MySQL 101

Андрей Аксенов, AddConf, 2012

ПРО БАЗЫ ДАННЫХ

Ряд шокирующих новостей

- ШН1. select * from table where id=X, mysql, python, **10K rps == 0.01M rps**
- ШН2. hash::find(), C++, **1-10M rps**
- ШН3. **Любые** БД (SQL, NoSQL, WtfQL) – они про **простоту**, а не скорость
- ШН4. Скорость важна «через тоже»; нужна адская => пиши спецхранилку

Про реляционные базы

- Все есть таблица (Table)
- В которой есть строки (Rows)
- В которых есть колонки (Columns)
- Набор которых фиксирован (Schema)

- Откуда простота?
- Изоляция физического от логического

Про уровни

- Физический == как хранить байтики
- Логический == что писать в программе

- Увы, про физический **надо** знать
- Иначе тормоза неизбежны, как закат

Про логический уровень

- Два вида, DDL + DML
- DDL, definition == про схему == CREATE TABLE, ALTER TABLE, CREATE INDEX
- DML, manipulation == про данные == SELECT, INSERT, DELETE, UPDATE

Про всё сложно

- Реализация физуровня это хранилка строк, индексы, оптимизатор, планы запросов, буфера, кеши, логи, репликация, ...
- Детали реализации привязаны и сильно зависят от конкретной БД
- Но, есть и вечные ценности

Про B-tree

- Индексы всегда B-tree — хорошо подходят
- Это спецструктура, которая
  - Хранит **все** пары (key, row_ptr)
  - Умеет быстро «дай все row_ptr, где key=X» (выборка по ключу)
  - Умеет быстро «дай все row_ptr, где key>=X && key<=Y» (выбора по диапазону)
  - Разумно быстро (**не** мгновенно!) обновляется

Про B-tree, пример

- select * from 1_million_rows where id=123 **без** индекса == yyy... :(
- select * from 1_million_rows where id=123 с индексом по **title** == yyy... :(
- select * from 1_million_rows where id=123 с индексом по id == ooo! :)

ПРО ТРАНЗАКЦИИ

Про транзакции

- Транзакция это набор операций над данными, имеющий ряд свойств
- ACID
  - Atomicity
  - Consistency
  - Isolation
  - Durability

Про физику транзакций

- Durability в теории?
  - В идеале, данные НИКОГДА не теряются
  - Ну, в смысле, после COMMIT
  - Ну, в смысле, если в ДЦ не попала молния
- Durability на практике?
  - Реализация, Write Ahead Log
  - Оптимизация, менее жесткие гарантии

ПРО MYSQL

Нет (теперь) никакой ложки

- Ложку украли где-то между 4.0 и 5.0
- Подключаемые движки (pluggable SE)
  - **По-разному** реализующие физику хранения, индексации, выборок
- MySQL это общий код всего остального!
  - Сетевой протокол, разбор SQL, оптимизатор, репликация, итп
- Отчасти рулится конфигом, **/etc/my.cnf**

ПРО MYISAM

Кратко в целом

- Нетранзакционный
- Иногда может потерять данные
- Кеширует только индексы, строки нет
- Так себе масштабируется по ядрам
- Блокирующая вставка в 1 поток
- Как следствие, ТОРМОЗА при write load

Хранилка строк

- Файл /var/lib/mysql/$databaseName/$tableName.MYD
- Строки (грубо говоря) дописываются в конец по мере поступления
- Важно, строки НИКАК не кешируются, каждый раз read() через OS!

Индексы

- Файл /var/lib/mysql/$databaseName/$tableName.MYI
- Кешируются в памяти
- Ключевая директива **key_buffer**

Зачем использовать

- Обычно (в 99%) случаях незачем
- Обычно InnoDB решительно лучше
- Может, сколько-то осмысленно для постоянного дописывания (append) маловажных архивных данных
  - см. сравнительно быстрая вставка

Что и как тюнить про MyISAM

- Если не используется?
  - Скрутить key_buffer вниз (~16-32M)
- Если используется?
  - Скрутить **key_buffer** вверх
  - по размеру горячих (!) *.MYI
  - Иначе ТОРМОЗА

Что и как тюнить про MyISAM

- Скрутить вниз **sort_buffer**, **read_buffer**, **read_rnd_buffer**, если вдруг «настроили»
- Дефолтные небольшие значения обычно хорошо работают
- Эти буфера аллокаются на каждый тред! 32M sort_buffer * 100 **max_connections** = минус 3.2 GB зазря

ПРО INNODB

Кратко в целом

- Транзакционный
- Данные не теряет
- Кеширует и индексы, и строки
- Получше масштабируется по ядрам
- Блокировки и вставок, и чтений на уровне отдельных строк, а не всей таблицы

Хранилка строк + индексов

- Все хранится вместе, в т.н. tablespace
- Все кешируется вместе, ключевая директива **innodb_buffer_pool_size**
- Хранится страничками по 8 KB, весь tablespace I/O вроде тоже страничками

Хранилка строк + индексов

- По умолчанию, все таблицы всех баз (!!!) кладут все данные в один файл /var/lib/mysql/ibdata1, который никогда не уменьшается (адъ!)
- При наличии **innodb_file_per_table** все лежит в /var/lib/mysql/$databaseName/ $tablename.ibdМясо

Зачем использовать

- Наиболее внятный движок в (стоковом) MySQL
  - Быстрый
  - Надежный
  - Транзакционный
  - Масштабируемый

Что и как тюнить

- Если не используется
  - скрутить **innodb_buffer_pool_size** вниз
- Если используется
  - все несколько длиннее :)
  - плюс ряд дефолтов удивляет

Опасные умолчания

- **innodb_file_per_table** нету, диск утекает
- **innodb_buffer_pool_size = 32M**
  - Адово мало, надо 128M ... 128G
- **innodb_flush_log_at_trx_commit = 0**
  - Адово медленно (если не BBU), надо 2
- **innodb_log_file_size = 5M**
  - Тупо маловато, надо 16M ... 1G

innodb_file_per_table

- Без него – неконтролируемый рост ibdata1
- На свежей копии нужно **сразу** включать
- На существующей копии можно сменить — но процесс нууудный
  - Dump, Unlink, Change, Import...

innodb_buffer_pool_size

- Мега Кеш Всего!
  - И данных
  - И индексов
  - И даже data dictionary
- Крутить вверх до упора, чо (~80%)

innodb_flush_log_at_trx_commit

- Все пишется в WAL, ибо durability
- Лог это что? Страховка, причем 2 видов!
  - Креш демона, креш машины
  - 0, fflush txn + fsync txn
  - 1, fflush 1 sec + fsync 1 sec
  - 2, fflsuh txn + fsync 1 sec
  - 0 для a) RAID b) BBU c) дрова d) **живое** BBU :)
  - 2 для обычных деревенских пареньков

innodb_log_file_size

- Все пишется в WAL, ибо durability
- Бесконечно писать нельзя
- По пределу размера флаш грязных страниц
- То. чем больше лог, тем реже запись!
- Поэтому...

innodb_log_file_size

- При RO нагрузке неважно совсем
- При RW нагрузке важно
- При WO нагрузке (импорт) адово важно
- 128M .. 1G ok, более 2G нельзя
- Более innodb_buffer_pool_size/2 тупо нет смысла

итого

О чем говорил иностранец

- Оно внутри устроено — вот так
- Оно из коробки настроено — вот так
- Надо из коробки настраивать — не так!
- Надо каждому — в т.ч. разработчикам
- Потому что скорость разработки
- Потому что тестирование во внятной среде

ВСЕ!