MySQL Insert Buffer의 성능 평가 및 성능 향상 방안에 대한 연구

이황교, 오기환, 이상원
성균관대학교 컴퓨터공학과
hgyolee@gmail.com, wurikiji@skku.edu, swlee@skku.edu

Research for Performance Evaluation and Improvement Plan about MySQL Insert Buffer

Hwanggyo Lee, Gihwan Oh Sang-won Lee
Dept of Computer Engineering, Sungkyunkwan University

요 약

MySQL 데이터베이스에서 사용하는 Non-clustered Secondary Index는 디스크 접근 시 Random한 입출력을 유발하여 디스크 장치의 성능을 저하시키는 문제점을 가지고 있다. 이를 해결하기 위해 MySQL의 Storage Engine은 Insert Buffer를 사용하여 Random한 디스크 접근을 방지한다. Benchmark를 통해 성능 평가를 진행한 결과 Insert Buffer를 사용하는 것만으로 성능이 개선되는 것을 확인하였다. 또한, 현재의 Insert Buffer를 수정하여 더 큰 성능 향상을 이끌어낼 수 있는 부분을 발견하여 이에 대한 아이디어를 제시, 간단한 구현을 통해 추가적인 성능 향상을 보였다.

1. 서론

MySQL의 Storage Engine인 InnoDB에서 테이블은 하나의 Primary Index를 가지며, Non-clustered Secondary Index를 0개 또는 하나 이상 가질 수 있다. Index들은 모두 B-tree 구조를 가진다. 테이블에 레코드가 삽입(INSERT)되면, 해당 레코드는 먼저 Primary Index에 삽입된 후, Secondary Index에 삽입되게 된다. 이 때, Non-clustered Secondary Index로 인해 디스크 입출력이 Random하게 발생하게 된다. 삽입 외에 갱신(UPDATE), 삭제(DELETE) 연산의 경우도 비슷하게 Random한 입출력 패턴이 나타난다. Random한 입출력의 발생은 결과적으로 디스크 장치의 성능 하락을 가져온다. 이러한 Random 입출력으로 인한 성능 저하를 해결하고자 InnoDB에서는 Insert Buffer를 사용하고 있다.

2. Insert Buffer

Secondary Index에 INSERT, UPDATE, DELETE 연산이 요청되는 경우, InnoDB는 먼저 요청된 페이지가 메모리의 Buffer Pool 상에 존재하는지 확인한다. 만약 그 페이지가 Buffer Pool에 있는 경우에는 요청된 연산을 바로 적용시킨다. 하지만, Buffer Pool에 없는 경우에는 해당 페이지를 찾기 위해 디스크에 접근하게 되는데, Secondary Index의 경우 이 과정에서 Random한 입출력이 발생하게 된다. 이를 줄이고자 InnoDB는 Insert Buffer를 사용한다. InnoDB는 Buffer Pool 내의 일정 용량을 Insert Buffer를 위해 제공하고, Buffer Pool 내에 없는 페이지에 요청된 연산에 대해서는 디스크에 접근하지 않고 Insert Buffer에 해당 연산을 저장한다. Insert Buffer 역시 다른 Index들과 마찬가지로 B-tree 구조를 지닌다.

Insert Buffer를 사용했을 경우 얻을 수 있는 성능 향상의 정도를 알아보기 위해 실험을 진행하였다. 실험은 SSD(Solid State Drive)와 하드 디스크에서 각각 Insert Buffer를 사용한 경우와 사용하지 않은 경우의 성능을 MySQL Benchmark Tool 중 하나인 'Sysbench'를 사용하여 측정하는 것으로 진행하였다. 실험은 Intel Core i5 4670(3.4GHz x 4) / 8GB DRAM / Ubuntu 14.04.1 환경에서 진행하였으며, SSD와 하드 디스크는 각각 Samsung 850 pro 256GB / Western Digital WD10EZEX 1TB를 사용하였다. 데이터베이스의 크기는 100GB(10GB x 10), Buffer Pool 크기는 1GB, 50개 스레드를 사용하여 각 실험 당 1시간 동안 Benchmark를 진행하였고 그 결과는 다음과 같다.

1) 본 연구는 지식경제부 및 한국산업기술평가관리원의 산업융합원천기술개발사업(정보통신)의 일환으로 수행하였음. [10041244, 스마트TV 2.0 소프트웨어 플랫폼]

2) 이 논문은 2015년도 정부(미래창조과학부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임 (R0126-15-1108, FlashSQL:비휘발성 메모리 기반 개방형 고성능 DBMS 개발)

<표 1> SSD와 HDD에서 Insert Buffer 성능 측정

장치	SSD		하드 디스크	
Insert Buffer	On	Off	On	Off
트랜잭션 처리 (1초당, 회)	946.17	792.81	18.68	13.71
R/W 요청 (1초당, 회)	17031.05	14270.55	336.26	246.77

실험 결과, SSD에서는 Insert Buffer를 사용한 경우가 사용하지 않은 것에 비해 19% 정도 향상된 트랜잭션 처리 성능을 보였으며, 하드 디스크에서는 36% 정도 향상된 성능을 보이는 것으로 나타났다.

앞서 언급한 대로 Insert Buffer의 페이지들도 Buffer Pool 내에 존재하기 때문에 Buffer Pool의 Replacement 정책에 따라 Buffer Pool에서 쫓겨나는 경우가 발생한다. 이 때 Insert Buffer의 페이지들은 디스크 내에 있는 System Tablespace에 저장되고, 이 과정에서 당연히 디스크 입출력이 발생하게 된다.

Insert Buffer의 입출력이 전체 입출력에서 차지하는 비율을 확인하기 위해 다른 실험을 진행하였다. 위에서 진행한 Sysbench Benchmark 실험을 하는 동시에, Linux의 'blktrace'를 사용하여 Benchmark가 진행 중인 SSD와 하드 디스크의 입출력 패턴을 추적하고 저장하였다. blktrace로 저장한 디스크 입출력 패턴을 분석한 결과는 다음과 같다.

<표 2> blktrace 결과 분석

장치		SSD		하드 디스크	
Insert Buffer		On	Off	On	Off
INDEX	읽기	52860783	45496691	776643	700321
	쓰기	15481472	19396386	194294	249256
IBUF _INDEX	읽기	3160746	0	780	0
	쓰기	3408553	0	1219	0

blktrace를 통해 확인된 페이지들의 주소를 통해 실제 디스크 내의 페이지를 찾아 속성별로 분류하고, 분류된 페이지들에 요청된 입출력 수를 분석한 결과이다. <표 2>는 분석한 결과에서 절대 다수를 차지하며 현 실험에서 필요한 INDEX, IBUF_INDEX 페이지에 대한 결과만 정리한 것이다. INDEX는 일반적인 테이블 Index 페이지를 나타내고, IBUF_INDEX는 Insert Buffer 페이지를 나타낸다.

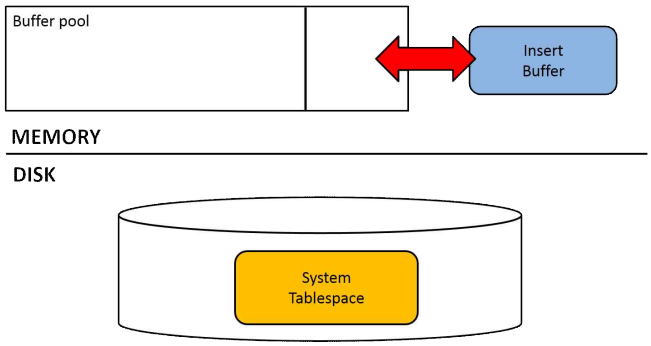
SSD와 하드 디스크의 경우 모두, INDEX 페이지를 읽는 연산이 쓰기에 비해 약 3 ~ 4배가량 더 많이 발생하는 것을 확인할 수 있다. 그런데 IBUF_INDEX에 대해서는 오히려 쓰기 연산이 읽기보다 더 많이 요청된 것을 확인할 수 있고, 특히 SSD에서는 IBUF_INDEX에 대한 쓰기 연산이 전체 쓰기의 약 22% 정도로 상당히 많은 부분을 차지하는 것으로 나타났다.

Insert Buffer를 On, Off 했을 때의 실험 결과를 비교해 보면, Off인 경우에 On일 때보다 INDEX 페이지에 대한 쓰기 연산이 더 많이 발생한 것을 확인할 수 있다. 읽기 연산의 수는 줄어 전체 디스크 입출력의 수가 감소했음에도 불구하고 Off일 때의 성능이 더 낮은 것으로 보아, Insert Buffer를 사용하지 않아 디스크로 바로 요청되는 Random한 입출력의 수가 증가했고 이것이 성능 저하로 연결되었음을 알 수 있다.

Benchmark 결과와 blktrace 분석 결과를 통해 Insert Buffer를 사용하는 것이 데이터베이스의 성능을 향상시키는 데에 도움이 되며, 특히 디스크 쓰기 측면에서 효과를 볼 수 있음을 확인할 수 있었다. SSD의 경우, 쓰기 연산의 속도가 읽기 연산보다 느리다는 특성을 가지고 있기 때문에 Insert Buffer를 개량하면 더 큰 성능 향상을 얻을 수 있을 것이라고 판단된다.

3. Insert Buffer 개선 아이디어

앞선 실험을 통해 Insert Buffer로 인해 발생하는 디스크 입출력을 줄일 수 있다면 Insert Buffer를 사용하여 얻는 성능 향상을 더 크게 가져올 수 있으며, SSD에서 보다 큰 효과를 얻을 수 있을 것을 확인하였다. Insert Buffer로 인한 디스크 입출력을 줄이기 위해 Insert Buffer를 디스크 내의 System Tablespace가 아닌 메모리에 저장하는 방안을 구상하게 되었고, 본 연구에서는 이를 간단하게 구현해볼 수 있는 Ramdisk 활용법을 제시한다.



(그림 1) Ramdisk를 사용하여 메모리 내에 위치시킨 Insert Buffer

Ramdisk를 사용하면 메모리의 일정 영역을 디스크 파티션처럼 사용할 수 있기 때문에, MySQL이 기존에 사용하던 입출력 Routine을 수정하지 않고도 손쉽게 Insert Buffer를 떼어내서 사용할 수 있다는 장점이 있다. Buffer Pool 내의 Insert Buffer 페이지가 쫓겨날 경우, 해당 페이지는 디스크에 저장되는 것이 아닌 Ramdisk 내의 Insert Buffer 영역에 저장되게 된다.

Insert Buffer를 Ramdisk로 옮긴 후, 앞서 진행했던 Sysbench Benchmark, blktrace 분석 실험을 SSD 상에서 동일하게 진행하여, SSD에서의 성능 향상 효과가 어느 정도로 나타나는지 확인하였다. Ramdisk의 최대 크기는 2GB로 지정하였으며, 나머지 조건은 모두 앞선 실험들과 동일하다.

<표 3> SSD에서 Insert Buffer on Ramdisk 성능 측정

Insert Buffer		On	Off	Ramdisk
트랜잭션 처리 (1초당, 회)		946.17	792.81	1071.19
R/W 요청 (1초당, 회)		17031.05	14270.55	19281.51
INDEX	읽기	52860783	45496691	52295720
	쓰기	15481472	19396386	14160546
IBUF _INDEX	읽기	3160746	0	0
	쓰기	3408553	0	0

Insert Buffer를 Ramdisk에 저장하고 Sysbench Benchmark를 실행한 결과, Insert Buffer를 단순히 On 시킨 상태와 비교해서는 13%, Off 대비 35% 정도 트랜잭션 처리 성능이 향상되는 것으로 나타났다. blktrace 분석 결과에서 IBUF_INDEX 페이지의 입출력 처리가 나타나지 않은 것으로 Insert Buffer에 대한 입출력이 SSD로 요청되지 않았다는 것도 확인할 수 있다. 측정된 총 입출력의 수는 Insert Buffer를 옮기지 않고 On 시킨 상황에서는 74,911,554 회, Off 상황에서는 64,893,077 회, Ramdisk에 저장한 상황에서는 66,456,266 회로 나타났다. Insert Buffer를 Ramdisk에 저장한 결과와 단순 On의 결과를 비교하면 Ramdisk에 저장된 상황에서는 Insert Buffer 페이지에 대한 입출력이 발생하지 않아 전체 입출력의 수가 단순 On 상황에 비해 줄어들게 되어 성능 향상에 영향을 준 것을 알 수 있고, Off 상황과 비교하여서는 전체 입출력 수는 비슷하지만 성능 차이는 크게 나는 것으로 보아 Insert Buffer를 사용하지 않을 경우에는 Random한 입출력으로 인한 성능 저하가 발생한다는 것을 다시 한 번 확인할 수 있었다.

4. 결론 및 향후 연구

MySQL의 InnoDB는 Random 입출력으로 인해 발생하는 성능 저하를 막고자 Insert Buffer를 사용한다. 실험을 통해, Insert Buffer를 사용하는 것이 성능 향상에 도움이 된다는 점과 Insert Buffer를 수정하여 더 큰 성능 향상을 이끌어 낼 수 있음을 모두 확인할 수 있었다. 본 연구에서는 개선 방안으로 Insert Buffer를 디스크 장치보다 속도가 훨씬 빠른 메모리에 저장하는 방법을 제시하였으며, Ramdisk를 사용한 간단한 구현을 통해 디스크 입출력은 줄고 데이터베이스 성능은 향상된다는 결과도 확인하였다. 다만, 이번 실험에 사용한 메모리 장치는 일반적으로 사용하는 DRAM으로 휘발성을 가지기 때문에 저장된 Insert Buffer를 영구적으로 저장하는 데에는 문제가 있다는 한계점을 지니고 있다. 따라서 향후 비휘발성 장치를 활용하면서 동일한, 또는 이번 연구 이상의 성능 향상을 이끌어 낼 수 있는 방안에 대하여 연구를 지속할 계획이다.

참고문헌

[1] http://mysqlserverteam.com/the-innodb-change-buffer/

[2] Baron Schwartz, Peter Zaitsev, Vadim Tkachenko, Jeremy D. Zawodny, Arjen Lentz, Derek J. Balling, “High Performance MySQL”, 2nd Ed. O’Reilly Media

[3] DongZhe Ma, JianHua Feng, GuoLiang Li, “A Survey of Address Translation Technologies for Flash Memories”, ACM Computing Survey, 2014

[4] Sang-won Lee, Bongki Moon, “Design of Flash-Based DBMS: An In-Page Logging Approach”, ACM SIGMOD, 2007