올 플래시 스토리지를 위한 플래시 캐시 확장 기법

안미진O, 오기환, 이상원
성균관대학교
{meeeejin, wurikiji, swlee}@skku.edu

Flash-Based Extended Cache for All-Flash Storage

Mijin Ahn, Gihwan Oh, Sang-Won Lee
Sungkyunkwan University

요 약

플래시메모리 저장장치와 하드디스크의 용량당 가격 차이 때문에, 하드디스크를 SSD와 같은 플래시메모리 저장장치로 완전히 교체하는 것보다 추가하여 사용하는 것이 경제적이라는 맥락에서 제안된 기법이 Flash-aware Cache Extension(FaCE)이다. FaCE는 낮은 오버헤드로 플래시 메모리를 DRAM 버퍼의 확장으로 사용하는 캐싱 전략이다. 하지만 많은 기업들이 SSD를 기본 저장장치로 사용함으로써, HDD를 저장장치로 사용하는 기존의 FaCE 기법은 활용되기 힘들다. 본 논문에서는 기존 저장장치를 상대적으로 저렴한 상용 SSD로, 플래시 캐시를 고성능, 고비용의 NVMe SSD로 사용함으로써 FaCE 기법의 취지를 유지하면서 FaCE가 어떠한 성능을 보이는지 분석하였다. 이를 통해 FaCE 기법이 SSD 기반의 시스템에서도 활용될 수 있음을 보였다.

1. 서 론

플래시 메모리 SSD는 비-휘발(Non-Volatile) 메모리 저장장치로, 전기 신호로 움직이기 때문에 매우 빠른 읽기/쓰기 성능을 보여준다. 지난 몇 년 동안 SSD 관련 기술이 빠르게 발전하면서, 대용량 데이터를 처리하는 다양한 기업에서 SSD를 기본 저장장치로 사용하고 있다. 하지만 SSD의 단위 용량당 가격은 HDD의 단위 용량당 가격보다 여전히 비싸다. 따라서 경제적 요인을 고려하여 HDD를 SSD로 완전히 교체하는 것보다 HDD에 SSD를 추가하여 성능을 향상시키는 것이 더 합리적이다.

이러한 맥락으로 SSD를 DRAM 버퍼와 HDD 사이의 캐시로 사용하는 Flash-aware Cache Extension(FaCE) [1] 기법이 등장했다. FaCE 기법에서는 어떤 페이지가 DRAM 버퍼에서 교체될 때, 그 페이지를 데이터 저장장치인 HDD가 아닌 플래시 메모리 SSD 캐시에 저장한다. 이후 그 페이지에 대한 접근이 발생하면, 느린 HDD 대신 플래시 캐시에서 페이지를 읽어옴으로써 전체 IO 성능을 향상시킨다. 즉, FaCE의 목적은 소량의 플래시 메모리를 추가로 사용해 큰 성능 향상을 이루는 것이다.

FaCE를 제안한 논문[1]에서는 오픈 소스 DBMS인 PostgreSQL에 FaCE 기법을 구현해 성능 향상을 증명했다. 관련 선행 연구로는 FaCE 기법을 상용 DBMS인 알티베이스 HDB에 구현하여 FaCE 기법이 성능 향상에 크게 기여하는 것을 보여준 연구[2]와 FaCE 기법을 MySQL에 구현해 마찬가지로 성능 향상을 보여준 연구[3]가 있다.

FaCE와 관련된 기존의 연구들은 저장장치를 HDD로 사용한 연구들이다. 하지만 모든 저장장치를 SSD로 교체하려는 올-플래시 스토리지(All-Flash Storage) 시대가 도래하면서, 많은 기업들이 기본 저장장치를 SSD로 교체하고 있다. 또한, FaCE 논문이 등장했을 때보다 SSD의 용량당 가격은 훨씬 떨어졌기 때문에, 기본 저장장치를 상대적으로 저렴한 상용 SSD로 교체하는 것은 가격 면에서도 경제적이다.

본 논문에서는 FaCE 기법을 오픈 소스이자 범용 DBMS인 MySQL에 구현하였다. 이 때, 데이터 저장장치를 상용 SSD로 설정하고, 플래시 캐시는 NVMe SSD와 같은 고성능 SSD로 설정하였다. 또한 FaCE 기법을 사용한 경우에는, MySQL에서 원자 쓰기를 보장하기 위해 사용하는 InnoDB의 이중 쓰기 버퍼(DoubleWrite Buffer)를 사용하지 않기 때문에 중복된 쓰기 연산도 없앴다. 그 결과, 위의 상황에서 FaCE 기법을 적용한 MySQL이 어떠한 성능을 보이는지 분석하였다.

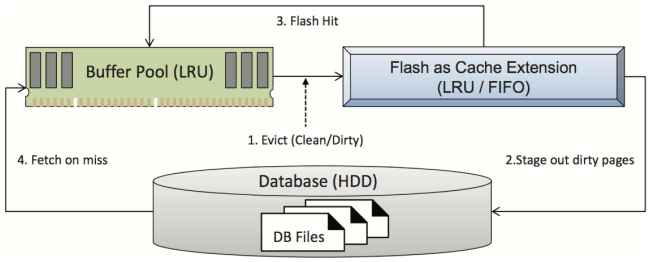
본 논문의 구성은 다음과 같다. 먼저 2장에서는 플래시 메모리 SSD의 특성을 활용한 FaCE에 대해 소개한다. 3장에서는 본 논문에서 수행한 성능 평가 환경과 그 결과를 보인다. 마지막으로 4장에서 결론을 맺음으로써 본 논문을 마무리한다.

2. 플래시 메모리와 FaCE

플래시 메모리 SSD는 전기 신호로 움직이기 때문에 HDD에 비해 높은 IO 성능을 보여준다. 그리고 순차 쓰기가 임의 쓰기보다 적게는 수배, 많게는 수십 배 가까이 빠르다[4].

기존의 버퍼 관리 정책에서는 버퍼 공간을 효율적으로 사용하고 버퍼 적중률(hit ratio)을 높이기 위해 LRU(Least Recently Used) 교체 정책을 사용한다.

하지만 이 교체 정책은 SSD에 임의 쓰기 패턴을 유발한다. FaCE 기법에서는, SSD의 경우 임의 쓰기보다 순차 쓰기의 성능이 더 좋다는 특징을 활용하기 위해, LRU 교체 정책 대신 FIFO 정책을 사용한다. 적은 양의 임의 쓰기를 큰 순차 쓰기로 변환함으로써, FaCE 기법은 SSD의 병렬성도 활용할 수 있고, 결과적으로 큰 성능 향상을 이룰 수 있다.



[그림 1] FaCE 기법의 기본 구조

FaCE의 기본 구조는 [그림 1]과 같다. FaCE에서는 DRAM 버퍼에서 특정 페이지가 교체될 때, FIFO 교체 정책을 따라 플래시 캐시에 순차적으로 쓴다. 이 때, 교체 대상이 되는 페이지가 수정된 페이지인 경우, 즉 더티 페이지(dirty page)인 경우에만 플래시 캐시에 쓴다. 클린 페이지(clean page)인 경우에는 위의 과정을 생략하고 바로 해당 프레임을 비운다. 그리고 플래시 캐시가 꽉 찼을 경우, FIFO 교체 정책에 따라 가장 오래 전에 쓰인 페이지부터 저장장치로 교체한다.

읽기 작업이 발생할 때, 즉 페이지에 대한 접근이 발생할 때, 해당 페이지가 DRAM 버퍼에 존재하지 않으면 FaCE에서는 해당 페이지를 플래시 캐시에서 먼저 찾는다. 플래시 캐시에 존재하는 경우, 플래시 캐시에서 바로 읽어오고, 그렇지 않은 경우에만 저장장치에서 읽어온다. 플래시 캐시에 존재하는 페이지에 대한 접근이 발생한 경우, 느린 저장장치가 아니라 플래시 캐시에서 읽기 작업을 수행하기 때문에 IO 성능을 크게 향상시킬 수 있다.

또한 FaCE는 플래시 메모리의 비휘발성도 활용한다. 일단 데이터 페이지가 DRAM 버퍼에서 쫓겨난 후 플래시 캐시에 쓰이면, 데이터베이스가 의도치 않게 종료된 경우에도 플래시 캐시 자체가 비휘발성이기 때문에 데이터 페이지의 복구가 가능하다. 또한, 복구 과정도 HDD와 같은 느린 저장장치가 아니라, 플래시 캐시로 사용되는 빠른 SSD에서 진행되므로 복구 시간도 단축 가능하다.

3. 성능 평가

본 논문에서는 FaCE 기법을 MySQL에 구현하였다. 그리고 기본 저장장치를 상대적으로 저렴한 상용 SSD로 설정하였고, 플래시 캐시로 사용되는 SSD는 고성능, 고비용 SSD인 NVMe SSD로 설정하였다. 이러한 환경에서 FaCE 기법을 적용한 MySQL이 어떠한 성능을 보이는 지 확인하였다.

본 논문에서 데이터 저장장치로 사용한 SSD는 Samsung 850 Pro(512GB)로, SSD 4개로 RAID-0를 구성해 실험을 수행했다. 그리고 플래시 캐시로 사용한 SSD는 고성능 SSD인 Intel P3700 NVMe SSD (800GB)이다. 자세한 실험 환경은 [표 1]과 같다.

[표 1] 실험 환경

운영체제	Ubuntu 14.04.3 LTS
프로세서	Intel® Xeon® CPU E5-2670 v3 @ 2.30GHz
메모리(RAM)	64GB
저장장치	Samsung 850 Pro SSD, Intel P3700 NVMe SSD
데이터베이스	MySQL 5.6.26
성능 평가 시 사용한 툴	SysBench

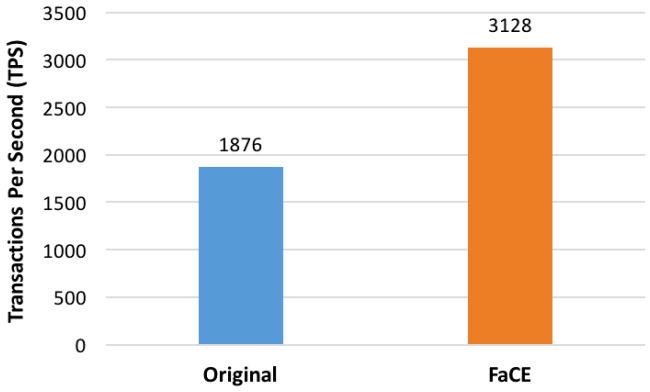
데이터베이스의 크기는 약 200GB이고, 플래시 캐시의 크기는 전체 데이터 크기의 20%인 40GB로 설정하였다. DRAM 버퍼의 크기는 4G로 설정하였고, 데이터 페이지의 크기는 16KB이다. 실험에 사용한 벤치마크 툴은 SysBench이다. 이 때, 데이터베이스에 동시에 접근하는 클라이언트의 개수는 16개로 설정하였고, 운영체제의 간섭을 제거하고 실험하기 위해 Direct IO 플래그를 켠 상태로 실험을 수행하였다.

그리고 FaCE가 구현된 MySQL의 경우, DRAM에서 교체된 페이지를 순차적으로 플래시 캐시에 기록하기 때문에 플래시 캐시에는 해당 페이지의 예전 복사본도 존재한다. 즉, 플래시 캐시 안에 같은 페이지에 대한 여러 버전의 복사본이 존재할 수 있다. 이러한 관점에서, FaCE를 적용한 MySQL의 경우에는 이중 쓰기 버퍼 없이도 복구가 가능하므로 이중 쓰기 버퍼를 끄고 실험을 진행하였다.

본 논문에서 성능 평가 대상이 되는 데이터베이스는 2가지로, 첫 번째는 소스 코드를 수정하지 않은 원본 MySQL 데이터베이스(Original)이다. 두 번째는 FaCE 기법을 적용한 MySQL 데이터베이스이다. 두 데이터베이스 모두 데이터 저장장치는 RAID-0로 묶은 상용 SSD이다. 그리고 FaCE의 플래시 캐시로는 앞서 언급한 고성능 SSD를 사용하였다. 이에 따른 실험 결과는 [그림 2]와 같다.

성능 측정 결과, 기본 저장장치를 SSD로 사용하고 플래시 캐시를 고성능, 고비용 NVMe SSD로 사용한 경우에도 FaCE 기법은 상당한 성능 향상을 보였다. [그림 2]에서 알 수 있듯이, 원본 MySQL 데이터베이스는 SysBench 결과 초당 1876개의 트랜잭션을 처리했다. 반면, FaCE를 적용한 MySQL의

경우, 같은 실험 환경에서 초당 3128개의 트랜잭션을 처리했다. 즉, FaCE를 적용한 MySQL이 원본 MySQL보다 초당 약 70% 더 많은 트랜잭션을 처리했다.



[그림 2] SysBench 성능 측정 결과

결과적으로, FaCE는 상용 SSD로 구성된 시스템에 총 데이터 크기의 20%에 해당하는 소량의 고성능, 고비용 NVMe SSD를 추가하는 것만으로 70%의 성능 향상을 보였다.

성능 향상의 원인으로는, 크게 두 가지가 있다. 첫 번째로, FaCE의 경우 순차 쓰기가 임의 쓰기보다 빠르다는 SSD의 성능을 잘 활용하기 때문에 기존 MySQL보다 더 좋은 성능을 보였다. 여러 번의 적은 양의 임의 쓰기를 한 번의 큰 순차 쓰기로 변환하기 때문에 IO 성능을 높일 수 있었다. 두 번째로, 플래시 캐시 자체가 비휘발성이기 때문에 MySQL의 이중 쓰기 버퍼를 사용하지 않아도 된다는 점이 성능 향상에 영향을 끼쳤다. 의도치 않게 데이터베이스가 종료되는 경우, 페이지의 일부만 쓰여진 상태인 부분 쓰기(partial write) 문제가 발생한다. 이를 해결하고자 MySQL에서는 DRAM 버퍼에서 페이지가 쫓겨났을 때, 이중 쓰기 버퍼에 해당 페이지를 먼저 쓰고, 이중 쓰기 버퍼에 페이지가 완전히 쓰인 이후에 데이터 파일에 해당 페이지를 쓴다. 즉, 기존 MySQL은 이중 쓰기 버퍼를 이용해 같은 페이지에 대한 쓰기 작업을 두 번 함으로써 부분 쓰기 문제를 해결하였다. 하지만 FaCE의 경우 플래시 캐시가 비휘발성이기 때문에 플래시 캐시에 페이지가 쓰이기만 하면 의도치 않은 종료에도 복구가 가능하다. 플래시 캐시에 페이지를 쓰다가 종료된 경우에도, 수정되지 않은 원본 페이지는 기존 저장장치에 보존되어 있으므로 복구가 가능하다. 따라서, FaCE는 MySQL의 이중 쓰기 버퍼를 사용하지 않아도 복구가 가능하므로, 이중 쓰기 버퍼를 사용하지 않음으로써 중복된 쓰기 연산을 없애 위와 같은 성능 향상을 이루었다.

4. 결론

본 논문에서는 MySQL에 FaCE 기법을 구현하고, 이러한 MySQL을 데이터 저장장치로는 상용 SSD, 플래시 캐시로는 고성능 SSD를 사용함으로써 어떠한 성능 향상을 보이는 지 측정하였다.

성능 측정 결과, SSD로 구성된 환경에서도 FaCE가 구현된 MySQL은 약 70%에 가까운 성능 향상을 보여주었다. 앞서 말했듯이, 상용 SSD의 가격이 많이 하락했지만 고성능 SSD의 가격은 여전히 높으므로, 본 연구와 같이 고성능 SSD를 소량 추가해 성능을 향상시키는 것이 더 경제적이다. 그리고 본 연구의 결과에서 알 수 있듯이, 전체 시스템을 고성능 SSD로 교체할 필요 없이 소량 추가함으로써, 합리적인 가격으로 큰 성능 향상을 이룰 수 있다.

사사

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

참고문헌

[1] Woon-Hak Kang, Sang-Won Lee and Bongki Moon, “Flash-based Extended Cache for Higher Throughput and Faster Recovery”, Proceedings of the VLDB Endowment (PVLDB), Vol. 5, No. 11, July 2012

[2] 구슬기, 강운학, 이상원, 문봉기, “알티베이스 DBMS 의 복구 가능한 플래시 캐시”, 정보과학회 논문지 데이터베이스, Vol. 40(1), 2013 년 2 월

[3] 안미진, 오기환, 강운학, 이상원, “MySQL을 위한 플래시 캐시 확장 기법”, 한국정보과학회 2016년 한국컴퓨터종합학술대회 논문집, p. 1899-1901, 2016년 6월

[4] Sang-Won Lee, Bongki Moon and Chanik Park, “Advances in Flash Memory SSD Technology for Enterprise Database Applications”, ACM SIGMOD, June 2009