버퍼 풀 크기에 따른 MySQL의 플러싱 패턴 분석

안미진○, 이상원
성균관대학교
{meeeejin, swlee}@skku.edu

Analyzing Flushing Patterns in MySQL by Varying the Size of Buffer Pool

Mijin Ahn, Sang-Won Lee
Sungkyunkwan University

요 약

버퍼 풀 또는 로그의 여유 공간이 부족할 때, 버퍼 관리자는 버퍼 교체 정책에 따라 버퍼의 페이지를 비우고 해제해야 한다. Dirty 페이지를 해제해야 하는 경우, 페이지를 비우기 전에 디스크에 쓰는 과정이 반드시 선행되어야 하기 때문에 이 과정은 매우 느리다. 이러한 상황을 피하기 위해, 대부분의 DBMS는 백그라운드 스레드를 사용해 주기적으로 dirty 페이지를 디스크에 플러시 하고, 사전에 여유 공간을 확보하려 한다. 본 논문에서는 MySQL이 SSD 상에서 구동될 때, 버퍼 풀 크기에 따라 플러싱 및 IO 패턴이 어떻게 변화하는지 파악하였다. 이를 통해, 버퍼 풀의 크기와 플러싱 패턴의 상관 관계를 분석하였다.

1. 서 론

대부분의 DBMS(database management system)는 테이블 및 인덱스 데이터를 메모리 상에 캐싱하기 위해, 버퍼 풀(buffer pool)이라 불리는 영역을 메모리에 유지하며, 버퍼 풀은 버퍼 관리자(buffer manager)에 의해 관리된다. 데이터 접근을 위한 디스크 I/O 작업은 메모리 상의 작업과 비교했을 때 상당한 시간이 걸리기 때문에, 한 번 디스크에서 읽은 데이터를 버퍼 풀에 캐싱함으로써, 추후 해당 데이터에 대한 접근을 빠르게 처리하고 DBMS의 전체적인 성능을 향상시킬 수 있다.

버퍼 풀이 가득 차 여유 공간이 없거나 트랜잭션 로그 영역이 부족할 때, 버퍼 관리자는 버퍼 교체 정책(예를 들어, LRU 정책)에 따라 쫓아낼 페이지를 선택하고 여유 공간을 만든다. 이 때, 쫓겨난 페이지가 수정 사항이 디스크에 반영되지 않은 dirty 페이지인 경우, 해당 페이지를 메모리에서 디스크로 써야 하고, 이 작업을 플러싱(flushing)이라 한다. 플러싱은 주로 백그라운드 작업으로 이루어지지만, 디스크 I/O를 수반하고 동기화 등의 문제로 유저 스레드에 영향을 끼치기 때문에 DBMS의 처리량에 상당한 영향을 끼친다. 그러나 플러싱 관련 설정을 수동으로 조정하는 것은 워크로드와 시스템 구성에 따라 크게 달라지기 때문에 적절하게 튜닝하기가 어렵다.

한편, 플래시 메모리 SSD는 비-휘발(non-volatile) 메모리 저장 장치로, 전기 신호로 움직이기 때문에 매우 빠른 읽기/쓰기 성능을 보여준다. 지난 몇 년간 플래시 메모리 관련 기술이 빠르게 발전하면서, 대용량 데이터를 처리하는 많은 기업들이 HDD 대신 SSD를 주 저장장치로 사용하고 있다[1].

본 논문에서는 오픈소스 데이터베이스인 MySQL이 SSD 상에서 구동될 때, 버퍼 풀 크기에 따라 플러싱 패턴이 어떠한 경향을 보이는지 분석하였다. OLTP 벤치마크인 TPC-C를 사용해 실험을 수행하였고, 버퍼 풀의 크기에 따라 MySQL의 플러싱 및 IO 패턴이 어떻게 변화하는지 파악하였다. 이를 통해, 버퍼 풀의 크기와 플러싱 패턴의 상관 관계를 분석하였다.

본 논문의 구성은 다음과 같다. 먼저 2장에서는 MySQL의 대표적인 플러싱 기법에 대해 소개한다. 3장에서는 본 논문에서 수행한 실험 환경과 그 결과를 보인다. 마지막으로 4장에서 결론을 맺음으로써 본 논문을 마무리한다.

2. MySQL/InnoDB의 플러싱 기법

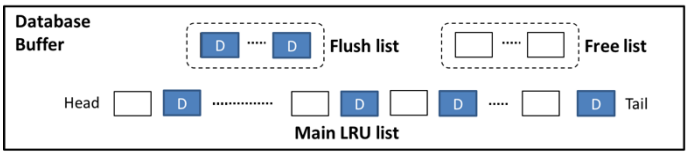

그림 1 InnoDB의 버퍼 풀에 존재하는 3가지 리스트

MySQL의 스토리지 엔진인 InnoDB는 데이터와 인덱스를 캐싱하기 위해 많은 데이터 페이지를 가지는 버퍼 풀을 메모리에 유지한다. 버퍼 풀의 크기가 클수록 메모리에 캐싱되는 데이터가 많아 지기 때문에, 쿼리 수행에 필요한 디스크 I/O 양을 줄일 수 있고, 이에 따라 DBMS의 전체적인 성능을 향상시킬 수 있다.

InnoDB는 3가지 리스트를 사용해 버퍼 풀의 페이지를 관리하는데, 이는 그림 1과 같고 그림에서 D는 dirty 페이지를 의미한다. 첫 번째로, free list는 사용 가능한 페이지 프레임, 즉 free 페이지를 모아 놓은 리스트이다. 두 번째로, LRU list는 버퍼 풀에 존재하는 모든 페이지를 저장한 리스트로, 가장 오랫동안 사용되지 않은(least recently used) 페이지를 tail에 유지한다. 마지막으로 flush list는 버퍼 풀에 존재하는 모든 dirty 페이지를 가지고 있는 리스트로, LSN 순으로 가장 오랫동안 수정되지 않은(least recently modified) 페이지를 head에 유지한다.

한편, 버퍼 풀의 페이지가 수정되었을 때, InnoDB는 수정된 dirty 페이지를 즉시 디스크에 쓰진 않는다. 대신, dirty 페이지들이 디스크에 쓰이기 전에 여러 번 수정되기를 바라면서, dirty 페이지를 메모리에 유지한다. 이를 쓰기 결합(write combining)[2]이라고 하며, InnoDB의 성능 최적화 방안 중 하나다. 하지만 InnoDB는 제한된 크기의 버퍼 풀을 가지기 때문에, 만약 InnoDB가 디스크에서 읽을 필요가 있는 페이지를 저장할 공간이 없다면, 공간을 확보하기 위해 dirty 페이지를 디스크에 쓰고 해제(free)해야 한다. 하지만 dirty 페이지를 해제하기 전에 디스크에 쓰는 과정이 반드시 선행되어야 하기 때문에 이 과정은 매우 느리다. 따라서, InnoDB는 page cleaner thread라 불리는 백그라운드 스레드를 사용해 주기적으로 dirty 페이지들을 디스크로 플러시 하고, 사전에 free 페이지를 확보하려고 한다. 이 스레드는 1초 주기로, 서버와 I/O 상태에 따라 하나의 플러싱만 수행한다. 본 논문에서는 InnoDB의 3가지 플러싱 방식 중 백그라운드 스레드가 수행하는 2가지 플러싱만 다룬다.

2.1 LRU list 플러싱

Page cleaner thread는 1초마다 깨어나 LRU tail부터 일정 범위 내의 모든 dirty 페이지를 디스크로 플러시 한다. 그 다음, 플러시 된 dirty 페이지들의 프레임을 비우고 free 페이지로 만들어 free list에 삽입한다. 이를 LRU list 플러싱이라 부른다. LRU list 플러싱의 탐색 범위의 기본값은 1,024로, 이는 플러시 할 dirty 페이지를 찾기 위해 LRU tail부터 1,024개의 페이지를 탐색한다는 의미이다.

2.2 Flush list 플러싱

버퍼 풀에 존재하는 dirty 페이지의 비율이 특정 값에 도달했을 때, page cleaner thread는 flush list를 사용해 플러싱을 수행한다. 이를 flush list 플러싱이라 하고, 이 기법은 flush list에서 가장 오래 전에 수정된 dirty 페이지를 플러시 대상으로 선택한다. 이 때, dirty 페이지 비율의 임계 값을 높은 워터마크(high-water mark)라 하고, 기본값은 75%이다.

또한, InnoDB는 flush list를 사용해 체크포인트를 수행한다. InnoDB의 트랜잭션 로그는 고정된 크기이며, 순환(circular) 방식으로 기록된다. 그러나 아직 플러시 되지 않은 dirty 페이지에 대한 변경 레코드를 포함한 경우, 해당 로그를 덮어 쓸 수 없다. 로그를 기록하는 활동이 한 바퀴를 돌아 로그의 tail에 부딪치게 되면, InnoDB가 로그의 일부 영역을 비우려고 서둘러서 일을 하는 동안, 서버는 정지해 있다. 이러한 상황을 피하기 위해, InnoDB는 fuzzy 체크포인트라고 불리는 과정을 수행한다. 먼저, flush list를 사용해 가장 오래된 dirty 페이지의 LSN(Log Sequence Number)을 찾는다. 가장 오래 전에 수정된 페이지는 로그에서 가장 뒤 쪽에 있으며, 가장 먼저 이 상황에 부딪히는 페이지이므로, 이 페이지의 LSN을 체크포인트의 낮은 워터마크(low-water mark)로 취급하고 작은 배치 단위로 플러싱 작업을 수행한다. 즉, InnoDB는 한 번에 모든 dirty 페이지를 플러시 하는 대신 fuzzy 체크포인트를 통해 작은 단위로 dirty 페이지를 플러시 하며, 가능한 한 트랜잭션 로그에 많은 공간을 확보하기 위해 낮은 워터마크를 높이려고 한다.

3. 실험

버퍼 풀의 크기에 따른 MySQL의 플러싱 경향을 분석하기 위해 본 논문에서는 아래의 표 1과 같은 환경에서 실험을 진행하였다. 데이터베이스의 크기는 약 10GB로, 이는 TPC-C 웨어하우스 100개에 해당한다. 클라이언트의 개수는 64개로 설정하였고, 버퍼 풀의 크기는 전체 데이터 크기의 15%, 30%, 45%, 60%, 75%로 바꾸어 가면서 실험을 진행하였다. 로그 데이터는 별도의 저장 장치에 따로 저장하였고, MySQL 서버를 위해 사용한 파일시스템은 ext4다. 그리고 운영체제의 간섭을 제거하고 실험을 하기 위해 Direct I/O 플래그를 켠 상태로 실험을 수행하였다.

버퍼 풀 크기에 따른 플러싱 비율을 측정한 결과는 그림 2와 같다. 실험 결과, 버퍼 풀의 크기가 15%일 때 플러싱 작업의 98%가 LRU list 플러싱이었다. 하지만 버퍼 풀의 크기를 증가시킬 수록 flush list 플러싱의 비율이 증가했다. 버퍼 풀의 크기가 75%일 때, LRU list 플러싱의 비율이 54%까지 감소한 반면, flush list 플러싱의 비율은 46%까지 증가하였다.

표 1 실험 환경

운영체제	Ubuntu 14.04.3 LTS
프로세서	Intel® Xeon® CPU E5-2670 v3 @ 2.30GHz
메모리(RAM)	64GB
데이터 저장장치	Samsung 960 PRO NVMe SSD
로그 저장장치	Samsung 850 PRO SSD
데이터베이스	MySQL 5.7.21
성능 평가 툴	tpcc-mysql[3]

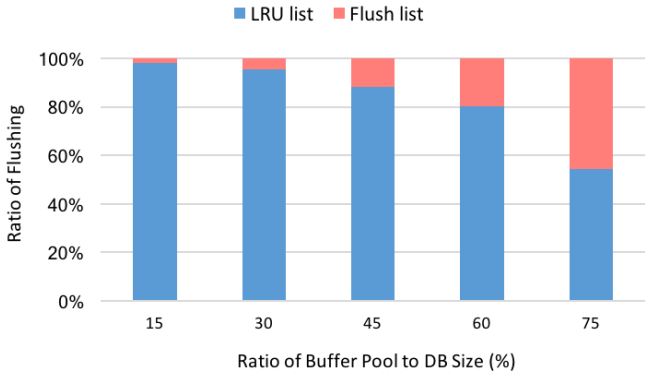

그림 2 버퍼 풀 크기에 따른 플러싱 비율 변화

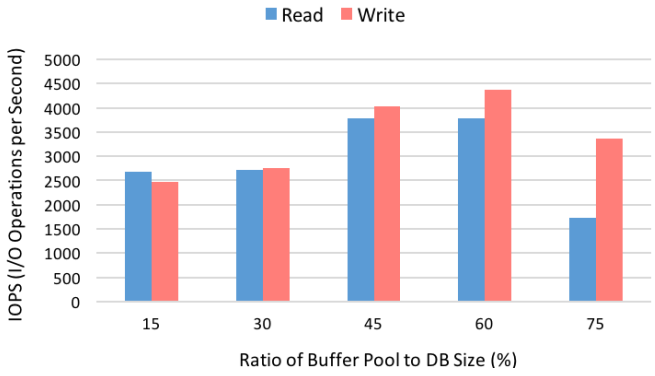

그림 3 버퍼 풀 크기에 따른 IOPS 변화

버퍼 풀 크기에 따라 플러싱 비율이 다른 이유는 버퍼 풀 크기가 커지면 버퍼에 캐싱될 수 있는 데이터의 양이 많아져 dirty 페이지의 비율이 75%에 도달할 확률이 높기 때문이다. 또한 버퍼 풀에 저장된 페이지가 디스크로 쫓겨날 확률도 상대적으로 낮기 때문에, 메모리 상에서 트랜잭션이 보다 활발히 처리되고 그에 따라 트랜잭션 로그도 보다 빠르게 쌓이게 된다. 앞서 설명했듯이, 트랜잭션 로그의 여유 공간이 부족해 서버 및 유저 스레드가 정지하는 상황을 피하기 위해, InnoDB는 보다 활발히 fuzzy 체크포인트를 수행한다. 이와 같은 이유로 버퍼 풀의 크기가 커질수록 flush list 플러싱의 비율이 높아진다.

다음으로, 그림 3은 버퍼 풀 크기에 따른 읽기 및 쓰기 IOPS 변화량을 측정한 결과이다. 그림에서 알 수 있듯이, 버퍼 풀의 크기가 15%일 때는 읽기 및 쓰기 IOPS가 각각 2683, 2470으로 비슷하게 측정되었다. 그러나 버퍼 풀의 크기를 75%까지 키웠을 때, 읽기 및 쓰기 IOPS는 각각 1725, 3363으로 쓰기 IOPS가 읽기 IOPS보다 약 2배 더 높았다.

버퍼 풀 크기를 키울 수록 쓰기 IOPS가 읽기 IOPS보다 상대적으로 더 증가하는 이유도 flush list 플러싱 비율이 증가한 것과 관련이 있다. 버퍼 풀이 커져 캐싱할 수 있는 데이터 페이지 개수가 늘어나면, 많은 읽기 작업을 디스크 접근 없이 메모리 상의 데이터로 처리할 수 있다. 반면, 트랜잭션 로그의 여유 공간을 확보하고 dirty 페이지의 비율을 높은 워터마크 값보다 낮추기 위해, 백그라운드 스레드는 flush list 플러싱을 활발하게 수행한다. 즉, 쓰기 작업은 지속적으로 활발히 이루어지지만 읽기 작업은 주로 메모리에서 처리되기 때문에, 버퍼 풀이 커지면 쓰기 IOPS가 읽기 IOPS보다 상대적으로 증가하게 된다.

표 2 버퍼 풀 크기에 따른 로그 저장장치의 IOPS

Ratio of Buffer Pool to DB size(%)	15	30	45	60	75
Write IOPS	6	8	21	32	43

표 2는 앞선 실험과 같은 조건으로 실험했을 때, 로그 저장장치의 쓰기 IOPS를 측정한 결과이다. 측정 결과, 버퍼 풀의 크기가 커질수록 트랜잭션 로그가 빠르게 생성되기 때문에 로그 저장장치로 플러시 되는 로그의 양도 실제로 증가했다.

4. 결론

본 논문에서는 버퍼 풀 크기에 따라 변화하는 MySQL의 플러싱 패턴을 확인하였다. 실험 결과, 버퍼 풀 크기가 커질 수록, dirty 페이지의 비율을 적정 수준으로 조정하고 트랜잭션 로그의 여유 공간을 확보하기 위해 flush list 플러싱의 비율이 늘어나고 쓰기 IOPS가 상당히 증가하는 것을 확인하였다. 이를 완화하기 위해, 로그의 크기를 키우는 것도 하나의 방안이 될 수 있지만, 이는 복구 시에 더 많은 시간을 필요로 한다. 따라서, 플러싱의 비율을 적절하게 튜닝하기 위해서는 데이터 및 로그 저장장치의 성능을 우선적으로 고려해야 하고, 워크로드의 특성에 맞는 버퍼 풀 크기를 사용해야 한다.

사사

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

이 성과는 2018년도 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임 (No. 2018R1A2B2005502).

이 논문은 2018년도 정부(과학기술정보통신부)의 재원으로 한국연구재단-차세대정보·컴퓨팅기술개발사업의 지원을 받아 수행된 연구임 (No. NRF-2015M3C4A7065696).

5. 참고문헌

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

[2] Intel, "Write Combining Memory Implementation Guidelines",http://download.intel.com/design/PentiumII/applnots/24442201.pdf, November 1998

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