멀티스트림 SSD 상에서 상이한 데이터 수명을 갖는 스트림 조합에 관한 연구

최소이 이상원
성균관대학교 소프트웨어학과

{ithdli swlee}@skku.edu

On Combining Data Streams with Different Life Times on Multi-Streamed SSD

So-Yee Choi Sang-Won Lee
SungKyunKwan University

요 약

플래시 메모리 SSD는 빠른 성능을 기반으로 데이터 센터 컴퓨팅 환경에서 차세대 주류 저장장치로 자리 잡아가고 있다. 하지만 SSD의 구조적인 특징으로 인하여 SSD의 내부에서 불가피하게 호스트에서 요청한 것보다 많은 쓰기 작업이 수행된다. 이는 오랜 시간동안 SSD을 사용함에 따라 성능저하가 발생하는 원인이 된다. 이를 해결하기 위해 데이터의 수명에 따라 데이터의 물리적 위치를 구분하는 멀티스트림 SSD 기법이 등장하였다. 본 논문에서는 멀티스트림 SSD 인터페이스와 NoSQL 데이터 베이스 엔진 데이터의 수명에 대한 패턴 분석을 통하여 멀티스트림 SSD 상에서 상이한 데이터 수명을 갖는 스트림들의 조합에 대해 연구하였다. 본 논문에서는 Couchbase사의 NoSQL 저장엔진인 ForestDB를 사용해 다양한 스트림 조합에 따라 YCBS 벤치마크를 수행하였으며, 쓰기 증폭 지수(WAF)를 기준으로 성능을 평가하였다. 성능 평가 결과 다양한 조합 방식에 따라 쓰기 증폭 지수(WAF)를 최대 27%정도 개선할 수 있었다.

1. 서 론

플래시메모리 Solid State Drive(SSD)는 빠른 접근 속도를 기반으로 새로운 주류 저장 장치로 자리 잡고 있다. 플래시 메모리 SSD는 풍부한 내부 컴퓨팅 자원을 제공해 단순한 읽기/쓰기 명령 외에 새로운 인터페이스 지원이 가능하다.

최신 SSD 인터페이스 기술의 핵심은 SSD의 구조적 특성으로 발생하는 Flash Translation Layer (FTL)의 Garbage Collection(GC) 병목을 줄이는 것이다. 일례로, 상이한 데이터 수명을 지닌 응용을 위해 복수 개의 쓰기 스트림을 지원하는 MultiStream SSD [2] (이하 MS-SSD)가 제안되었다. 하지만, 데이터 집약적 응용에서 MS-SSD 인터페이스의 효과적인 사용에 관한 연구는 아직 초기 단계이다.

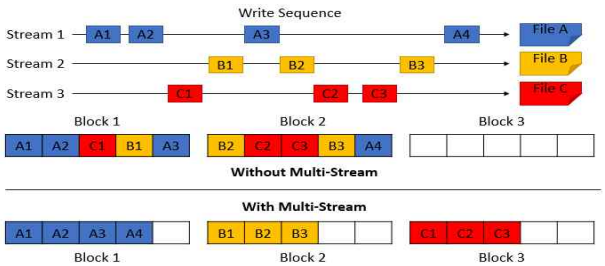
이러한 연구의 일환으로, 본 논문에서는 응용의 논리적 데이터 스트림과 MS-SSD의 물리적 스트림 사이의 조합 최적화 문제를 정의하고, 실험 결과를 제시, 분석하였다. 실험에서는 NoSQL 저장엔진 ForestDB를 이용해 상용 MS-SSD상에서 YCSB 벤치마크를 수행했다. 이 때, ForestDB 엔진의 네 가지 논리적 스트림과 MS-SSD 물리적 스트림 사이의 조합을 바꾸면서 논리적 데이터 스트림의 분류에 따른 성능 개선을 확인했다. 실험 결과, 쓰기 증폭 지수(WAF) 기준으로 조합 방식에 따라 최대 27% 성능 개선을 확인했다.

2. 멀티스트림 SSD (MS-SSD)

2-1. SSD의 특성

SSD는 기존의 주 저장장치인 하드디스크와 다르게 전기적인 신호로 작동한다. 이 때, SSD는 데이터 간섭이 발생하지 않도록 데이터를 쓰기 전에 해당 영역을 비우는 과정(Garbage Collection, 이하 GC)이 필수적이다. 일반적으로 GC의 성능이 SSD 성능의 병목이 된다. 빈 블록을 획득하기 위해서 페이지의 집합인 블록 단위의 삭제가 이루어지는데, 페이지 단위로 이뤄지는 읽기/쓰기와의 단위 차이로 삭제 하는 블록에 유효한 페이지와 유효하지 않은 페이지가 섞여있을 수 있기 때문이다. 즉, 유효한 페이지가 삭제되지 않도록 복사하는 카피백(Copyback)이 불가피하게 발생한다. 카피백은 SSD 내부에서 호스트에서 요청한 쓰기 외의 불가피한 쓰기 작업을 추가로 필요로 하기 때문에 GC 병목의 원인이 된다. 즉, 카피백의 양을 나타내는 WAF가 작을수록 SSD가 좋은 성능을 보인다. [1]

2-2. 멀티스트림 SSD 인터페이스 및 특징



[그림 1]. 멀티스트림 SSD 디자인

MS-SSD는 다양한 데이터 수명을 지닌 데이터를 *스트림*으로 나누어 물리적으로 구분하여 위치시킴으로써 GC 병목을 해소하고자 한다. [3][4]

*스트림*은 SSD 물리적 공간의 구분이다. 호스트 시스템과 SSD는 스트림 아이디를 공유하며, 아이디에 따라 데

이터의 위치를 지정한다. 같은 스트림의 데이터는 메모리상에서 동일한 블록에 써지며 다른 스트림의 데이터와 분리된다. 호스트 시스템에서는 데이터를 쓸 때, 해당 데이터에 대한 스트림 아이디를 제공 하며, SSD는 스트림에 대한 결정권 없이 호스트에서 제공한 아이디를 기반으로 데이터를 위치시킨다.

MS-SSD는 복수 개의 쓰기 스트림을 지원한다. MS-SSD는 동일한 생명주기의 데이터가 동일 블록에 위치함으로써 한 블록 내의 페이지들이 비슷한 시점에 수명이 다하도록 하여 GC병목의 최소화를 목표로 한다. MS-SSD 기술은 기술위원회 T10[8]에서 표준화 되었다.

본 논문에서는 삼성의 PM953 MS-SSD를 사용하였다. 본 논문에서는 실험을 위해 ForestDB의 쓰기 작업 시에 *fadvise*[2] 시스템 콜을 통해 스트림을 구분하였다. 이 때, 호스트에서 스트림을 데이터의 수명에 맞게 효율적으로 구분하는 것 역시 중요하다. 해당 문제에 대해서는 3장에서 언급하겠다.

3. ForestDB

3-1 ForestDB의 특징

ForestDB는 Couchbase사에서 개발한 Key-Value 데이터베이스이다. 기존의 B+ Tree 기반의 데이터베이스와 달린 Hirarchical B+ Tree를 사용하여 높은 읽기/쓰기 성능을 보인다. 또한 ForestDB는 기존의 in-place update방식을 사용하는 데이터베이스와 다르게 Append 방식으로 데이터를 쓰거나 업데이트한다.[5]

ForestDB는 여러 개의 데이터베이스 파일들로 구성되어 있다. 데이터의 타입은 크게 데이터베이스 헤더, 슈퍼 블록, 인덱스 노드, 데이터 페이지의 4가지 종류가 있다.

본 논문에서는 성능 측정을 위해 ForestDB에서 제공하는 ForestDB-Benchmark[7]를 사용하였다. 워크로드는 덮어쓰기 워크로드로 데이터베이스의 임의의 키를 선택해 해당하는 문서의 값을 업데이트하는 작업을 수행한다.

3-2 ForestDB 데이터의 분류 및 데이터 수명 분석

MS-SSD를 활용한 ForestDB 저장엔진의 최적화를 위해 ForestDB-Benchmark 에서 쓰기 작업을 수행할 때마다 스트림 아이디를 파일별, 데이터 타입별로 두 가지 방식으로 부여하였다.

[표 1] 데이터 타입별 수명 및 쓰기 비율

	인덱스 노드	헤더	데이터 페이지	슈퍼 블록
평균 수명(회)	848531	764571	1457289	752
쓰기 비율(%)	1.8	10.6	76.96	10.6

우선, 파일별로 스트림을 구별한 결과 MS-SSD의 효과가 거의 없었다. 이는 MS-SSD의 목적은 서로 다른 수명을 갖는 데이터의 구분을 통해 GC 오버헤드를 줄이는 것인데 데이터 파일별로 수명이 다르지 않기 때문이다.

다음으로 데이터 타입별 스트림 부여 방안에 대해 논하겠다. ForestDB 저장엔진은 4가지 데이터 타입을 사용한다. 이를 위해 ForestDB엔진을 수행하는 과정에서 각각의 데이터 타입별 블록 접근 양상을 분석하였다. [표1]은 forestDB의 사이즈를 약7.5GB로 설정하여 덮어쓰기 워크로드에 따라 4시간 동안 실험을 수행하여, 데이터 타입별로 평균 수명과 쓰기 비율을 나타낸다. 헤더, 인덱스 노드, 데이터 페이지는 동일한 LBA 영역을 공유하며 쓰기가 이뤄졌고 슈퍼블록은 별도의 영역에 쓰기가 이루어졌다. 먼저 슈퍼블록은 128개의 LBA (512KB)영역에 걸쳐 쓰기가 이루어졌으며 전체 쓰기 비율의 10%를 차지하였다. 각각의 LBA에 대해서 다음 데이터가 다시 써질 때까지의 수명은 평균적으로 752이다. 즉, 752번의 write 명령 이후에 데이터의 업데이트가 이루어졌다. 이는 다른 데이터 타입에 비해 현저히 작은 횟수로 슈퍼블록이 수명이 아주 짧은 *Hot* 함을 의미한다. 헤더, 인덱스 노드, 데이터 페이지는 총 1913960개의 LBA영역(7.3GB)을 공유한다. 각각의 타입별로 특정 LBA에서의 데이터 수명을 분석한 결과 인덱스 노드와 헤더는 각각 평균 848531, 764571로 비슷한 양상을 보였다. 반면 데이터 페이지의 수명은 평균 1457289로 인덱스 노트와 헤더보다 약 2배 정도 수명이 긴 *Cold* 데이터임을 확인하였다.

본 논문에서는 이러한 수명 분석 결과에 기반 해 4 가지, 1)스트림 비부여 2)데이터 타입별 스트림 부여 3)유사 수명을 보이는 인덱스 노드와 헤더를 같은 스트림 부여 4)상이한 수명을 보이는 슈퍼블록과 데이터 페이지를 하나의 스트림으로 부여, 방식으로 실험을 진행하였다.

4. 성능평가

4-1 성능 평가 환경 및 벤치마크 설정

[표 2] 실험 환경

운영체제	Ubuntu 14.04.5 LTS
파일시스템	Ext4 file system
SSD	Samsung PM953 MultiStream nvme SSD, 480GB
성능 평가 툴	ForestDB-Benchmark

본 논문의 실험 환경은 [표2]와 같다. ForestDB는 Append방식으로 데이터를 쓰다가 특정 비율이상 유효하지 않은 데이터가 생기면 업데이트 되어 유효하지 않은 데이터 영역을 재사용한다. 벤치마크 툴을 통해 임의의 키를 선택하여 해당하는 문서 값을 업데이트하는 워크로드를 사용했으며, 워크로드는 100% 쓰기로 구성하였다. 데이터베이스의 크기는 초기화를 포함하여 400GB이다. 데이터베이스 파일은 32개로 벤치마크 툴을 통해 32개의 데이터베이스 파일에 동시에 접근했다. 벤치마크 툴의 로그로 총 write양을 확인할 수 있으며, WAF 값의 경우 SSD의 smart 값인 *Host Write Command* 값과 *Data Units Written* 값을 SSD로부터 요청하여 얻은 후 계산하였다.

4-2 성능 평가 결과

[표 3] 스트림의 분류 여부에 따른 write양 비교

	총 쓰기 양(GB)
스트림 비구분	4203.41
타입 별 구분	4556.53

[표3]는 스트림 구분을 하지 않은 경우와 호스트에서 데이터의 타입별로 디폴트 포함 5개의 스트림으로 구분하여 24시간 동안 Forest-Benchmark를 수행했을 때 총 쓰

기 양을 보여준다. 스트림을 구분하였을 때 353GB의 데이터가 더 써졌다. 이는 MS-SSD을 수행했을 때 쓰기 연산에 대한 처리 성능이 더 좋아졌기 때문이다.

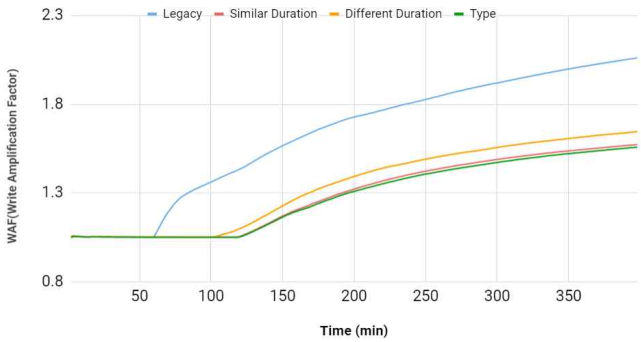
[표 4] 스트림의 분류 여부에 따른 WAF 비교

	동일 시간(24hr)	동일 양 (4203.4GB)
비 구분	2.399	2.399
타입 별 구분	1.749	1.737
성능 개선(%)	27.0946	27.5948

[표4]은 동일 시간 동안 쓰기 작업을 했을 때와 동일한 양을 썼을 때 MS-SSD의 사용 여부와 데이터의 수명 구분에 따른 WAF 값을 비교 한 표이다. [표4]의 마지막 행의 성능 개선은 스트림 구분을 하지 않았을 때의 WAF를 A, 켰을 때를 B라고 했을 때 '((A-B)/A)*100' 로 WAF의 감소량을 의미한다. 동일 시간(24시간) 후 27.09%의 두드러진 성능 개선을 보였다. 이 때 WAF는 데이터의 쓰기양이 많을수록 증가하는데 [표3]에 따르면 동일 시간동안 실험을 수행했을 때 타입별로 스트림을 구별 시 쓰기가 더 많이 이뤄졌다. 따라서 WAF값 감소량을 동일한 양의 쓰기에 대해 비교하였다. 동일한 양(4203.4GB)를 썼을 때 27.59%정도의 성능 개선을 보여, 동일 시간 비교 시보다 0.5% 정도 더 감소량을 보였다.

[표 5] 스트림 분류 방식에 따른 WAF 비교

스트림 구분	동일 시간 (6.5hr)	
	WAF	감소량 (%)
비 구분	2.011	-
타입 별 구분	1.508	25.01
동일 수명 통합	1.523	24.26
상이 수명 통합	1.595	20.69



[그림 2] 시간에 따른 WAF 변화

[표5]는 스트림 구별 방식에 따른 WAF값을 비교한 표이다. [그림2]는 시간에 따른 WAF의 변화를 나타낸 그래프로 x축은 시간(분), y축은 WAF를 의미한다. 두 자료는 ForestDB의 데이터를 모두 같은 데이터 타입별로 스트림을 구분한 경우, 비슷한 수명의 타입(헤더와 인덱스 노드)을 묶고 데이터 페이지와 슈퍼블록을 구분한 경우, 서로 다른 수명의 타입(데이터 페이지와 슈퍼블록)을 묶고 헤더와 인덱스를 구분한 경우, 또 MS-SSD을 사용하지 않은 경우의 4가지 경우에 대해, 각각 6시간 30분 동안의 실험을 통해 얻은 WAF 값의 양상을 보여준다.

자료에 따르면 스트림 구분을 하지 않은 경우, 즉, MS-SSD를 사용하지 않은 경우가 항상 가장 높은 WAF값을 보였다. 스트림을 구분한 경우에 대해서 살펴보면 타입별로 4개의 스트림으로 나누었을 때(Type)와 비슷한 수명 패턴으로 스트림을 나누었을 때(Similar Duration), WAF값 변화량이 비슷하게 나타났다. 즉, 비슷한 수명을 가지는 데이터는 스트림을 구분하는 것과 그렇지 않은 것의 차이가 크지 않음을 확인하였다. 한편 상이한 수명 패턴을 보이는 데이터 타입으로 스트림을 조합했을 때(Different Duration) 스트림 구분을 하지 않았을 때보다 WAF값이 20% 감소하며 월등한 성능이 개선을 보이기는 했으나 다른 경우에 비해서는 큰 WAF값을 보였다. 다만 무시할 수 없는 성능 개선을 보여 하나의 타입이라도 분류했을 때에 성능 개선을 보일 수 있음을 확인하였다.

5. 결론

본 논문에서는 ForestDB의 논리적 데이터 스트림과 MS-SSD의 물리적 스트림 사이의 조합에 따른 MS-SSD에서의 성능을 측정했다.

실험 결과, 유사한 수명의 데이터는 구별을 하는 것과 하지 않은 것의 차이가 두드러지지 않은 반면 동일하지 않은 수명의 데이터 타입을 묶었을 때는 상대적으로 적은 성능을 보였다. 이를 통해 논리적 데이터의 수명에 따른 스트림의 올바른 구분이 중요함을 확인하였다.

다만 수명의 차이가 크게 나는 두 데이터를 묶었음에도 타입별로 구분했을 때와 WAF가 크게 차이나지 않는 점은 추후에 수명에 따른 데이터 타입의 다양한 조합을 통한 실험을 통해 추가적인 연구가 필요함을 시사한다.

더불어, 항상 스트림을 구분한 경우 스트림을 구분하지 않았을 때보다 성능이 좋아짐을 확인하여 SSD 상에서의 NoSQL엔진의 최적화 가능성 또한 확인하였다.

사사

이 논문은 2017년도 삼성전자(클러스터과제)의 지원을 받아 수행된 연구임.

본 연구는 미래창조과학부 및 정보통신기술진흥센터의 SW컴퓨팅산업원천기술개발사업(SW스타랩)의 연구결과로 수행되었음" (IITP-2015-0-00314)

참고 문헌

[1] Peter Desnoyers, "Analytic modeling of SSD write performance", 2012

[2] Jeong-Uk Kang, Jeeseouk Hyun, Hyunjoo Maeng, Sangyeun Cho, "Multi-streamed Solid-State Drive", 2014

[3] Fei Yang, Kun Dou, Siyu Chen, Mengwei How, Jeong-UK Kang, Sangyeun Cho, "Optimizing NoSQL DB on Flash: A Case Study of RocksDB", 2015

[4] Jun He, SudarsunKannan, Andrea C.Arpaci-Dusseau, Remzi H.Arpaci-Dusseau, "The Unwritten Contract of Solid State Drives", 2017

[5] Mijin Ahn, Gihwan Oh, Sang-Won Lee, "SSD and File System Performance Tests on NoSQL Workload" 2015

[6] ForestDB, https://github.com/couchbase/forestdb

[7] ForesDB-Benchmark, https://github.com/couchbaselabs/ForestDB-Benchmark

[8] T10, http://www.t10.org/