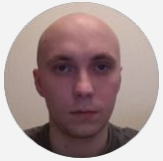

Николай Ихалайнен
Percona

PGDAY'17 RUSSIA
КОНФЕРЕНЦИЯ ПО БАЗАМ ДАННЫХ

Обработка транзакций в MySQL-derived базах

ACID, MVCC, репликация

PERCONA

Примеры исп. транзакций

- Перевод денег со счёта на счёт
- Заказ авиабилетов с пересадками
- Регистрация брака (двухфазный коммит)
- Продажа дома (spheres of control)
- Скандинавский аукцион – неравномерное время

Обработка транзакций

Способ обработки данных/вычислений при котором работу дробят на отдельные **неделимые** операции, называемые транзакциями

Из определения

- Частично выполненная транзакция - это не работа, а повреждение данных
- Завершённая транзакция переводит систему в другое состояние с сохранением логических ограничений (constraints)

Из определения

- Две одновременных транзакции всегда выглядят как одна выполненная за другой (нет действий в середине)
- Слабое ограничение: транзакции отражают реальный мир, нельзя терять завершённую (committed) работу.

ACID

- Atomicity: изменение или было полностью или его не было
- Consistency: Корректное изменение состояния системы
- Isolation: При одновременном исполнении транзакции T, остальные транзакции исполняются или до T или после, но не одновременно

ACID транзакции

- Isolation: каждая программа изолирована от работы и сбоев параллельной работы других
- Granularity: Эффекты отдельных транзакций можно откатить (rollback), дискретность сбоев
- Consistency: Rollback делает все структуры согласованными
- Durability: после commit никакая работа не теряется

Consistent

Это не тоже самое, что в теореме CAP
Просто, данные не сломаны

Внутри транзакции

- Statements
- Save points
- Nested transactions
- Resource managers coordination

- Internal actions for rollback
- Actions to block other transactions modifications
- Maintain WAL
- Sync with replication log

В MySQL

- Нет вложенных транзакций (но можно эмулировать через save points)
- Не все действия транзакционные
  - DDL (большой шаг в MySQL 8.0)
  - Нетранзакционные движки хранения
  - Блокировка таблиц (lock/unlock table)

Нетранзакционные движки

- MyISAM, Merge, Memory, CSV, ARCHIVE
- Blackhole (не durable :D)
- Federated (может быть атомарным для транзакционных движков)
- MariaDB: Aria – crash safe & atomic

Основной - InnoDB

- Полная поддержка ACID
- Консистентный с логами репликации
- MVCC
- save points

InnoDB Atomicity

- Частичные изменения не видны из-за MVCC
- Изменения текущей транзакции можно откатить с Undo log

InnoDB Undo logging

- Строка указывает на предыдущую версию данных
- Предыдущие версии образуют цепочку всех изменений записи
- DELETE: помечает запись флагом

InnoDB undo logging2

- Накладные расходы 1 бит и 7 байт DB_ROLL_PTR для каждой строки
- Реализовано как 128 сегментов отката

TokuDB MVCC

- Не такое, как в InnoDB
- Нет undo space и потоков Purge
- Очистка, как часть нормальных операций
- Итерация по leafentries пока не получена нужная транзакция

TokuDB rollback

- Каждая транзакция поддерживает лог отката при операциях на Fractal Tree Index
- Слишком большой rollback log помещается в файл

InnoDB Isolation

- Реализована на блокировках
- REPEATABLE READ
  - по-умолчанию
  - Снапшот начиная с первого чтения
  - Для уникальных индексов и условий блокируются только найденные строки

InnoDB REPEATABLE READ

- Для неуникальной выборки
  - Блокируется диапазон строк
  - Включая промежутки между строками (gap, next-key locking)

InnoDB READ COMMITTED

- Gap locking выключены
- Блокировки на несовпавшие строки отпускаются после проверки WHERE
- UPDATE: semi-consistent read
  - Чтение последней версии
  - При совпадении – повтор чтения с блокировкой

InnoDB READ UNCOMMITTED

- Такой же как RC
- SELECT старается вернуть самую последнюю версию
- Нет существенных улучшений производительности

InnoDB SERIALIZABLE

- Такой же как RR
- Все SELECT в транзакциях становятся SELECT … LOCK IN SHARED MODE
- Одиночные с autocommit=1 как RR
- Очень плохо масштабируется
- Обычно легче использовать RR и для нужных запросов LOCK IN SHARED MODE

InnoDB реализация блокировок

PGDAY RUSSIA '17

- Multiple Granularity Locking
  - “Intention locks” на таблицы
  - Это не тоже самое что и insert intention gap lock
- Gap locks – избегаем фантомы
- 100k строк – 44КБ оперативной памяти

Сервисы подсистемы блокировки

- Создать, получить, освободить блокировки строк
- Блокировки таблиц
- Структуры данных для поиска блокировки
- Будить потоки спящие на блокировках
- Deadlock detection

PGDAY'17 RUSSIA

Lock bitmap

```
struct lock_rec_t {
        ulint   space;                  /*!< space id */
        ulint   page_no;                /*!< page number */
        ulint   n_bits;                 /*!< number of bits in the lock
                                        bitmap; NOTE: the lock bitmap is
                                        placed immediately after the
                                        lock struct */
};
```

Транзакции и блокировки

- Массив блокировок для каждой транзакции
- Глобальный хеш по (space_id, page_no)
- Implicit Row locks
  - Вычисляемые x-locks на запись
    - Не касается gap блокировок

TokuDB Isolation

- Two-phase locking
  - Фаза расширения – получаем все нужные блокировки, не отпускаем полученные
  - Фаза схлопывания – лишние отпускаем, новых не получаем

TokuDB Isolation2

- Блокировки на чтение – эксклюзивные
  - Была реализация с разделяемыми
- На запись – эксклюзивные

TokuDB Isolation3

- Serializable
  - Insert select
    - Получить все блокировки чтения
    - Получить необходимые блокировки на запись
  - Create select
    - Все чтение
    - Блокировка таблицы на запись
  - Select, select lock in shared mode – все чтения

TokuDB Isolation4

- Repeatable read
  - Insert select, create select
    - Как и serializable
  - SELECT: снапшот, нет блокировок
- Read committed
  - Insert select: все блокировки записи
  - Select: такой же как в RR
- Read uncommitted: create select как RC

TokuDB Isolation5

- На всех уровнях изоляции:
  - Select for update – блокировки записи
  - Update, delete:
    - Чтения на все диапазоны
    - Записи – все модифицированные ключи

TokuDB блокировки реализация

- Lock tree
  - Структура в оперативной памяти
  - Все блокировки живых транзакций
  - Все транзакции которые ждут блокировок

TokuDB lock ranges

- Тип транзакции и блокировки
- Копия левой и правой границы
- Такая же как точечная, если границы равны

TokuDB Lock tree escalation

- Склеивание диапазонов блокировок очень больших транзакций
- Включая промежутки
- Для больших транзакций (и если ½ памяти под блокировки занята)
- Другие большие транзакции ждут на блокировке,
- Маленькие транзакции работают

MyRocks Isolation

- Только Read committed & Repeatable reads
- Используются снапшоты (объект RocksDB)
- SELECT – всегда на момент создания снапшота
- INSERT, UPDATE – ошибка, если запись изменена другой транзакцией после

MyRocks Isolation2

- Repeatable read: снапшот живёт всё время транзакции
- Read committed: снапшот обновляется после каждого запроса
- Снапшот создаётся как можно позднее – при первом чтении данных

MyRocks lock manager

- Эксклюзивные и разделяемые блокировки
- Блокировку строк можно выключить через unique_check=0, например для пакетной загрузки

MyRocks блокировки

- Изменение строки после старта транзакции
  - Разрешено в READ COMMITTED
  - Запрещено в Postgres и MyRocks RR
  - InnoDB разрешает для RC и RR

Consistency

- Innodb – transaction id + undo log
- TokuDB&MyRocks – снапшоты на базе структур (FT и LSM)
- Non-transactional: блокировка таблиц
- DDL: metadata locking
- Replication & NDB – eventual consistency

Durability

- Реализовано на WAL
- Свой для каждого движка хранения
- Свои настройки сброса на диск после транзакции
- XA-транзакции между движком и логом репликации

InnoDB durability

- Набор файлов формирующих циклический буфер redo log
- Fuzzy checkpointing
  - Модифицированные страницы сбрасываются маленькими порциями
  - Не надо останавливать запросы

InnoDB durability

- LSN используется для создания transaction id
- Double write segment: пишем страницу в два места – лекарство от частичной записи на кривых файловых системах
- Позиция репликации на слейве может быть востановлена
- Group commit & 2PC с логом репликации

TokuDB durability

- Group commit поддерживается
- Periodic checkpoint
  - Может работать без большой просадки
- Может сбрасывать лог транзакций для внутренних нужд

MyRocks durability

- XA & group commit
- WAL для операций над memtable
- Архивация после сброса на диск memtable
- WAL – набор записей без упаковки и сжатия
- Каждая 32КБ и CRC

MyRocks durability

- Несколько моделей восстановления
  - Не стартовать при последней битой записи в WAL
  - (умолчание) не стартовать при любом повреждении
  - Накатывать WAL до повреждения
  - Пропустить повреждение

Replication & clusters

- Global Transaction Id:
  - Изменения каждого узла в логе репликации помечены уникальным идентификатором
  - Точное описание состояния системы
- Без GTID состояние тоже точное но найти позицию сложно

Percona XtraDB Cluster

- Galera replication isolation
  - SERIALIZABLE – не поддерживается, но может работать между транзакциями запущенными на одном узле
  - RU, RC, RR – так же как и для InnoDB
  - COMMIT может возвращает ошибку при конфликте (строчка модифицирована разными транзакциями на разных узлах)

Инструментарий

- verbose slow log: InnoDB_trx_id
- Performance schema: какие запросы были в транзакции
- INFORMATION_SCHEMA: innodb_trx, tokudb_trx, rocksdb_trx
  - Текущие транзакции

Percona Live Europe Call for Papers & Registration are Open!

Championing Open Source Databases

- MySQL, MongoDB, Open Source Databases
- Time Series Databases, PostgreSQL, RocksDB
- Developers, Business/Case Studies, Operations
- September 25-27th, 2017
- Radisson Blu Royal Hotel, Dublin, Ireland

Submit Your Proposal by July 17th!
www.percona.com/live/e17

© 2017 Percona