Deduce-for-Speed

SSD의 쓰기연산 감소는 호스트 워크로드의 실행시간 감소에 직접적인 연관성이 없다.

대조적으로, 스토리지 디바이스의 중복제거는 여러개의 페이지로 구성된 유닛이나 청크단위로 수행된다.

이 때, 피지컬 페이지 접근에 있어서 순서관계가 파괴된다.(파일 중간에 중복된 페이지가 존재할 때)이는 상당한 읽기연산을 퍼포먼스를 감소시킨다.

(HDD보다 랜덤읽기가 빠른 SDD라 할지라도) 대부분의 경우 읽기요청이 쓰기요청보다 더 많이 발생한다.

더 나아가 중복된 페이지는 그렇지 않은 페이지보다 더 접근할 확률이 높다.(수정이 자주 발생하는 페이지라서 그런 듯..)

결과적으로, 중복제거에 의한 읽기 퍼포먼스 저하가 스토리지 중복제거로 인한 이점을 상쇄한다.

따라서 본 연구는 Dedup-for-Speed(DFS)를 제안한다.

DFS는 FTL을 지원하는 중복제거이다. 다만 기존에 존재하는 중복제거 접근방법과 달리, DFS의 주 목적은 읽기성능을 개선하는 것이다.

DFS는 이를위해 중복제거를 통해 얻은 추가적인 여유공간을 고속플래시모드를 사용하는데 투자한다(?).

점진적으로 증가하는 스토리지 용량에 대한 수요를 감당하기 위해 최근에는 플래시 셀에 비트 저장 밀도를 높이는 방법이 사용되고 있다. 현재 상업용 SSD는 multi-level cell(MLC) TLC, QLC를 사용한다. 이들은 각각 하나의 플래시 셀에 2개, 3개, 4개의 비트를 저장할 수 있다. 그러나 플래시 고밀도 플래시 메모리는 그렇지 않은 플래시 메모리보다 용량은 늘어나지만 읽기/쓰기 성능이 떨어진다.

고밀도 플래시 칩의 구조는 근본적으로 저밀도 칩의 구조와 동일하고 저밀도 모드로 프로그래밍되어 있다. 대부분의 상업용 SSD컨트롤러는 플래시 모드를 변경할 수 있다. 몇몇 SSD는 MLC 칩을 pseudo-SLC모드로 프로그래밍 하여 빠른쓰기버퍼로 사용한다.

DFS는 중복된 페이지를 저밀도 플래시 모드에 저장한다. 이를 fast flash modes라고 부른다. 물리적 페이지에서 더 많은 복제가 발생할수록 추가공간을 사용하지 않고 페이지를 프로그래밍하는 데 사용할 수 있는 밀도가 낮아진다. 중복된 페이지를 저장하기 위한 프로그래밍 밀도는 지정된 SSD의 지정된 용량을 보증하면서 가장 낮은 것으로 결정된다. 예를들어

QLC SSD에서 DFS는 두개의 중복 페이지를 pseudo-MLC페이지에 저장하고 4배로 중복된 페이지는 pSLC에 저장한다.

DFS와 기존의 중복제거 메커니즘은 반복되는 페이지 쓰기를 생략함으로써 쓰기 퍼포먼스를 개선할 수 있다. 그러나 실제적인 fast flash로의 중복페이지의 마이그레이션은 호스트에서의 IO요청 수행으로 인해 발생하는 인터럽트를 피하기 위해 백그라운드에서 수행된다. 이러한 중복제거와 페이지 마이그레이션의 병행으로 인해 DFS는 최소한의 오버헤드로 읽기와 쓰기 레이턴시를 개선할 수 있다. DFS는 또한 일반적인 스토리지 중복제거로 인한 이점도 얻을 수 있다.(GC감소, wear-and tear of flash chip)

DFS는 SSDsim에서 구현되고 6개의 실제 워크로드를 사용하여 평가됨

Background and related work

Deduplication in Flash SSDs

SSD의 FTL은 LPN과 PPN을 매핑한다. LPN은 호스트에 보여짐

따라서 두개의 LPN이 식별될 경우 중복제거가 수행되어 하나의 PPN으로 매핑되도록 한다. 이러한 구조적 이점으로 인해 대부분의 중복제거 연구는 SSD에서 진행되었다.

SSD는 명시된 용량보다 대략 7~28% 정도의 추가적인 여유공간을 가지는데 이를 over-provisioning space라고 부른다.

SSD내에서의 중복제거는 OPS를 증가시키는 효과를 가진다.

OPS영역이 커지면 SSD는 긴 GC빈도를 유지하고 GC 시 복사되는 유효페이지의 수를 줄일 수 있다. (왜? : ssd 쓰기 시 free space가 부족할 때 OPS를 사용) 결과적으로 쓰기증폭계수가 감소 , 성능개선. 감소된 쓰기증폭계수는 또한 플래시 셀의 수명을 증가시킬 수 있다. 그러므로 SSD의 중복제거는 쓰기증폭계수와 GC오버헤드를 개선할 수 있다. 더 나아가 쓰기연산동안 수행되는 중복제거는 플래시 메모리의 쓰기를 생략할 수 있다. 이로인해 쓰기지연과 증폭계수를 감소시킬수 있다.

플래시 페이지 단위로 매핑이 수행되는 page-based FTL에서, 페이지들은 기본적으로 청크 단위로 사용된다. 이는 중복제거 단위로도 사용된다. CAFTL, RemapSSD DRACO는 일반적으로 플래시 페이지 단위로 들어오는 쓰기 요청을 중복 제거합니다.

페이지의 유효성을 확인하기 위해 바이트반위로 페이지를 비교하는 것은 추가적인 읽기연산과 시간소모를 유발한다. 그러므로 대부분의 중복제거기법은 페이지의 해시값을 얻어 SHA-1, CRC 같은 해시함수를 page fingerprint로 사용한다. 페이지보다 상당히 작은 fingerprint를 사용하는 것은 페이지를 구별하는 오버헤드를 현저히 감소시킨다.

비록 fingerpring가 페이지보다 상당히 작은 공간만을 차지하지만, 모든 페이지에 대한 fingerprint를 저장하는 것인 일반적으로 불가능하다. 왜냐하면 상당히 많은 DRAM 공간을 요구하기 때문이다(inline deduplication: 쓰기연산 수행전 중복데이터를 제거하는 것, out-of-line deduplication: 모든 데이터를 저장한 후에 주복제거를 수행하는 것).또한 상당히 많은 fingerprint저장은 중복제거를 위한 검색오버헤드에 의해 쓰기지연을 발생시킨다. 이러한 문제를 해결하기 위해 CAFTL과 SmartDedup은 inline 중복제거와 out-of-line 중복제거를 둘다 수행한다. CAFTL은 쓰기요청에서 content-based 샘플링을 수행하여 중복데이터 가능성이 더 높은 페이지의 fingerprint를 생성한다. 생성된 fingerprint는 fingerprint 공간에 저장된다. 생성되지 않은 페이지의 fingerprint는 out of line 중복제거 중 생성되고 SSD가 idle 상태일 때 비교한다. 플래시메모리를 저장장치로 사용한는 스마트 디바이스의 SmartDedup 은 2단계의 fingerprint저장소를 가진다. in-memory와 on-disk. inline 중복제거는 메모리의 fingerprint저장소를 사용하여 수행된다. 반면에 out-of-line deduplication은 디스크에 저장된 fingerprint를 사용한다.

SSD의 GC와 웨어레벨링 때문에 LPN에 매핑된 PPN 위치는 FTL에 의해 수시로 이동될 수 있다. 그러므로 FTL이 물리페이지를 이동할 때 동반되는 L-P 매핑은 반드시 수행되어야 한다. 이를 위해 물리적 페이지를 참조하는 LPN은 페이지 쓰기 시 물리적 페이지의 대역 외 공간에 기록된다.

중복제거가 수행될 때 물리적 페이지에 매핑되는 LPN의 수는 시간에 따라 증가한다. 그러나 플래시 메모리는 덮어쓰기가 불가능하고 대역 외 공간은 크기제한이 있기 때문에 FTL은 해당 물리적 페이지의 OBB영역에 추가 LPN매핑을 기록할 수 없다. 그러므로 중복제거는 DRAM에 추가적인 자료구조를 통해 관리하는데 P-N매핑을 기록할 수 있다. CAFTL은 2단계의 간접매핑 구조를 사용한다. 만역 페이지가 중복되지 않았다면 L2P테이블(기본 매핑 테이블)의 항목에 해당 PPN이 포함된다. 그러나 중복된 LPN은 보조 매핑테이블의 인덱스를 가리킨다. 보조 매핑테이블은 LPN에 매핑된 PPN을 포함한다. 물리 페이지의 위치가 변경될 때 모든 LPN은 해당 보조 매핑 테이블의 PPN을 업데이트 하여 변경된 PPN에 매핑된다.

RemapSSD는 GC 및 웨어레벨링이 플래시 블록단위로 수행된다는 사실에 중점을 두고 플래시 블록 단위로 분할된 NVRAM 세그먼트에 P2L매핑을 저장한다. 중복제거가 발생하면 P2L 매핑정보는 flash block의 세그먼트 공간에 log-structured방식으로 저장된다. 페이지가 이동할 때 P2L매핑은 NVRAM세그먼트를 읽어서 찾는다.

파일시스템과 데이터베이스는 저널링이나 write-ahead logging을 사용하여 crash 일관성과 원자적인 업데이트를 보장한다. 이는 동일한 데이터락 프로세스에서 두번 쓰여진다.

따라서, 어떤 FTL은 호스트가 제공하는 반복적인 쓰기정보를 받음으로 중복제거를 수행한다. 또한 페이지 내 중복데이터의 오프셋이 다른 경우 중복 제거가 발생하지 않는 문제를 극복하기 위해 콘텐츠 기반 중복제거를 위한 가변크기 청크 기술이 제안되었다.

그러나 컨텐츠 기반 중복제거는 높은 연산 오버헤드를 생성하는데 이는 청크의 브레이크 포인트를 결정하기 위해 로빈함수를 활용하여 분산 해시함수를 수행하기 때문이다.

Deduplication Impact on Reads

현대 SSD의 높은 성능은 내부 다이와 채널에 병렬적으로 접근함으로써 달성될 수 있다. 연속된 주소의 페이지는 스트라이핑 되어 플래시 칩에 저장된다.

그러므로 플래시 칩의 병렬성을 최대화 하기 위한 순차적인 접근은 높은 성능을 달설할 수 있다. (SSD의 내부 병렬처리: 물리적인 한계로 플래시 칩의 전송 대역폭이 최대 30~40mb인 문제를 해결하기 위해 여러개의 플래시 칩에 내부적으로 동시에 접근할 수 있도록 하는 기술 FTL에 의해 물리적으로 연속된 주소가 아니라도 순차적인 데이터를 다른 칩에 저장할 수 있음)

대조적으로 랜덤 엑세스는 내부 병렬화를 완전이 활용할 수 없다. 이는 순차적인 접근보다 상당히 높은 지연률을 보인다.

하드디스크 시대부터 현대의 SSD까지 스토리지 시스템은 가능한 연속적인 데이터를 저장하도록 설계되어왔다. 그러나 만약 데이터 중간에 중복제거로 인한 fragment가 존재하면 그 fragment는 물리적으로 스토리지 공간을 차지하지 않고 이전에 저장된 데이터를 가리킬 것이다. 그러므로 순차적인 데이터 읽기는 더이상 완전히 순차적으로 접근하지 않는다. 이러한 행동으로 인해 순차읽기는 중복제거 스토리지에서 낮은 성능을 보이는 경향이 있다.

백업 시스템은 몇몇의 동일한 파일을 포함한다(저널링, 스냅샷 등). 따라서 중복제거로 인해 상당한 공간이 절약될 수 있다.

몇몇의 백업 시스템에서는 읽기와 쓰기는 컨테이너 단위로 수행되는데, 이는 파일시스템의 블록 크기보다 큰 크기를 가진다.

4MB크기의 컨테이너 중간에 있는 4KB 청크가 중복제거가 되었다고 가정하자. 그러면 스토리지에서 컨테이너를 읽는 하나의 순차읽기는 여러개의 읽기연산으로 분할될 것이다. 이는 fragmentation문제로 인한 성능저하로 이어진다.

이러한 문제를 해결하기 위해 Lillibridge는 3가지 기술을 제안했다. 캐시 크기를 늘림, 컨테이너 캐핑, 포워드 어셈블리 영역 사용 했다. Ng and Lee는 가상머신 이미지 백업 시스템에서 최신 이미지에 대한 읽기를 최적화 하기 위해 역 중복제거의 개념을 제안했다.

이것은 오래된 데이터로부터 중복을 제거한다. 이렇게 한면 이전 데이터에서 중복 항목이 제거되므로 새 데이터 레이아웃을 가능한 순차적으로 유지하며 fragmentation을 이전 데이터로 이동한다.

zou는 새로운 관리친화적인 중복제거 프레임워크를 제안했다.

이는 이상적인 데이터 레이아웃을 생성하는 데이터분류 접근법을 사용함으로써 백업 워크로드의 지역성을 유지한다.

Zhou는 DupHunter를 제안했다. 이는 새로운 도커 레지스트리 아키텍쳐로 도커 이미지 레이어를 중복제거한다. 그들은 이미지 다운로드 시간이 중복제거로 인한 fragmentation으로

길어지는 것을 보였다. 이를 해결하기 위해 두가지의 스토리지 계층으로 관리한다.

Our Approach

Design overview

DFS는 중복제거 FTL으로 읽기와 쓰기 성능을 모두 향상시킨다. 기존에 존재하는 중복제거 FTL과 같이 DFS는 플래시 메모리 사용량과 쓰기성능을 증가시키기 위해 중복데이터 저장을 방지한다.

또한 DFS는 중복제거로 얻은 여유공간을 활용하여 고속 플래시 모드에서 중복 데이터를 저장한다. pSLC 또는 pMLC모드를 사용하여 중복 페이지의 읽기 지연을 개선한다.

이전에 언급한 대로, 빠른 플래시 모드는 대부분의 플래시 메모리에서 지원한다. 이는 MLC, TLC, QLC플래시 칩을 포함한다. 그리고 대부분의 SSD컨트롤러는 플래시 블럭이 빠른 플래시 모드를 사용하도록 프로그램 할 수 있다.

상업 SSD는 쓰기 버퍼에 pSLC를 사용한다. 읽기가 자주 발생하는 데이터의 경우, 만약 데이터가 빠른 플래시 모드를 사용하면 데이터의 읽기접근 성능이 상당히 개선될 수 있다.

그러나 페이지 쓰기에서 빠른 플래시 모드는 두배에서 4배까지 더 많은 스토리지 용량을 소비한다. (왜?)

그러므로 빠른 플래시 모드로 저장된 데이터의 총합이 증가하게 되면 SSD 용량의 효율성이 감소한다.

그러나 특정 페이지의 중복 수에 따라 페이지를 고속 플래시 모드로 저장하면 SSD의 용량을 효과적으로 보존할 수 있다.

예를들어 두번 중복된 페이지들이 QLC SSD에서 pMLC모드로 저장되어 있으면 유효 용량은 동일하게 유지된다.

또한 4번 이상 중복된 페이지들이 pSLC 모드로 저장되어 있으면 유효 용량은 증가할 것이다.

이러한 정책을 기반으로 DFS는 페이지 중복 횟수에 따라 적절한 플래시 모드를 선택하여 저장한다.

DFS에서는 중복 페이지들은 하나의 물리 페이지에 저장된다. 그러나 여러개의 논리페이지가 이를 참조한다.

만약 각 논리 페이지의 읽기접근 빈도가 같으면 중복도가 높은 페이지의 읽기접근 빈도도 높다.(Section 4에서 증명)

비록 DFS가 빠른 플래시 모드에서 중복 페이지들만 저장하지만 복제된 페이지는 읽기 빈도가 높아 효율성이 높을 것으로 기대된다. 또한 DFS는 in-line 중복제거도 수행한다.

그러므로 중복 쓰기를 감소함으로써 DFS는 또한 쓰기응답시간 개선에 대한 이점을 가진다. 이는 기존 중복제거 효과와 동일하다

다음설명에서는 하나의 셀에 4개 비트를 저장할 수 있는 QLC SSD를 가정한다. 다만 모든 DFS구조는 고밀도 플래시 셀을 사용하여SSD를 일반화할 수 있다.

DFS는 추가적인 중복제거와 프로모션 모듈을 가진다. 그림1처럼 일반적인 페이지 기반 FTL에 기반한다. dedup 모듈은 호스트로부터 받아온 쓰기요청에 대해 inline 중복제거를 수행한다. DFS는 페이지 기반 FTL을 기반으로 하므로 페이지 단위 중복제거를 수행한다.

프로모션 모듈은 프로모션 연산을 담당하는데, 이는 중복된 페이지를 적절히 빠른 플래시 모드 페이지로 이동시키는 역할을 수행한다.

DFS는 후처리 프로모션 메커니즘을 통해 프로모션 연산으로 인해 호스트의 읽기와 쓰기가 방해받는 것을 방지한다. 이는, 결과적으로 성능에 불리한 영향을 끼치는 것을 최소화한다.

빠른 플래시 모드 페이지에서 쓰기요청이 발생하면 DFS는 처음 QLC 페이지에 페이지를 쓴다. 차례로 변경된 중복횟수에 따라 DFS는 빠른 플래시 모드의 페이지를 강등하는데, 이는 빠른 플래시 모드에 저장된 중복제거된 페이지를 적절한 플래시 모드 페이지로 이동시키는 것이다.

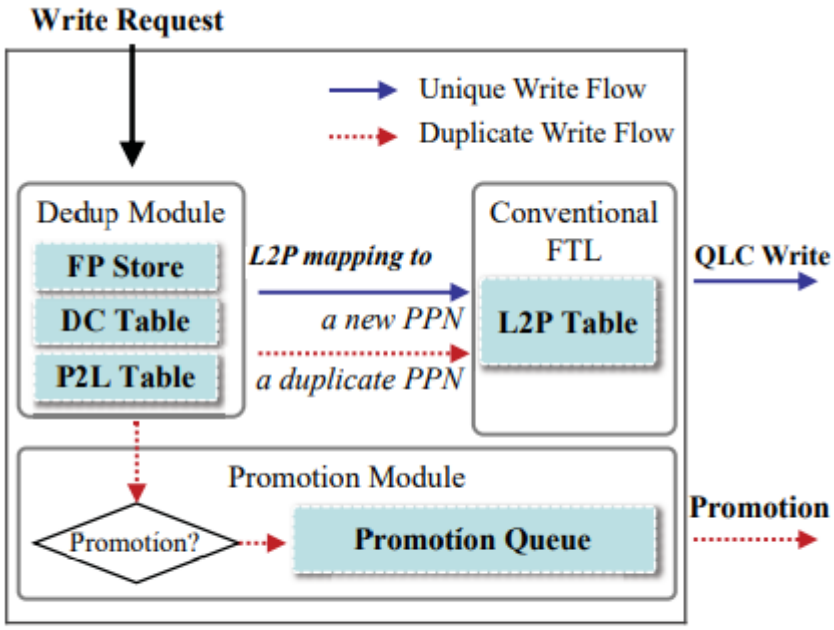

Figure 1: Overview of DFS FTL

DFS는 GC로 진행되는 강등 메커니즘을 통해 중복제거를 중단으로 인한 일시적인 유효용량 감소를 해결한다.

dedup모듈의 연산은 inline 중복제거 절차와 비슷하다.

dedup모듈은 3가지 주요 자료구조를 비교한다.

fingerprint(FP) 저장소는 저장된 페이지의 해시값을 저장한ㅏ.

쓰기요청이 발생하면 dedup모듈은 페이지의 해시값을 비교하여 FP저장소에 계산된 해시값이 존재하는지 아닌지 결정한다.

그러고 나서 쓰기요청이 중복인지 판단한다.

중복제거로 인해 여러개의 논리 페이지들이 물리적 페이지에 저장될 때, P2L매핑 테이블은 물리적 페이지를 가리키는 논리페이지 주소를 저장한다.

P2L매핑 테이블은 또한 GC또는 웨어레벨링으로 인해 복제된 페이지를 포함하는 물리적 페이지의 위치가 변경될 때 수정될 L2P 테이블의 관련 항목을 식별하는데 사용된다.

그러므로 이 데이블은 프로모션과 강등 연산을 수행하는데 있어 필수적이다.(왜?)

마지막으로, 중복 횟수 테이블(DC)은 물리적 페이지에서 중복제거된 논리적 페이지의 수를 저장한다.

DC 테이블은 프로모션과 디모션을 위한 빠른 플래시 모드 결정에서 사용된다.

쓰기 요청에서 DFS는 쓰기 페이지 중복 여부에 따라 다르게 동작한다.

만약 쓰기 페이지의 FP가 FP저장소에 있는 어떤 FP와도 매칭돼지 않는다면 페이지는 기존 쓰기 경로에 따라 QLC 블록의 빈 페이지에 쓰여진다.

그리고 LPN의 L2P 매핑은 그에따라 업데이트 된다.

또한 해당 페이지의 해시값은 FP 저장소에 저장된다.

그러나, 쓰기페이지의 해시값을 FP저장소에서 찾는 경우, 물리페이지 할당과 플래시 쓰기가 생략된다.

그런 다음 쓰기 요청 대상 논리 페이지에 해당하는 L2P 항목이 FP저장소에 있는 물리적 페이지에 매핑된다.

마지막으로, 쓰기요청에서 논리 페이지의 번호는 해당 물리 페이지의 P2L테이블 엔트리에 추가된다. 그리고 물리 페이지의 중복 횟수가 1씩증가한다.

SSD가 idle 상태일때 프로모션 모듈은 중복제거된 페이지를 승격시킨다. 왜냐하면 전체 물리페이지에서 승격후보를 찾는것은 높은 비용을 요구하기 때문이다. 따라서 프로모션 모듈은 중복제거가 발생할 때 물리페이지의 중복횟수를 추적하면서 현재 페이지의 플래시 모드가 페이지 중복횟수가 변경되었을 때도 적합한지 확인한다.

만약 물리 페이지의 플래시 모드가 중복 횟수에 비해 낮은경우, 해당 페이지는 프로모션 큐로 들어간다. 프로모션 큐의 물리페이지들의 승격은 SSD가 idle상태일 경우 수행된다. 이는 호스트의 읽기와 쓰기요청을 방해하지 않기때문이다.

Deduplication

페이지들의 해시값(SHA-1) FP는 FP저장소에 저장된다. (그림2)

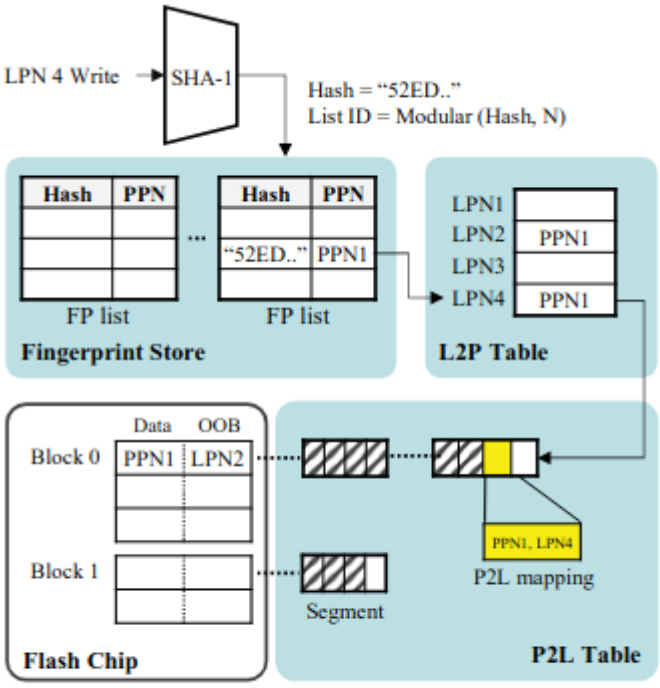

Figure 2: Data structure manipulation for the inline deduplication of a write operation

FP저장소는 DRAM에 위치한다. 따라서 dedup모듈은 write요청을 처리하는 동안 빠르게 존재하는 FP를 탐색할 수 있다.

각 FP 저장소의 엔트리는 FP와 PPN의 쌍을 들고있다. 만약 FP저장소의 전체 엔트리들을 하나의 리스트로 관리한다면 매칭되는 FP엔트리의 탐색시간이 전체 엔트리 개수에 따라 증가할 것이다.

그러므로 탐색 오버헤드를 줄이기 위해 FP저장소는 N개의 리스트로 나뉜다.

FP f를 가진 페이지 엔트리는 f%N번째 리스트에 들어간다.

만약 새로운 페이지의 FP를 f_new라고하면, f_new%N번째 리스트에서 FP들을 선형탐색한다.

N이 클 수록 검색오버헤드는 작다. 그러나 N이 증가하면 리스트 헤더를 저장하기 위한 메모리 소모량이 증가는 문제가 있다. 그러므로 N은 FP저장소의 크기와 사용가능한 메모리를 고려하여 적절하게 결정해야 한다. 본 연구에서는 256으로 설정하였다.

페이지 쓰기로 인해 새로운FP가 생성되면, FP와 PPN 쌍은 FP저장소에 새로운 엔트리로 삽입된다. 각 FP 리스트는 링크드 리스트로 관리된다. 그리고 FP 탐색동안 맞는 FP를 찾을 경우 해당 엔트리는 리스트의 헤드로 옮겨진다. 그러므로 가장 최근에 중복제거된 FP가 리스트 헤드에 위치한다. 그리고 가장 일치하지 않은 FP가 맨 뒤에 위치한다.

전체적으로 이러한 방법으로 FP를 관리하는 것은 특히 자주 중복제거되는 페이지에서 탐색 시간을 줄인다.

FP저장소는 사전에 정의된 엔트리 개수만 관리한다. 엔트리 개수가 초과할 경우 FP저장소에서 가장 오래된 엔트리를 삭제한다.(LRU) 이전연구들은

128K FP 엔트리로 만족스러운 중복제거 성능을 달성했다.

128K FP엔트리는 대략 3.5MB를 차지한다. 이러한 작은 크기로 인해 FTL L2P 매핑 플래시에 영구적으로 저장될 수 있다.

중복제거가 쓰기요청동안 수행될 때 DFS는 먼저 해당 L2P 매핑 엔트리를 업데이트한다. L2P매핑은 FTL의 기존 L2P 테이블에서LPN 엔트리(항목)에서의 중복 페이지의 PPN을 저장함으로써 쉽게 완료된다.

L2P매핑 업데이트가 완료된 이후, FTL은 호스트에게 쓰기요청이 완료됨을 알린다. 그리고 P2L테이블에 새로운 P2L항목 추가를 시작한다.

P2L매핑 항목(엔트리)는 PPN과 LPN의 쌍이다. 그리고 이러한 쌍은 중복제거가 발생할때마다 생성된다.

P2L 테이블은 반드시 중복제거된 페이지에서의 모든 P2L정보를 저장해야하기 때문에 상대적으로 큰 메모리가 요구된다.

그러무로 어떤 중복제거 FTL은 SSD의 NVRAM 영역을 사용하여 P2L매핑 정보를 저장한다.

DFS도한 P2L매핑 테이블을 NVRAM에 저장한다. P2L 항목을 저장하기 위한 공간은 세그먼트 단위호 NVRAM에 할당된다.

각 세그먼트는 1KB의 크기를 가진다. 이는 각 P2L항목이 8바이트(PPN 4바이트, LPN 4바이트)가지기 때문이다. 따라서 128 P2L 항목이 세그먼트에 저장된다. P2L 항목들은 Log-structured 방식으로 세그먼트에 순차적으로 저장된다.

GC 또는 웨어레벨링 연산은 물리적 페이지의 위치를바꾼다.

이때 물리적 페이지를 탐조하는 LPN의 모든 L2P항목을 업데이트해야한다.

왜냐하면 GC 웨어레벨링은 블록단위로 수행되는데 P2L매핑항목을 저장하는 세그먼트가 각 블록단위로 할당되기 때문이다. (그림2)

그리고 블록에 속한 페이지들의 P2L 매핑항목은 블록에 할당된 세그먼트에 기록된다.

P2L 항목이 논리페이지의 중복제거로 생성되므로 P2L매핑 테이블의 크기는 대략 중복제거된 스토리지 용량의1/512가 된다.

DC 테이블은 모든 PPN의 중복횟수를 저장한다. 각 항목은 오직 1바이트의 크기만 차지하기 때문에 1TB SSD에서는 256MB가 DC테이블로 요구된다. 이것또한 NVRAM에 저장된다.

1바이트 카운터는 다양한 작업의 중복특성을 고려해도 충분히 크기 때문에. 이를 초과할 경우 더이상 기록되지 않는다.

쓰기절차의 마지막 단계에서 FTL은 L2P매핑이 변경되는 PPN의 중복횟수를 수정한다. 페이지를 업데이트하면 중복횟수가 감소하지만, 반면에 중복제거는 페이지의 중복횟수를 증가시킨다(?)

중복횟수가 현재 플래시 모드보다 빠른 플래시 모드를 사용하기에 충분히 높을 때 PPN은 승격 큐에 삽입된다.

중복쓰기는 L2P P2L DC 테이블을 하나씩 업데이트한다. 그러므로 이러한 메타데이터 구조는 업데이트 도중 크래시가 발생할때에도 무결성을 갖추어야 한다.

앞서 언급한 것 처럼 쓰기요청은 L2P매핑이 업데이트 될 때 호스트에 의해 지속적으로 인식됩니다.

DFS에서 L2P의 무결성은 기존 FTL과 동일한 방법으로 보장된다.

L2P테이블의 무결성 보장에 기반하여 크래시가 발생했을 때 복구절차로 DFS는 NVRAM을 초기화하고 L2P테이블 항목을 읽어들여 P2L과 DC테이블을 재구성한다.

Promotion and Demotion

앞서 서술한 대로 승격은 호스트 요청을 방해하는 것을 피하기 위해 idle 한 경우 프로모션 모듈에 의해 수행된다.

idle시간에, 프로모션 모듈은 프로모션 큐에서 가장 오래된 PPN을 디큐한다. 프로모션 큐의 정책은 FIFO을 따른다. PPN이 프로모션 큐에 있는 동안 PPN의 DC는 증가하거나 감소할 수 있다.

그러므로 프로모션 큐로부터 PPN을 디큐한 후 프로모션 모듈은 대략적인 플래시모드를 결정학 위해 현재 PPN의 DC를 재확인한다.

만약 중복제거가 짧은시간에 자주 발생하면 QLC 물리적 페이지가 pSLC모드로 써지는 2단계 승격도 가능하다.

반대로 PPN의 DC값이 큐에 있는동안 감소하면 프로모션 모듈은 PPN에 대해 어떤 작업도 수행하지 않고 큐의 다음 PPN으로 이동한다.

DFS는 지원되는 각 플래시 모드에 대해 열린 블록을 유지한다.

특정 플래시 모드로 쓰여질 페이지는 해당 오픈 블록에서 할당된다.

예를들어 SLC free 페이지는 SCL 오픈블록으부터 할당된다. 그리고 만약 SLC오픈블록에 free페이지가 없을경우 이전 GC로 인해 생성된 free블록 중 하나를 SLC 오픈블록으로 할당한다.

PPN 승격을 위한 플래모드가 결정되었다고 가정했을 때, 기존 PPN은 해당 플래시모드의 free페이지로 복사된다. 복사 완료되면 해당 PPN이 속한블록의 P2L세그먼트는 PPN괘 매핑되는 LPN을 탐색한다. FTL의 L2P테이블에서는 해당 LPN 매핑은 더 빠른 플래시 모드로 작성된 PPN으로 업데이트된다.

P2L 매핑 항목또한 PPN 블록의 매핑 세그먼트로 이동한다.

그 후, 기존 물리 페이지는 무효화되고(invalid) P2L매핑 항목은 삭제된다.

마지막으로 PPN의 중복제거 횟수는 오리지널 PPN의 값으로 설정된다. 드리고 오리지널 PPN의 중복제거 횟수는 리셋된다. 이렇게 승격이 종료된다.

페이지의 프로모션이 완료되면 프로모션 모듈은 호스트 리퀘스트나 GC job이 대기중인지 확인한다. 만약 처리해야할 작업이 없으면 프로모션 큐에서 다음 페이지의 승격작업이 수행된다.이러한 행동은 큐 내 페이지가 없을 때 까지 반복된다.

빠른 플래시모드에서 저장된 중복제거된 페이지에서 쓰기가 발생할 경우

추가적인 물리페이지가 사용된다. 그러므로 SSD의 명시적 용량을 보장하기 위해 빠른 플래시모드에서 적절한 고밀도 페이지로의 강등이 수행되어야 한다.

단순하게 고밀도 페이지로 복사하는 것은 효과적인 용량 회수에 있어서 도움되지 않는다. 이는 빠른 플래시 모드의 페이지가 invalidate되더라도 복사된 페이지가 여전이 공간을 차지하고 있기때문이다.

그러므로 강등은 invalid page가 지워지고 free page로 바뀌는 GC동안 수행되어야 한다.

일반적으로 지워질 victim블록을 선택할 때 GC는 가장 invalid page가 많은 블록을 선택한다. 이는 유효 페이지를 복사하는 횟수를 줄여서 용량 확보를 극대화하기 위해서이다.

그러나 DFS에서는 Table1에 표시된 것 처럼 심지어 valid 페이지에서도 강등을 통해 용량 회수가 발생 할 수 있다.

Table 1: Number of reclaimable pages from GC of a page depending on its current flash mode (row) and target flash mode (column).

Flash Mode	SLC	MLC	QLC
SLC	0	2	3
MLC	-2	0	1
QLC	-3	-1	0

거꾸로, 추가적인 용량소모가 승격에서 발생할수도 있다.

그러므로 단순하게 invalid 페이지 수 기반으로 블록을 선택하는 대신,

GC를 통해 최대용량을 생성할 수 잇는 블록을 victim블록으로 선택한다.

victim블록이 선택되면 FTL은 victim 블록 내 각 페이지의 중복횟수를 확인하고 각 페이지에 대해 적절한 플래시 모드를 결정한다. 그러고나서 valid페이지를 결정된 플래시 모드의 빈 페이지로 복사한다.

그 후, FTL은 오리지널 페이지들을 무효화한다. 그리고 victim block의 모든 페이지가 무효화 되었을 때 블록 지우기연산을 수행한다.

옮겨진 페이지들의 L2P매핑과 P2L매핑은 그 후 변경된 PPN을 따라 업데이트된다. 그리고 GC가 완료된다.

Evaluation

본 연구에서 제안하는 기법을 평가하기 위해 SSDsim으로 구현하였다. SSDsim은 SSD의 내부 병렬화를 시뮬레이션 할 수 있다. 이는 상용SSD에서 사용하는 사실상 표준 아키텍쳐를 나타낸다. DFS 구현을 위해 본 연구는 free플래시 블럭을 다른 프로그래밍 모드로 변환할 수 있도록 SSDsim을 수정했다.

이는 SSDsim이 읽거나 쓸 페이지의 프로그래밍 모드에 따라 다중 타이밍 파라미터를 사용하도록 요구한다.

Table 2는 프로그래밍 모드에서 각 읽기, 쓰기, 지우기의 타이밍 파라미터를 나타낸다.

Table 2: Parameters of flash memory used in evaluation [21, 34].

	SLC	MLC	QLC
Read Time	25 μs	50 μs	100 μs
Write Time	300 μs	900 μs	1500 μs
Erase Time	1.8 ms	3 ms	6 ms

DFS가 inline 중복제거를 수행한다는 점을 감안할 때, 쓰기경로의 중복제거 작업은 쓰기 대기 시간을 연장할 수 있다.

DFS의 inline 중복제거 매커니즘은 Remap-SSD와 비슷하다. 그러므로 본 연구는 inline 중복제거의 컴퓨팅 오버헤드를 Remap-SSD분석값과 동일하다고 가정했다.

DFS의 FP 탐색이 기존 Remap-SSD보다 최적화 된 것을 가정할 때, 이러한 가정은 DFS에 불리하게 작용한다.

읽기와 쓰기지연은 시뮬레이터 내부에서 측정되었다. 그러므로 호스트 측 대기시간을 제외하고 요청을 처리하기 위해 스토리지 측 경과시간을 다룬다.

우리는 DFS의 실험결과를 오리지널 SSDsim, DFS 매커니즘으로 FP 탐색이 최적화된 Remap-SSD를 비교했다. 모든 경우에서 Table2의 파라미터가 사용되었다.

스토리지 디바이스의 중복제거 효과를 시뮬레이션 하기위해 시뮬레이션을 위한 I/O 워크로트를 트레이스는 접근된 블록의 내용 또는 적어도 해시값과 해당 블록 번호가 포함된다.

그러나 대부분의 I/O 트레이스는 블록 넘버의 엑세스 시퀀스만 포함하고 있다.

그러므로 해당 트레이스로는 중복제거 기법을 평가할 수 없다.

왜냐하면 중복된 페이지는 해당 워크로드에서 유일 페이지들과 구별될 수 없기 때문이다.

그러므로 우리는 FIU I/O 트레이스라는 가상의 합성 워크로드를 사용했다. 이는 접근된 블록의 해시값을 포함한다.

Analysis with Hypothetical Workloads

다양한 상황에서 제안기법의 효과와 행동특성을 평가하기 위해서는 시뮬레이션에 사용되는 I/O 트레이스를 제어하는 것은 필수적이다. 그러므로 본 연구는 가상 워크로드가 다양한 특성을 가지도록 합성하고 이를 DFS로 평가했다.

본 연구는 가상 워크로드를 생성하기 위한 몇가지 파라미터를 결정했다. 이는 유일 데이터의 주기를 결정한다.

데이터셋에 대한 엑세스는 높은 수준의 지역성을 같은 것으로 알려져 있다. 그러므로 Zipf's law가 스토리지 컨텐츠 분산에 많이 사용된다. 그러므로 본 연구에서는 Zipf분산을 가상 워크로드의 유니크 데이터 주기로 결정했다.

먼저 우리는 1GB의 순차쓰기를 수행한다. Zipf 파라미터는 중복데이터의 양과 중복 수를 제어하기 위해 변경되었다. M 컨텐츠 집합에서 n번재 인기 컨텐츠의 확률은 아래와 같이 정의된다.

$$P(n) = C\frac{1}{n^s} \text{ ,where } C = \frac{1}{\sum_{n=1}^{M} 1/n^s}$$

파라미터 s가 존재하는 Zipf 분포에서 가장 인기있는 컨텐츠의 확률은 두번째와 세번째로 인기있는 컨텐츠 확률의 2^s, 3^s 이다.