플래시 메모리 GC (가비지 콜렉션) 오버헤드를 줄이기 위한 블록 링크드 리스트 기법

구소현*, 김성수*, 정태선*
*아주대학교 컴퓨터공학과
e-mail: uriinhyoung@ajou.ac.kr

Block Linked List Scheme to Reduce GC (Garbage Collection) Overhead in Flash Memory

Sohyun Koo*, Sungsoo Kim*, Tae-Sun Chung*
*Dept of Computer Engineering, Ajou University

요 약

플래시 메모리는 소형 저장 장치뿐만 아니라 대용량 저장장치까지 응용되고 있다. 하지만 기존의 하드디스크 (HDD)와 다르게 플래시 메모리는 읽기, 쓰기, 소거 연산의 속도가 다르고 쓰기 전 지우기 (erase before write)라는 특성 때문에 FTL의 한 메커니즘인 GC (Garbage Collection)를 수행할 때 많은 오버헤드가 발생한다. 이에 이 논문은 DRAM의 공간을 효율적으로 활용하고 유효한 페이지 복사와 소거 연산의 횟수를 줄여 전체적인 플래시 메모리 GC 오버헤드를 줄이기 위한 블록 링크드 리스트 기법을 제안한다. 블록 링크드 리스트 기법은 같은 LBN에 해당하는 데이터를 로그 블록에 적고 해당 로그 블록들을 링크드 리스트로 관리해 소거 연산을 미룰 수 있다. 링크드 리스트들에 관한 정보는 DRAM에 테이블 형태로 적는다. 이때 테이블에는 블록 주소들이 적히므로 페이지 단위로 링크드 리스트를 관리하는 다른 기법에 비해 DRAM의 공간을 효율적으로 활용하게 된다.

1. 서론

플래시 메모리는 고속 입출력이 가능하고 크기가 작고 소비전력이 적은 특성을 가진다. 이러한 특성으로 인해 스마트 폰, USB 등과 같은 작은 저장 장치뿐만 아니라 최근 데이터 센터에서도 SSD (Solid State Disk)에 클라우드 컴퓨팅 기법을 적용하려는 사례가 늘고 있다 [1].

기존의 하드디스크 (HDD)와 다르게 플래시 메모리는 읽기, 쓰기, 소거 연산의 속도가 다르고 쓰기 전 지우기 (erase before write) 특성을 가진다. 쓰기와 소거 연산의 속도가 읽기 연산의 속도보다 더 느리고 읽기, 쓰기 연산은 섹터 단위로 이루어지는 것과 달리 소거 연산은 블록 단위로 이루어지는 특성이 있다. 따라서 플래시 메모리에서는 쓰기와 소거 연산을 줄이는 것이 주요 연구 분야이다.

FTL (Flash Translation Layer)은 플래시 메모리를 구성하는 소프트웨어 계층으로 파일시스템으로부터 받은 논리 주소를 플래시 메모리의 물리 주소로 사상하는 메커니즘[4] 과 업데이트 시 발생하는 많은 유효하지 않은 페이지 (invalid page)를 지우는 메커니즘인 GC (Garbage Collection)을 포함한다[2]. 일반적으로 플래시 메모리의 공간보다 쓰여질 데이터가 더 많을 경우 기존의 데이터를 업데이트해야 하므로 GC가 필요하다. GC에는 세가지 단계가 있는데 먼저 지워질 희생 블록 (Victim Block)을 고르고 희생 블록에 있는 유효한 페이지 (valid page)를 프리 블록에 복사한다. 마지막으로 희생 블록을 지워서 프리 블록 리스트에 추가한다. 이 때 유효한 페이지를 다른 블록으로 복사할 때와 희생 블록을 지울 때 GC 오버헤드가 발생한다. 따라서 본 논문에서는 유효한 페이지 복사와 소거 연산 수를 줄여 전체적인 플래시 메모리 GC 오버헤드를 줄이고 DRAM을 효과적으로 사용하기 위한 블록 링크드 리스트 기법을 제안한다.

2. 관련연구

본 논문에서는 논리 주소를 물리 주소로 사상하는 섹터단위 주소사상, 블록단위 주소사상을 혼합하는 방식인 FAST기법의 기본 구조를 바탕으로 플래시 메모리 GC 오버헤드를 줄이기 위한 블록 링크드 리스트 기법을 설계하였다. FAST에서는 데이터 블록과 로그 블록을 가지며 데이터 블록에 데이터가 가득 차 있는 상태일 때 로그 블록에 데이터를 적는다. 이 때 원본 데이터블록들에 대해서는 블록단위 주소사상을 하고, 로그블록에 쓰이는 임의의 섹터들에 대해서는 완전연관 섹터단위 주소 사상을 한다 [4]. 하지만 로그블록에 해당하는 사상 테이블이 DRAM에 위치해 있기 때문에 DRAM에 섹터 수 큼의 공간이 필요하다.

MNFTL [3] 에서는 유효한 페이지 복사와 소거 연산 수를 줄여 GC 오버헤드를 감소시키기 위한 방법으로

(그림 1) 쓰기 연산 예시

물리 페이지 주소 (PPN)를 링크드 리스트 형태로 관리하는 방법을 제안하였다. 하지만 이는 여전히 별도의 페이지 단위 주소사상을 필요로 하므로 DRAM에 페이지 수만큼의 공간을 사용한다. 본 논문은 하나의 LBN (논리 블록 주소)에 해당하는 로그블록을 DRAM에서 블록 단위로 관리해 DRAM을 효과적으로 사용할 수 있고 GC 오버헤드를 감소시킬 수 있는 기법을 제안한다.

3. 블록 링크드 리스트 기법

본 논문에서 제안하는 블록 링크드 리스트 기법은 논리 블록 주소가 같은 데이터들을 로그 블록에 적고 해당 로그 블록들을 링크드 리스트로 연결해 관리한다. DRAM에는 데이터 블록을 위한 블록 단위의 데이터 사상 테이블과 로그 블록을 위한 블록 단위의 데이터 사상 테이블 그리고 링크드 테이블이 위치한다. 링크드 테이블에는 하나의 논리 블록 주소에 해당하는 링크드 리스트를 위한 정보들이 있다 (*논리 블록 주소*와 링크드 리스트의 *헤더 포인터*, 데이터가 적힌 섹터를 파악하는 *섹터 포인터*, 하나의 논리 블록 주소에 몇 개의 블록이 연결되어 있는지를 파악하는 *블록 카운터*).

3.1. 쓰기 연산

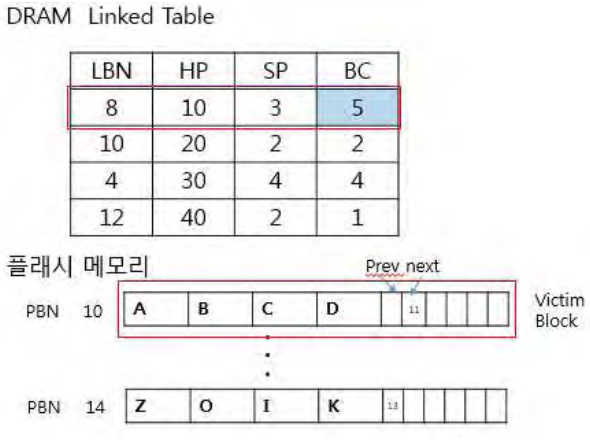
그림 1은 블록 링크드 리스트 기법의 쓰기 연산을 보여주고 있다. 이 예는 한 블록에 4개의 섹터가 있고 데이터 블록은 가득 차 있다고 가정한다. 먼저 데이터 A와 논리 섹터 주소 35가 쓰인다. 논리 섹터 주소 35는 섹터 개수인 4로 나눠지고 몫은 LBN (논리블록주소), 나머지는 offset값으로 취한다. 데이터 블록이 가득 차 있기 때문에 사용 가능한 로그 블록 <PBN (물리블록주소) 10>의 SP (섹터 포인터)가 가리키는 위치 (초기값 0)에 데이터를 적고 SP값을 증가시킨다. DRAM에 있는 링크드 테이블에 해당 LBN을 적고 처음 할당 받은 로그 블록의 PBN을 HP (헤더 포인터)에 적는다. 현재 링크드 리스트에 연결되어 있는 블록의 개수를 BC (블록 카운터)에 업데이트 한다. 데이터가 쓰이면 로그블록의 여분 영역에 데이터의 offset을 적는다. 하나의 로그 블록이 꽉 차면 다른 로그 블록에 이전 로그 블록의 여분영역에 새 로그 블록의 PBN을 적는다. 새 로그 블록의 여분영역에도 이전 로그 블록의 PBN을 적어둬서 읽기 연산 시 적절한 읽기 연산을 가능하게 한다.

3.2. GC 메커니즘

그림 2에서는 블록 링크드 리스트 기법의 GC 메커니즘을 설명하고 있다. 플래시 메모리에 로그 블록이 꽉 차게 되면 GC를 수행하게 되는데 첫 단계로 희생 블록을 고른다. 희생 블록은 DRAM의 링크드 테이블에서 BC 값이 가장 높은, 즉 가장 많은 블록이 연결되어있는 링크드 리스트의 HP의 값을 PBN으로 하는 블록으로 고른다. 그림 2에서는 LBN 8의 링크드 리스트에 가장 많은 수인 5개의 로그 블록이 연결되어있다. LBN 8 링크드 리스트의 가장 앞에 위치한 로그 블록인 PBN 10이 희생 블록이 된다. 가장 많은 로그 블록이 연결되어 있는 링크드 리스트는 해당 LBN이 업데이트가 가장 빈번하게 일어난 것을 의미하고 맨 앞 블록은 가장 오래된 데이터들이 있으므로 유효하지 않은 페이지들이 있을 확률이 다른 로그 블록들보다 상대적으로 높다. 해당 링크드 리스트의 HP는 희생 블록의 여분 영역의 다음 로그 블록의 PBN으로 업데이트 하고 BC도 하나를 낮춘다. 그림 2에서는 BC가 4가 된다.

4. 결론 및 향후 연구

본 논문에서는 같은 LBN에 해당하는 데이터가 쓰인 로그 블록을 링크드 리스트로 관리하는 기법을 제안한다. 링



(그림 2) GC 메커니즘 예시

크드 리스트를 블록으로 관리하기 때문에 기존의 MNFTL [3]에서 페이지 단위로 링크드 리스트를 관리할 때보다 DRAM의 공간을 효율적으로 사용한다. 이 기법에서는 유효하지 않은 페이지가 가장 많이 있는 블록을 희생 블록으로 결정한 후 GC 메커니즘을 수행해서 유효한 페이지 복사 횟수와 소거 연산 횟수를 줄여 전체적인 GC 오버헤드를 줄일 수 있다.

향후 연구로는 해당 기법의 GC 메커니즘에서 희생 블록을 고를 때 마모도 평준화 (Wear Leveling)을 고려하여 유효하지 않은 페이지가 적은 블록을 고르면서 전체 플래시 메모리에서 희생 블록이 고르게 골라질 수 있는 방법을 연구해 보고 해당 기법을 기존의 다양한 FTL 기법과 함께 유효한 페이지 복사 횟수와 소거 연산 횟수 등에 대한 성능 평가를 할 수 있겠다.

5. 사사글

이 논문은 2014년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임 (No. 2013R1A1A2A10012956).

참고문헌

[1] 김관용, “스토리지 업계가 플래시에 집중하는 이유”, 아이뉴스24, 2013. 04. 11

[2] Dongzhe Ma, Jianhua Feng, and Guoliang Li, “A Survey of Address Translation Technologies for Flash Memories”, ACM Computing Surveys, Vol. 46, No. 3, Article 36, 2014. 01

[3] Zhiwei Qin, Yi Wang, Dui Lio, Zili Shao, and Yong Guan, “MNFTL: An Efficient Flash Translation Layer for MLC NAND Flash Memory Storage Systems”, ACM DAC, 2011. 06

[4] Sang-won Lee, Dong-Joo Park, Tae-Sun Chung, Dong-Ho Lee, Sangwon Park and Ha-Joo Song, “A Log Buffer-Based Flash Translation Layer Using Fully-Associative Sector Translation”, ACM Transactions on Embedded Computing Systems, Vol. 6, No. 3, Article 18, 2007. 07