락의 개수를 24개, 파티션은 클러스터당 4bit씩 할당한다고 가정한다. 이때 각 FTL별 매핑 테이블의 크기는 다음의 표 1과 같다. IOPM은 CPM과 DFTL에 비해 대략 8분의 1정도의 공간을 요구한다.

	PMT	CBMT	PPMT	VPB
DFTL	16MB			
CPM	16MB	192KB		
IOPM		1MB	2MB	512KB

표 1. 매핑 방식 별 메모리 요구 공간 비교

4.2 Workload

IOPM과 CPM, DFTL의 성능을 비교하기 위해, 시뮬레이터를 구현하여 실험을 진행하였다. 실험은 fileserver와 varmail workload를 사용하여 실험하였다. 각 workload는 F2FS 파일시스템 위에서 돌려졌다. 순차적으로 전체 용량의 70%를 쓰기 한 뒤에 workload를 돌렸다. 다음은 FTL의 GC 수행시간과 매핑 정보 로딩시간을 누적그래프로 나타낸 것이다. IOPM의 성능은 DFTL과 비슷하게 나온 것을 확인할 수 있다. 이는 실제 workload들이 순차적으로 쓰기를 수행할 가능성이 높기 때문이다. 표 2를 참조하면 하나의 파티션의 크기의 평균을 확인할 수 있다.

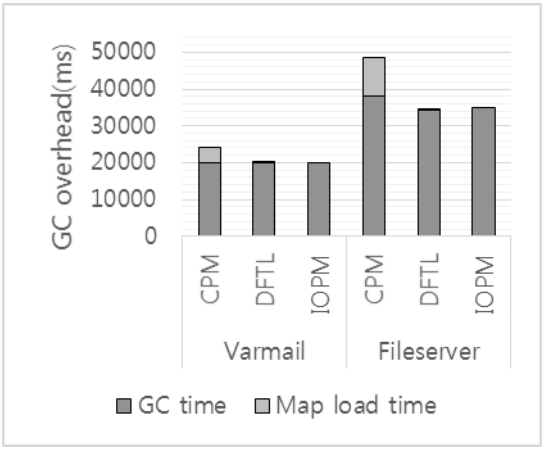

그림 3. GC 오버헤드

Workload	Size(개)	Workload	Size(개)
webserver	342.0	fileserver	586.7
varmail	115.0	webproxy	294.0

표 2. Workload별 *파티션의 평균 페이지 개수

5. 결론 및 향후 연구

IOPM은 DFTL이나 CPM에 비해 메모리 사이즈를 줄일 수 있다는 장점이 있다. 따라서 DFTL과 CPM이 가지고 있는 매핑 로딩 비용이 들지 않는다. 그러나 Partition의 크기가 크고 PartitionGC가 자주 일어나게 될 경우, 비용이 증가한다는 단점이 있다. 이를 개선 할 수 있는 PartitionGC 알고리즘에 대한 향후 연구가 필요하다.

참고 문헌

[1] A. Gupta, Y. Kim, and B. Urgaonkar, “DFTL: a flash translation layer employing demand-based selective caching of page-level address mappings”, In Proc. of the USENIX ASPLOS, pp 229-240, 2009.

[2] H. Kim, D. Shin, “Clustered Page-Level Mapping for Flash Memory-Based Storage Devices”, IEEE Trans. On Consumer Electronics 61(1):44-51, 2014

[3] 안정철, 신동군, “순서 증가를 인식하는 블록 단위 매핑 기법을 사용하여 로그 블록 방식의 임의 쓰기 성능 개선”, 한국정보과학회 2016년 한국컴퓨터종합학술대회 논문집, 1649-1651, 2016