TPC-C 워크로드에서의 MySQL/InnoDB 락 경합 분석

김경민 박종혁 안미진 이상원
성균관대학교
{lufovic77, akindo19, meeeejin, swlee}@skku.edu

Analysis of MySQL/InnoDB Lock Contention on TPC-C Workload

Kyungmin Kim, Jong-Hyeok Park, Mijin An, Sang-Won Lee
Sungkyunkwan University

요 약

TPC-C는 실제 OLTP 워크로드에 적합한 트랜잭션들과 테이블, 그리고 튜플 접근 특성을 가진 벤치마크이다. PMM은 MySQL과 같은 상용 데이터베이스에 대해 TPC-C 벤치마크를 이용한 실험 수행을 그래프나 테이블 형식으로 분석할 수 있는 오픈소스 툴이다. 본 논문에서는 TPC-C 벤치마크 결과를 PMM으로 살펴보던 중, 5개의 테이블 중 유독 Stock 테이블에서 락 경합이 심한 것을 발견하고, 쓰기 락과 읽기 락에 대해 락 경합의 원인을 PMM과 소스코드 분석을 통해 각각 확인하였다. Stock 테이블에 접근하는 New order 트랜잭션은 TPC-C 워크로드에서 가장 많은 비율을 차지하며, 빈번한 update와 읽기 시 gap lock을 쥐는 접근 형태를 보이기 때문에 Stock 테이블에서 락 경합이 심한 것을 확인하였다.

1. 서 론

TPC-C (Transaction Processing Performance Council)는 제품을 관리, 판매, 그리고 분배하는 산업들 중 대표적으로 도매업을 묘사하는 트랜잭션들로 이루어진 벤치마크이다. TPC-C를 구성하는 5개의 트랜잭션들은 동시성을 띠며, 각기 다른 복잡도를 가지고 있다. 이들은 9개의 테이블로 이루어진 데이터베이스를 접근하며, 성능은 분당 트랜잭션 수인 TpmC (Transactions per minute)로 나타낼 수 있다. [1]

TPC-C 데이터베이스는 warehouse가 가장 위에 있는 계층적 구조를 띄고 있기 때문에, 각 테이블들의 튜플 개수는 warehouse수에 의해 변동된다. 5개의 트랜잭션들 중 New order와 Payment 트랜잭션이 전체 트랜잭션의 약 90%를 차지하며, New order는 대략 50%, Payment는 최소 43%를 차지한다고 볼 수 있다. [2]

TPC-C는 TPC-A와는 다르게 튜플에 대해 일관된 접근이 아닌 편향된 접근을 한다. 이러한 접근은 실제 OLTP (Online transaction processing) 워크로드에서 일반적이기 때문에 데이터베이스 플랫폼들의 성능을 비교하는데 매우 효과적이다.

PMM (Percona Monitoring and Management) 이란 MySQL을 비롯한 여러 데이터베이스 엔진들에 대해 다양한 metric을 그래프로 확인할 수 있는 툴이다. PMM을 이용하면, 실시간으로 MySQL의 쿼리 정보 및 I/O 정보들을 수집하여 그래프로 시각화 하여 볼 수 있다.

PMM을 이용하여 TPC-C 결과를 그래프로 분석하던 중, 유독 Stock 테이블에서 락 대기 시간이 긴 것을 발견하였다. 락 경합이 심해지면 동시성이 저하되기 때문에, 최근 많이 사용되는 멀티코어 환경에서 성능상의 이득을 볼 수 없어 이유를 규명해 내는 것이 매우 중요하다. 본 논문에서는 Percona에서 제공하는 PMM 툴을 기반으로 어떤 트랜잭션들이 충돌을 일으키며, 왜 Stock 테이블에서 락 경합이 극심한지를 밝혀낸다.

본 논문의 구성은 다음과 같다. 2장에서는 Percona에서 제공하는 PMM 오픈소스 툴에 대해 알아본다. 3장에서는 TPC-C 실험 환경과 결과를 PMM을 통해 소개하고, 4장에서는 Stock 테이블에서 락 경합이 많이 발생하는 이유를 TPC-C 소스코드 분석을 통해 알아낸다. 마지막으로 5장에서는 결론을 제시하고 논문을 마무리한다.

2. Percona Monitoring and Management [3]

PMM (Percona Monitoring and Management)은 MySQL, MongoDB을 포함한 상용 데이터베이스 엔진들의 성능을 관리 및 모니터링 할 수 있는 오픈소스 플랫폼이다. PMM은 PMM Server와 Client로 구성되며, PMM Server에 Client를 연결하면 PMM Server에서 여러 정보들을 수집하여 테이블이나 그래프의 형태로 정리하여 웹을 통해 보여준다. 특히 MySQL에서는 row 별 락 시간, I/O, 로깅 성능, 버퍼 풀 I/O 등 성능을 평가할 수 있는 다양한 척도들이 있고, 5.6 버전 이후부터는 performance schema 옵션을

이용하여 쿼리 정보에 대한 자세한 정보를 얻을 수 있다.

[표 1] Performance schema 옵션

InnoDB metrics	innodb_monitor_enable=all
Slow query log	performance_schema=ON

[표 1]을 MySQL 서버 configuration 파일에 추가하면 performance schema 옵션을 사용할 수 있게 된다.
본 논문에서는 PMM에서 제공하는 다양한 성능평가 환경 중, Performance Schema SQL and External Locks (seconds)를 이용하여 특정 테이블의 읽기와 쓰기에 걸리는 락 대기 시간을 분석하였다.

3. TPC-C 실험 결과

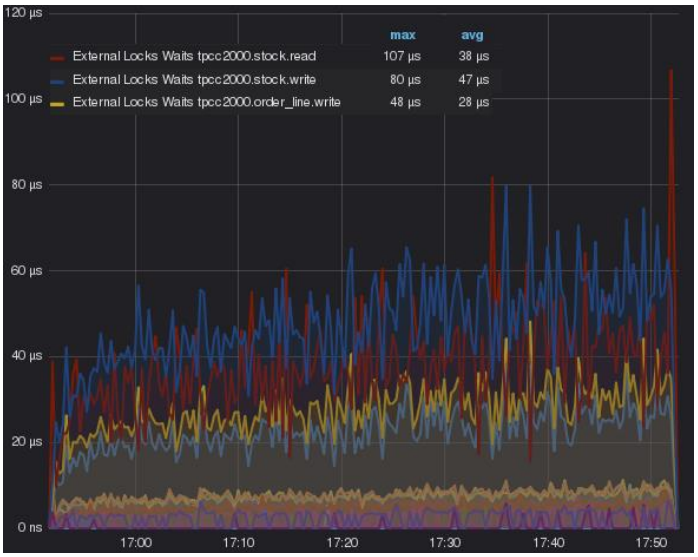
3-1 실험 환경

본 논문에서는 MySQL 5.7.24 stable version으로 TPC-C 벤치마크를 수행했으며, 이를 PMM과 연결하여 결과를 분석하였다. 데이터베이스의 크기는 2000 warehouse이며, 버퍼 크기는 16GB이다. 실험은 64 클라이언트로 3600초 동안 진행했다. 자세한 실험 환경은 [표 2]와 같다.

[표 2] 실험 환경

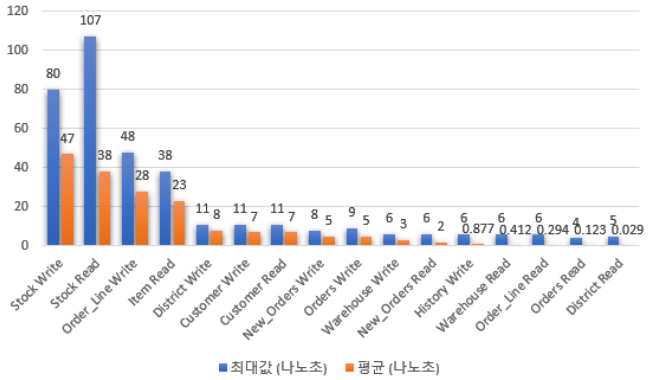
OS	Ubuntu 16.04.6 LTS
Processor	Intel(R) Core(TM) i7-6700 CPU @ 3.40GHz
Memory	23 GB
SSD (data)	Samsung SSD 850 PRO 256GB
SSD (log)	Samsung SSD 840 PRO 256GB
Benchmark	TPC-C (2000 warehouse 200GB)

3-2 PMM 관측



[그림 1]. TPC-C 실험 PMM 관측

[그림 1]은 위에서 수행한 TPC-C 실험을 PMM에서 제공하는 성능평가 환경 중 하나인 External Locks Waits를 통해 확인한 결과이다. 상위 두개의 Stock read와 Stock write 그래프가 전체적으로 높은 수치를 보이는 것을 알 수 있다.



[그림 2] 락 대기시간

[그림 2]는 각 테이블에 접근하는 트랜잭션마다 가지는 락 대기시간을 평균치와 최대치로 정리한 결과이다. 이를 통해 유독 Stock 테이블에서 쓰기와 읽기 락 대기 시간이 다른 테이블에 비해 높은 것을 확인할 수 있다.

4. 락 경합 원인 분석

4-1 Multi-Version Concurrency Control

MySQL/InnoDB는 동시성 제어 기법으로 MVCC (Multi-Version Concurrency Control)을 사용한다. MVCC는 각 row마다 변경이 일어나면 복사본을 만들어 old version으로 저장하여 버전 관리를 한다. 한 트랜잭션이 특정 오브젝트에 대해 쓰고 있는 중에 다른 트랜잭션이 동일한 오브젝트를 읽으려고 하면 예전 버전을 읽게 하기 때문에 Read-Write나 Write-Read 충돌이 발생하지 않는 것이 기본적인 가정이다. 다만, Write-Write (이하 W-W) 충돌은 발생할 것이라는 가정을 가지고 분석을 시작했다.

4-2 Stock 테이블 쓰기 락 대기

Stock 테이블에 접근하는 트랜잭션 중 쓰기를 하는 쿼리가 존재하는 트랜잭션은 New order 트랜잭션 밖에 없으며, 해당 쿼리는 [표 3]과 같다.

[표 3] Stock 테이블 쓰기 SQL문

New order 트랜잭션
UPDATE Stock SET s_quantity = :s_quantity WHERE s_i_id = :ol_i_id AND s_w_id = :ol_supply_w_id;

New order 트랜잭션은 TPC-C 벤치마크에서 가장 빈번한 트랜잭션이며, 전체 트랜잭션 중 비율은 이미 소개하였듯이 최대 45%를 차지한다.

각 New order 트랜잭션은 평균적으로 10개의 item에 대한 주문을 진행하기 때문에, 각 item들에 대응되는 stock의 튜플 들을 update하는 [표 3]의 쿼리에서 W-W 충돌이 빈번하게 발생함을 알 수 있다.

따라서, Stock 테이블에서 쓰기 락 대기시간이 긴 이유는 stock에 쓰기 동작을 하는 New order

트랜잭션이 TPC-C에서 가장 자주 일어나며, [표 3]의 update 쿼리의 특성상 주문하는 item의 개수에 영향을 받기 때문임을 밝혀낼 수 있다.

4-3 Stock 테이블 읽기 락 대기

MVCC의 기본적인 가정은 쓰기 동작이 읽기를 방해하지 않는 것이다. 그러나, PMM 결과에서는 Stock 테이블 읽기에서도 락 대기 시간이 높게 측정되어 Stock 테이블을 읽는 쿼리가 존재하는 트랜잭션들을 분석했다. Stock 테이블을 읽는 트랜잭션은 Stock level과 New order 트랜잭션인데, 기본적인 select 문으로 구성된 Stock level 트랜잭션과는 달리 New order에서는 [표 4]와 같이 읽기 시 gap lock을 쥐는 것을 알 수 있다.

[표 4] Stock 테이블 읽기 SQL문

New order 트랜잭션
SELECT s_quantity, s_data, s_dist_xx INTO :s_quantity, :s_data, :s_dist_xx FROM Stock WHERE s_i_id = :ol_i_id AND s_w_id = :ol_supply_w_id FOR UPDATE;

Gap lock은 주어진 범위에 존재하는 모든 인덱스 레코드들에 대해 락을 부여하는 락킹 기법으로, phantom 현상을 방지하는 효과가 있지만, 동시성을 저하시키는 단점이 존재한다. [4] 역시 가장 빈번한 트랜잭션인 New order에 Stock 테이블에 대해 gap lock을 쥐는 쿼리 문이 존재하는 만큼, Stock 테이블에서 읽기 락 대기시간이 긴 이유를 밝혀낼 수 있다.

5. 결론

본 논문에서는 TPC-C 벤치마크를 이용하여 실험을 하였고, 이 결과를 PMM으로 분석하여 Stock 테이블에서 트랜잭션 간에 락 경합이 가장 심한 원인을 분석했다.

소스코드 분석 결과, 쓰기 락 경합의 경우 New order 트랜잭션이 TPC-C에서 가장 빈번하게 일어나는 트랜잭션이며, 각 트랜잭션당 10개의 item에 대응되는 Stock 튜플들을 update 하기 때문에 락 경합이 심한 것을 확인하였다.

읽기 락 경합의 경우, 역시 New order 트랜잭션에서 Stock 테이블에 gap lock을 거는 쿼리가 존재했기 때문에 락 경합이 심한 것을 확인하였다.

향후연구로는 PMM 설정을 세분화하여 대기 시간이 긴 쿼리문과 여러 트랜잭션 간의 락 충돌을 확인해보고, Stock 테이블 다음으로 락 경합이 심했던 order line 테이블에 대한 락 충돌 분석 연구를 수행할 것이다.

사사

이 논문은 2019년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임 (No.2015-0-00314,비휘발성 메모리 기반 개방형 고성능 DBMS 개발)

이 성과는 2019년도 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임 (No. 2018R1A2B2005502).

참고 문헌

[1] Transaction Processing Performance Council, http://www.tpc.org/tpcc/ (2019-04-30 방문)

[2] Stonebraker, Michael, et al. "The end of an architectural era:(it's time for a complete rewrite)." Proceedings of the 33rd international conference on Very large data bases. VLDB Endowment, 2007

[3] Percona Monitoring and Management, https://www.percona.com/ (2019-05-05 방문)

[4] MySQL 5.7 Reference Manual, https://dev.mysql.com/doc/refman/5.7/en/innodb-locking.html (2019-05-06 방문)