MySQL을 위한 플래시 캐시 확장 기법

안미진, 오기환, 이상원 강운학
{meeeejin, wurikiji, swlee}@skku.edu woonhak.kang@gatech.edu

Flash Memory as Cache Extension for MySQL

Mijin Ahn, Gihwan Oh, Sang-Won Lee Woon-Hak Kang
Sungkyunkwan University Georgia Institute of Technology

요 약

플래시메모리 저장장치의 용량당 가격은 크게 감소했지만, 하드디스크와 플래시메모리 저장장치 사이에는 여전히 큰 가격 차이가 존재한다. 그러므로 하드디스크를 플래시메모리 저장장치로 완전히 대체하는 것보다 하드디스크의 성능을 보완하기 위해 플래시메모리 저장장치를 추가하여 사용하는 것이 더 경제적이다. 이러한 맥락으로 제안된 Flash-aware Cache Extension(FaCE)은 낮은 오버헤드로 플래시메모리를 DRAM 버퍼의 확장으로 사용하는 캐싱 전략이다. FaCE는 FIFO 알고리즘을 플래시 캐시 관리 정책으로 사용하여 임의 쓰기를 순차 쓰기로 변환하는데, 이는 플래시메모리의 쓰기 측면에서 매우 효율적이다. 본 논문에서는 MySQL에서 데이터베이스의 원자 쓰기를 보장하기 위해 사용하는 InnoDB의 이중쓰기(doublewrite) 버퍼를 사용하지 않음으로써, 중복된 쓰기 연산을 없애고 성능 향상과 원자 쓰기를 모두 보장하는 FaCE의 확장을 제안한다. 이를 통해 FaCE를 적용한 MySQL이 기존의 MySQL에 비해 어떠한 성능을 보이는지 분석하였다.

1. 서 론

플래시 메모리는 비-휘발(Non-Volatile) 메모리 저장장치이다. 그 중 NAND 플래시메모리 기반의 SSD(Solid State Drive)는 전기 신호로 움직이기 때문에 기계적 장치로 움직이는 HDD(Hard Disk Drive)에 비해 매우 빠른 읽기/쓰기 성능을 보여주고, 이에 따라 IO 처리량을 크게 증가시킨다. 지난 몇 년 동안 SSD 관련 기술이 빠르게 발전하면서, 대용량 데이터를 처리하는 업체를 포함한 다양한 분야에서 SSD가 기존 저장장치인 HDD의 대안이 되고 있다. 하지만 SSD의 단위 용량당 가격은 HDD의 단위 용량당 가격보다 여전히 훨씬 비싸다. 그러므로 대용량 데이터를 다루는 어플리케이션의 경우 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. 다른 캐싱 기법과 마찬가지로, FaCE의 목적은 소량의 자원을 투자해 큰 성능 향상을 제공하는 것이다. 이를 위해 FaCE에서는 어떤 페이지가 DRAM 버퍼에서 교체될 때 그 페이지를 HDD가 아닌 플래시메모리 저장장치인 SSD, 즉 플래시 캐시에 저장한다. 이후 그 페이지에 대한 접근이 다시 발생했을 때, 느린 HDD 대신 플래시 캐시에서 페이지를 읽어옴으로써 시스템 전체의 IO 성능을 높였다. 또한 FaCE에서는 시스템 동작 중 장애가 일어나 DBMS 복구가 필요한 상황에서 플래시 캐시를 DBMS를 복구하는 데에 사용함으로써 DBMS 복구 시간을 크게 단축시켰다.

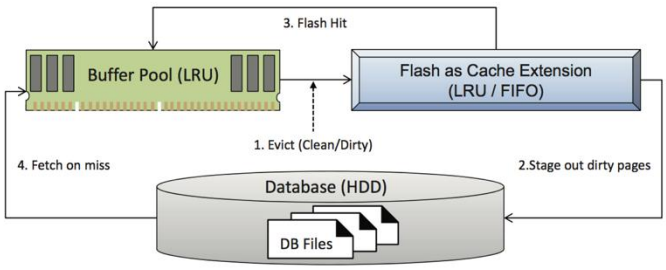
FaCE를 제안한 논문[1]에서는 오픈소스 DBMS인 PostgreSQL에 FaCE 기법을 구현해, 플래시 캐시를 사용하면 높은 성능 향상을 이룰 수 있음을 보였다. 뿐만 아니라 플래시 캐시를 이용해 DBMS 복구 시간을 크게 단축시킬 수 있음을 보였다. 관련 선행연구로는 FaCE 기법을 상용 DBMS인 알티베이스 HDB에 구현하여 FaCE 기법이 성능 향상에 크게 기여하는 것을 보여준 연구[2]가 있다.

본 논문에서는 FaCE 기법을 범용 DBMS이자 오픈소스 DBMS인 MySQL에 구현하였다. 이 때, MySQL에서 원자 쓰기를 보장하기 위해 사용하는 InnoDB의 이중쓰기 버퍼를 사용하지 않고 FaCE 기법을 통해 원자 쓰기를 보장함으로써, 중복된 쓰기 연산을 없앴다. 그 결과, FaCE 기법을 적용한 MySQL이 기존의 MySQL과 비교했을 때 어떠한 성능을 보이는 지 분석하였다.

본 논문의 구성은 다음과 같다. 먼저 2장에서는 플래시메모리 기반 SSD의 특성을 간단히 알아보고, 이를 활용하여 만든 FaCE에 대해 소개한다. 3장에서는 본 논문에서 수행한 성능 평가 환경과 그 결과를 보인다. 마지막으로 4장에서는 결론 및 향후 연구를 소개하고 논문을 마무리한다.

2. 플래시메모리 SSD와 FaCE

플래시메모리 SSD는 기계적 부품으로 구동되는 HDD와 달리 메모리 소자들로 이루어져 있다. 따라서 HDD와 달리 지연시간이 존재하지 않는다. 또한 SSD는 일반적으로 다수의 플래시메모리 칩으로 구성되며, 채널이라 불리는 단위로 동시에 접근이 가능하다. 따라서 HDD에 비해 높은 IO 동시성을 제공한다. 그리고 SSD는 순차 쓰기가 임의 쓰기보다 적게는 수배, 많게는 수십 배 가까이 빠르다[3].



[그림 1] FaCE의 기본 구조

기존의 버퍼 관리 정책은 버퍼 공간을 효율적으로 사용하고 버퍼 적중률(hit ratio)을 높이기 위해 LRU(Least Recently Used) 교체 정책을 사용한다. 하지만 이 정책은 SSD에 임의 쓰기 패턴을 유발하는데, 앞서 말했듯이 SSD는 임의 쓰기에 비해 순차 쓰기가 훨씬 빠르다. 이러한 특성을 잘 활용하기 위해, 플래시메모리 SSD 캐시 관리 정책을 기존의 LRU에서 FIFO(First-In First-Out) 방식으로 변경한 것이 FaCE 기법이다.

FaCE의 기본 구조는 [그림 1]과 같다[1]. FaCE에서는 DRAM 버퍼에서 특정 페이지가 교체될 때, 플래시 캐시에 항상 순차적으로 쓰여진다. 이를 위해 캐시를 원형 큐 형태로 유지하면서, 가장 오래전에 쓰여진 페이지부터 순차적으로 덮어쓴다. 이 때 플래시 캐시에서 교체될 페이지가 HDD에서 읽혀진 이후에 수정된 경우(즉, dirty 페이지인 경우)에만 그 페이지를 HDD에 쓴다. 그리고 페이지에 대한 접근이 발생할 때, 느린 HDD가 아니라 플래시 캐시에서 페이지를 읽어오기 때문에 IO 성능을 크게 향상시킬 수 있다.

하지만 이러한 방식으로 캐싱을 하면, SSD 캐시에 같은 페이지에 대한 여러 복사본이 존재하기 때문에 캐시 공간의 활용도는 낮아지고 따라서 캐시 적중률도 떨어진다. 그러나 쓰기 처리량의 관점에서 보면, FaCE 방식이 기존의 방식보다 최대 3배 이상의 성능 향상을 보인다. 또한 FaCE 방식을 사용하면 DBMS 복구가 필요할 때, 비휘발성인 플래시 캐시를 이용해 복구를 진행함으로써 복구 시간도 크게 단축시킬 수 있다.

본 논문에서는 이러한 FaCE 기법을 MySQL에 구현함으로써 MySQL의 경우엔 실제로 어떠한 성능을 보이는 지 확인하였다.

3. 성능 평가

본 논문에서 플래시 캐시와 데이터 저장장치로 사용한 SSD는 Samsung 840 Pro(256GB)이다. 그리고 HDD를 데이터 저장장치로 사용할 땐 HDD 8개로 RAID-0를 구성해 실험을 수행했는데, 이 때 사용한 RAID 컨트롤러는 Intel RS2WG160이다. 그리고 각 HDD는 15k-RPM Seagate ST3146356SS(146.8GB)이다. 자세한 실험 환경은 [표 1]과 같다.

[표 1] 실험 환경

운영체제	Ubuntu 14.04.3 LTS
프로세서	Intel® Core™ i7-4770 CPU @ 3.40GHz
메모리(RAM)	8.00GB
저장장치	Samsung 840 Pro SSD, Seagate ST3146356SS
데이터베이스	MySQL 5.6.26
성능 평가 시 사용한 툴	tpcc-mysql

데이터베이스의 크기는 약 10GB로 이는 TPC-C 웨어하우스 100개에 해당한다. 플래시 캐시의 크기는 전체 데이터 크기의 20%인 2GB로 설정하였고, DRAM 버퍼의 크기는 200MB로 설정하였다. 그리고 데이터 페이지의 크기는 16K이고, TPC-C 벤치마크의 데이터와 워크로드는 tpcc-mysql[4]을 통해 생성하였다. 이 때, 데이터베이스에 동시에 접근하는 클라이언트의 개수는 16개로 설정하였고, 운영체제의 간섭을 제거하고 실험을 하기 위해 Direct IO 플래그를 켠 상태로 실험을 수행하였다.

또한 FaCE를 적용한 MySQL의 경우에는 HDD의 오래된 데이터 페이지를 덮어 쓰지 않고 최신 복사본을 플래시 캐시에 기록하는데, 이는 MySQL InnoDB의 이중쓰기 버퍼의 동작 방식과 유사한 방식이다. 이러한 관점에서, FaCE를 적용한 MySQL의 경우에는 이중쓰기 버퍼 없이도 원자 쓰기를 보장하므로 이중쓰기 버퍼를 사용하지 않았다.

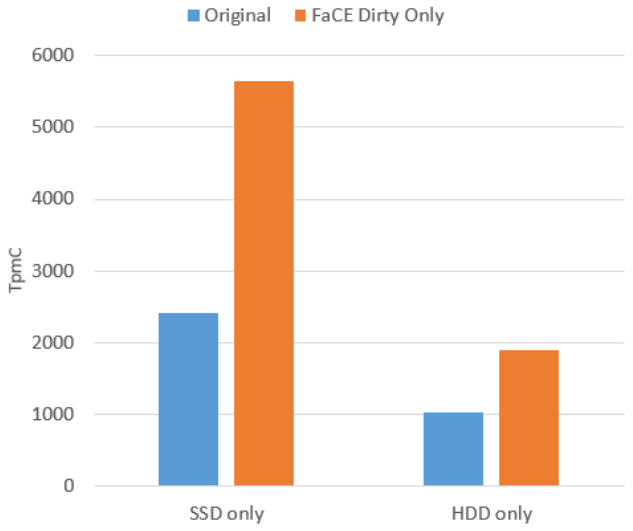
본 논문에서 성능 평가 대상이 되는 데이터베이스는 2가지로, 첫 번째는 소스 코드를 수정하지 않은 원본 MySQL 데이터베이스(Original)이다. 두 번째는 수정된 페이지, 즉 dirty 페이지에 대해서만 FaCE 기법을 적용한 MySQL 데이터베이스이다. 이 데이터베이스는 FaCE 기법을 제안한 논문[1]에서 정의한 대로, 캐싱 대상을 dirty 페이지로 한정하기 때문에 “FaCE Dirty Only”라 칭한다. 그리고 데이터를 저장하는 위치에 따라서도 2가지의 경우가 존재한다. 첫 번째는 모든 데이터를 SSD에 저장하는 경우(SSD only)이고, 두 번째는 모든 데이터를 HDD로 구성된 RAID-0에

저장하는 경우(HDD only)이다. 본 논문에서는 성능 평가 대상이 되는 데이터베이스와 데이터의 저장 장치를 달리해 총 4가지 실험을 수행했다. 이에 따른 실험 결과는 [표 2]와 같다.

[표 2] TPC-C 성능 측정 결과 (TpmC)

성능 평가 대상	데이터베이스 저장 위치	
	SSD only	HDD only
Original	2419	1029
FaCE Dirty Only	5646	1903

[표 2]를 그래프로 나타내면 [그림 2]와 같다.



[그림 2] TPC-C 성능 측정 결과 (TpmC)

성능 측정 결과, SSD only와 HDD only 두 경우 모두 FaCE Dirty Only를 적용한 MySQL이 더 높은 성능을 보였다. 먼저 SSD only의 경우엔, 원본 MySQL 데이터베이스가 2419 TpmC를 보여주는 반면 FaCE Dirty Only를 적용한 MySQL은 5646 TpmC를 보였다. 즉, FaCE Dirty Only를 적용한 MySQL이 원본 MySQL보다 약 2.3배의 성능 향상을 보였다. 다음으로 HDD only의 경우엔, 원본 MySQL 데이터베이스가 1029 TpmC를 보여주는 반면 FaCE Dirty Only를 적용한 MySQL은 1903 TpmC를 보였다. 즉, FaCE Dirty Only를 적용한 MySQL이 원본 MySQL보다 약 1.8배의 성능 향상을 보였다.

FaCE는 SSD만으로 구성된 시스템과 8개의 HDD로 구성된 RAID-0 시스템에 총 데이터 크기의 20%에 해당하는 소량의 SSD를 추가하는 것만으로도 각각 2.3배, 1.8배의 성능 향상을 보였다. 그리고 앞서 말했듯이, FaCE의 경우엔 플래시 캐시 자체가 비휘발성이고 같은 페이지에 대해서 여러 복사본이 플래시 캐시 내에 존재하므로 MySQL의 원자 쓰기를 보장하는 InnoDB의 이중쓰기 버퍼를 사용하지 않아도 된다. 즉 FaCE를 적용한 MySQL은 중복된 쓰기 연산을 없앴기 때문에 원본 MySQL보다 더 나은 성능을 보였다.

4. 결론

본 논문에서는 상용 DBMS 중 하나인 MySQL에 플래시메모리를 DRAM 버퍼의 확장으로 사용하는 캐싱 기법인 FaCE를 구현해 어떠한 성능 향상을 보이는지 측정하였다.

성능 측정 결과, 모든 데이터를 SSD에 저장하는 시스템의 경우엔 FaCE Dirty Only를 적용한 MySQL이 원본 MySQL보다 약 2.3배의 성능 향상을 보였다. 그리고 모든 데이터를 8-RAID로 구성된 HDD에 저장하는 시스템의 경우엔 FaCE Dirty Only를 적용한 MySQL이 원본 MySQL보다 약 1.8배의 성능 향상을 보였다. 즉 소량의 플래시메모리를 플래시메모리 캐시로 사용하고 InnoDB의 이중쓰기 버퍼를 사용하지 않음으로써, 적게는 1.8배, 많게는 2.3배까지의 성능 향상을 보였다.

향후 연구에서는 FaCE를 제안한 논문[1]에서 소개한 데이터베이스 복구와 관련된 부분을 MySQL에 구현할 계획이다. 즉, 플래시 캐시에 저장된 데이터 페이지를 이용하여 DBMS 복구(recovery) 작업 시 플래시메모리 캐시에 저장된 페이지의 정보까지 복원할 수 있도록 복구 모듈을 구현할 계획이다.

사사

이 논문은 2016년도 정부 (미래창조과학부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임 (R0126-16-1108, 비휘발성 메모리 기반 개방형 고성능 DBMS 개발)

본 연구는 2016년도 정부재원(미래창조과학부 여대학(원)생 공학 연구팀제 지원사업)으로 미래창조과학부, 한국연구재단과 한국여성과학 기술인지원센터의 지원을 받아 수행된 연구임

본 연구는 미래창조과학부 및 정보통신기술진흥센터의 SW중심대학지원사업의 연구결과로 수행되었음 (R2215-16-1005)

5. 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 DBMS의 복구 가능한 플래시 캐시”, 정보과학회 논문지 데이터베이스, Vol. 40(1), 2013년 2월

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

[4] tpcc-mysql, https://github.com/Percona-Lab/tpcc-mysql