MySQL 버퍼 풀 인스턴스 개수에 따른 Latch 경합 패턴 분석

양원석, 안미진, 이상원

성균관대학교

{aoo0227, meeeeejin, swlee}@skku.edu

Analyzing Latch Contention Patterns By Varying the Number of Buffer Pool Instances in MySQL

Won-Suk Yang, Mijin Ahn, Sang-Won Lee
Sungkyunkwan University

요 약

하나의 큰 버퍼 풀을 여러 개의 인스턴스로 나눈 다중 버퍼 풀을 구성함으로써, latch 경합을 줄일 수 있고, 이를 통해 동시성을 향상시킬 수 있다. 그러나 다중 버퍼 풀의 성능이 단일 버퍼 풀보다 항상 좋은 것은 아니며, 버퍼 풀 인스턴스 개수에 따라 성능 향상의 정도가 달라진다. 본 논문에서는 버퍼 풀 인스턴스의 개수가 latch 경합에 미치는 영향을 실증적으로 분석하기 위해, MySQL 5.7 서버에서 버퍼 풀 인스턴스 개수를 변경해가며 실험을 수행하였다. 대표적인 OLTP 워크로드인 TPC-C와 LinkBench 워크로드를 이용해 latch 경합 패턴을 분석하였고, 워크로드와 버퍼 풀 개수가 latch 경합에 끼치는 영향을 분석하였다.

1. 서 론

IT 기술이 발전하면서 페이스북이나 아마존과 같은 대규모 웹 서비스들이 등장하고 있는데, 이러한 웹 서비스의 규모가 커짐에 따라 DBMS에 요구되는 데이터 처리량도 함께 급증하고 있는 추세이다. 이로 인해 대용량 데이터를 효율적으로 관리해 줄 수 있는 DBMS의 중요성이 점점 더 높아지고 있다. 데스크탑과 서버 환경에서 주로 사용되는 대표적인 DBMS는 MySQL이며, 스토리지 엔진으로 InnoDB를 주로 사용한다. InnoDB 스토리지 엔진은 ACID를 보장하기 위해 데이터 처리를 트랜잭션(transaction) 단위로 지원하며, 디스크 및 서버에서 장애가 발생할 경우 복구가 용이하다. 이러한 InnoDB는 대용량 데이터 처리와 보다 나은 성능을 위해, 다중 버퍼 풀(multiple buffer pool) 기법을 사용한다. 이를 통해 여러 쓰레드(thread)의 동시 접근으로 발생하는 병목현상(bottleneck)을 완화시킬 수 있다. 그러나 버퍼 풀 인스턴스(instance)의 개수에 따라 latch 경합(contention)의 정도가 달라지기 때문에, 인스턴스 개수를 적절히 조정할 필요가 있다. 본 논문에서는 latch 경합을 최소화할 수 있는 조건을 찾기 위해 실험으로 latch 경합 패턴을 분석하였다.

본 논문의 대략적인 구성은 다음과 같다. 먼저 2장에서 배경지식으로 버퍼 풀과 latch 경합에 대해 소개한다. 3장에서는 버퍼 풀 인스턴스 개수에 따라 달라지는 latch 경합의 패턴을 실험을 통해 확인한다. 마지막으로 4장에서 실험을 통해 내린 결론과 향후 연구에 대해 소개하며 논문을 마무리한다.

2. 배경지식

2.1. MySQL/InnoDB의 버퍼 풀

InnoDB는 데이터를 메모리에 캐싱(caching)하기 위해 버퍼 풀[1]이라는 저장 영역을 사용한다. InnoDB 버퍼 풀이 어떤 식으로 동작하는지 이해하고, 자주 접근하는 데이터를 메모리에 저장하는 것의 이점에 대해 이해하는 것은 MySQL 튜닝의 중요한 일환이다. InnoDB 버퍼 풀은 다양한 면에서 성능을 향상시키는데, 이상적으로는 버퍼 풀의 크기를 가능한 한 큰 값으로 설정하여, 서버의 다른 프로세스들이 과도한 페이징 없이 실행할 수 있게 충분한 메모리를 남겨 둔다. 버퍼 풀이 클수록 InnoDB는 in-memory 데이터베이스와 같은 역할을 하는데, 디스크에서 데이터를 한 번 읽어 들인 다음부턴 데이터를 읽을 때 주로 메모리에서 읽어 온다. 버퍼 풀에 저장되거나 버퍼 풀에서 읽혀지는 각 페이지는 해시 함수를 사용하여 임의의 버퍼 풀 중 하나에 할당된다. 각 버퍼 풀은 3가지 리스트를 이용해 버퍼의 페이지를 관리한다. 리스트로는 free 페이지를 가지고 있는 free 리스트, dirty 페이지를 가지고 있는 flush 리스트, 마지막으로 버퍼에서 사용되는 페이지를 가지고 있는 LRU(least recently used) 리스트가 있다. 그리고 버퍼 풀에 속한 모든 데이터 자원들은 각 버퍼 풀 자체의 latch로 보호된다.

단일 버퍼 풀이라도 버퍼 풀의 크기가 크면 디스크 접근 없이 메모리상에서 많은 쿼리를 처리할 수 있다. 그러나 여러 쓰레드가 단일 버퍼 풀에 접근하려는 경우 병목현상이 발생할 수 있다. 따라서 수 GB의 버퍼 풀을 여러

인스턴스로 나누면, 멀티 코어 시스템의 MySQL 환경에서 확장성과 작업의 병렬성을 향상시킬 수 있다. 이를 통해 캐싱 되어 있는 페이지를 서로 다른 쓰레드가 읽거나 쓸 때 발생하는 latch 경합을 줄임으로써 동시성(concurrency)을 향상시킬 수 있다. 그러나 버퍼 풀 인스턴스가 과도하게 많아지면 성능이 떨어지기 때문에[2], 본 논문에서는 버퍼 풀 인스턴스의 개수를 다양하게 변경해 가며 실험을 수행하였고, 어떤 조건 하에서 성능이 향상되는지 알아보았다.

2.2. MySQL의 동기화(synchronization)를 위한 Latch

InnoDB는 여러 스레드가 공유 자원에 접근할 때 생기는 문제를 미연에 방지하고, 동기화를 효과적으로 수행하기 위해 latch라 불리는 구조체를 사용한다. Latch는 배타적 접근을 할 때 사용하는 mutex와 공유 접근을 허용하는 rw-lock을 모두 포함하는 일반적인 용어이다.

먼저 mutex(mutual exclusion)란, 직역하면 상호 배제라고 하는데, 실행된 쓰레드가 많은 환경에서, 임계 영역(critical section)(프로그램 상에서 동시에 실행될 경우 문제를 일으킬 수 있는 부분)을 가진 쓰레드들의 실행 시간(running time)이 서로 겹치지 않게 만든다[3]. 즉, 각각의 쓰레드가 공유 자원에 대한 작업을 단독적으로 실행할 수 있도록 자원에 대한 접근 제한을 강제하는 동기화 메커니즘이다. 이와는 대조적으로, rw-lock은 공유 접근을 허용하는 동기화 메커니즘이다. 한번 rw-lock이 획득되면, 다른 쓰레드는 해당 자원을 읽을 수는 있지만, 쓰기는 할 수 없다. 즉 읽기 작업에 대해서는 공유 접근을 허용한다.

이러한 latch는 여러 쓰레드가 동시에 실행될 때 필연적으로 경합을 하게 되므로, 본 논문에서는 워크로드 수행 중 latch 획득에 소요되는 시간을 측정하였다. 그리고 이 시간을 latch 경합의 패턴을 분석하는 성능 지표로 활용하였다.

3. 실 험

표 1 실험 환경

CPU	Intel Core i5-4670 @ 3.40GHz (4 CPUs)
Memory	7.5GB
OS	Ubuntu 16.04 LTS
Kernel	Version 4.13.0-39-generic
DBMS	MySQL 5.7.21

본 논문에서는 버퍼 풀 인스턴스 개수 변화에 따른 MySQL latch 경합의 패턴을 분석하기 위해, 위의 표 1과 같은 환경에서 실험을 진행하였다. 본 실험에서는 대표적인 OLTP 워크로드인 TPC-C[4]와 페이스북에서 제공하는 소셜 그래프 데이터 관련 워크로드인 LinkBench[5]를 사용해 벤치마크를 수행하였다. 실험에 사용한 데이터베이스의 크기는 약 50GB이고, 버퍼 풀의 크기는 데이터 크기의 10%인 5GB이다.

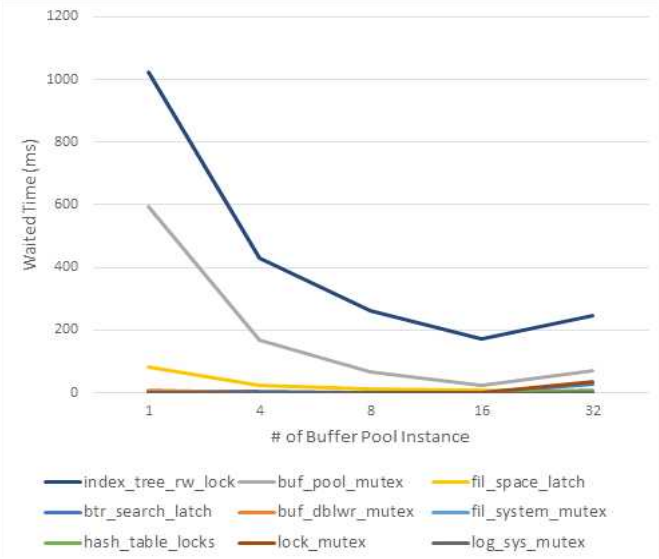

그림 1 버퍼 풀 인스턴스 개수에 따른 Latch 경합 (TPC-C)

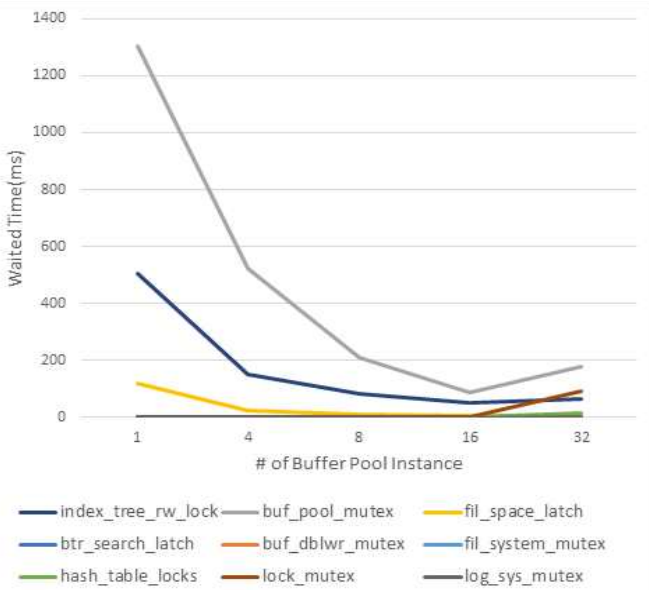

그림 2 버퍼 풀 인스턴스 개수에 따른 Latch 경합(LinkBench)

3.1. 성능 평가

2.2장에서 설명했듯이, latch 획득에 소요되는 시간을 latch 경합의 성능 지표로 활용하였으며, 약 한 시간 동안 TPC-C 벤치마크를 수행한 결과는 그림 1, LinkBench 벤치마크를 수행한 결과는 그림 2와 같다. 버퍼 풀 인스턴스의 개수를 1, 4, 8, 16, 32로 변화시켜가며 실험을 진행하였고,

그래프에 표시된 결과 값은 벤치마크 수행 중 latch 경합 때문에 발생한 대기 시간(waited time)을 버퍼 풀 인스턴스 개수로 나눈 값이다.

실험 결과, 인스턴스의 개수가 16개까지 증가했을 때, 버퍼 풀을 여러 개로 나눌수록 성능도 향상되었다. 그러나 인스턴스 개수가 32개로 너무 많아지게 되면, 오히려 성능이 나빠지는 것을 관찰할 수 있었다. 각각의 인스턴스에서 측정된 대기 시간을 합산했을 때, 대기 시간이 가장 짧았던 인스턴스 개수는 두 워크로드 모두 16이었다. 대기 시간이 가장 길었던 인스턴스 개수는 두 워크로드 모두 1, 즉 단일 버퍼 풀로, 16일 때와 비교 했을 때 TPC-C는 약 8.5배, LinkBench는 약 13배의 차이를 보여주었다. 즉, 다중 버퍼 풀을 구성하게 되면 latch 경합 문제를 상당 부분 완화시킬 수 있지만, 너무 많은 개수로 나누게 되면 버퍼 풀 자체의 용량이 작아지고, 그에 따른 역효과를 초래할 수 있다.

3.2. Latch 경합 패턴 분석

본 논문의 실험에서는 MySQL에 존재하는 여러 latch 중 9개의 주요 latch를 선정하여 패턴을 분석하였다. 실험 결과, 워크로드의 종류나 인스턴스 개수에 상관없이 경합이 심한 latch의 거의 대부분은 읽기나 쓰기 작업을 수행하기 위해 반드시 획득하여야 하는 latch들이었다. TPC-C 기준 경합이 가장 심한 latch는 데이터베이스 페이지에 접근하기 위해 인덱스 트리를 탐색할 때 획득해야 하는 rw-lock으로, index_tree_rw_lock이었다. LinkBench 기준 경합이 가장 심한 latch는 버퍼에 저장된 페이지에 직접적으로 접근해야 할 때 획득해야 하는 mutex인 buf_pool_mutex였다. 인스턴스 개수가 32일 때는 이전에는 두드러지지 않았던 latch들이 드러나는 현상도 볼 수 있는데, 이것 역시 버퍼 풀을 과도하게 많은 인스턴스로 나누면 역효과가 일어난다는 반증이기도 하다.

결과적으로, 버퍼 풀 인스턴스의 개수가 증가할수록 latch 경합 수치가 감소했고, 9개의 latch를 대기 시간 순으로 나열했을 때 그 순위 역시 1, 4, 8, 16일 때는 거의 동일했다. 그러나 인스턴스 개수를 32로 늘렸을 때는 latch 경합 수치가 역으로 증가하였고, 앞서 언급한 대기 시간 상위 1, 2위를 차지하는 두 가지 latch를 제외하고는 이전과 완전히 다른 패턴이 관찰되었다.

4. 결론 및 향후 계획

본 논문에서는 버퍼 풀 인스턴스의 개수에 따라 달라지는 latch 경합을 실험을 통해 비교 및 분석하였고, 그에 따른 latch들의 패턴을 관찰하였다. 실험 결과, 인스턴스 개수가 1인 단일 버퍼 풀보다 인스턴스 개수가 4, 8, 16, 32인 다중 버퍼 풀이 latch 경합을 줄이고 성능을 향상시켰다. 특히 인스턴스 개수가 16개일 경우엔 단일 버퍼 풀을 사용했을 때보다 TPC-C에선 약 8.5배, LinkBench에선 약 13배가량 경합이 줄어드는 현상을 관찰할 수 있었다. 또한 버퍼 풀을 여러 개의 인스턴스로 나눌수록 대체적으로 경합이 줄어들었지만, 인스턴스 개수가 32일 때, 즉 버퍼 풀을 과도하게 나눌 경우엔 latch 경합이 오히려 커지는 것을 확인할 수 있었다.

본 논문에서는 MySQL 5.7 버전 기준으로 latch 경합을 분석했지만, 향후 연구로 MySQL 5.6, 8.0 등 다른 버전의 latch 경합도 비교 및 분석할 계획이다. 이를 통해 latch 경합의 전반적인 성능을 측정하고, 9개의 각 latch가 경합을 줄이기 위해 어떻게 개선되었는지 연구할 계획이다.

사사

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

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

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

본 연구는 과학기술정보통신부 및 정보통신기술진흥센터의 SW중심대학지원사업의 연구결과로 수행되었음(2015-0-00914)

5. 참고문헌

[1] MySQL, https://dev.mysql.com/doc/refman/5.5/en/innodb-buffer-pool.html

[2] 송용주, 이민호, 엄영익, “Multiple Buffer Pool이 MySQL 성능에 미치는 영향 분석”, 한국통신학회 2016년도 동계종합학술발표회, 204쪽, 2016년

[3] Andrew S. Tanenbaum, “Modern Operating Systems”, Pearson Education International, 2009

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

[5] LinkBench, https://github.com/facebookarchive/linkbench

[6] Stavros Harizopoulos, Daniel J. Abadi, Samuel Madden, Michael Stonebraker, “OLTP Through the Looking Glass, and What We Found There”, SIGMOD repeatability committee, 2008