

# Настройка MySQL под высокие нагрузки

---

Владимир Федорков  
DevConf, Москва 2017

# Правила поведения

---

- Скучать, сидя в аудитории из вежливости
- Мешать другим
- Пить чай и кофе
- Снисходительно посматривать на докладчика
- Заходить, выходить
- Задавать вопросы в любое время

# О докладчике

---

- Занимаюсь вытягиванием проектов из сложных жизненных ситуаций
  - В высоконагруженных проектах простых не бывает
- Специализируюсь на MySQL
  - И полнотекстовом поиске
- Работаю с разными клиентами по всему миру
- Все имена вымышлены, все совпадения случайны

# Три стадии развития цивилизации

---

- Что?
- Как?
- Где?

# Что тюнить? Куда смотреть?

---

- Железо
- Настройки OS
- Настройки MySQL
- Настройки Storage engines
  - InnoDB, MyRocks, MyISAM
- Запросы

# Вторая стадия: Как?

---

- Какие настройки использовать?
- Какие индексы делать?
- Как проектировать базу?
- Как потом ее менять?

# Где: определяем цели

---

- Глобальный вопрос всего проекта
  - Масштабирование
  - Расходы
  - Перспектива роста
- Регион
- Платформа
  - «Чистое» железо
  - Виртуализация
  - Облако

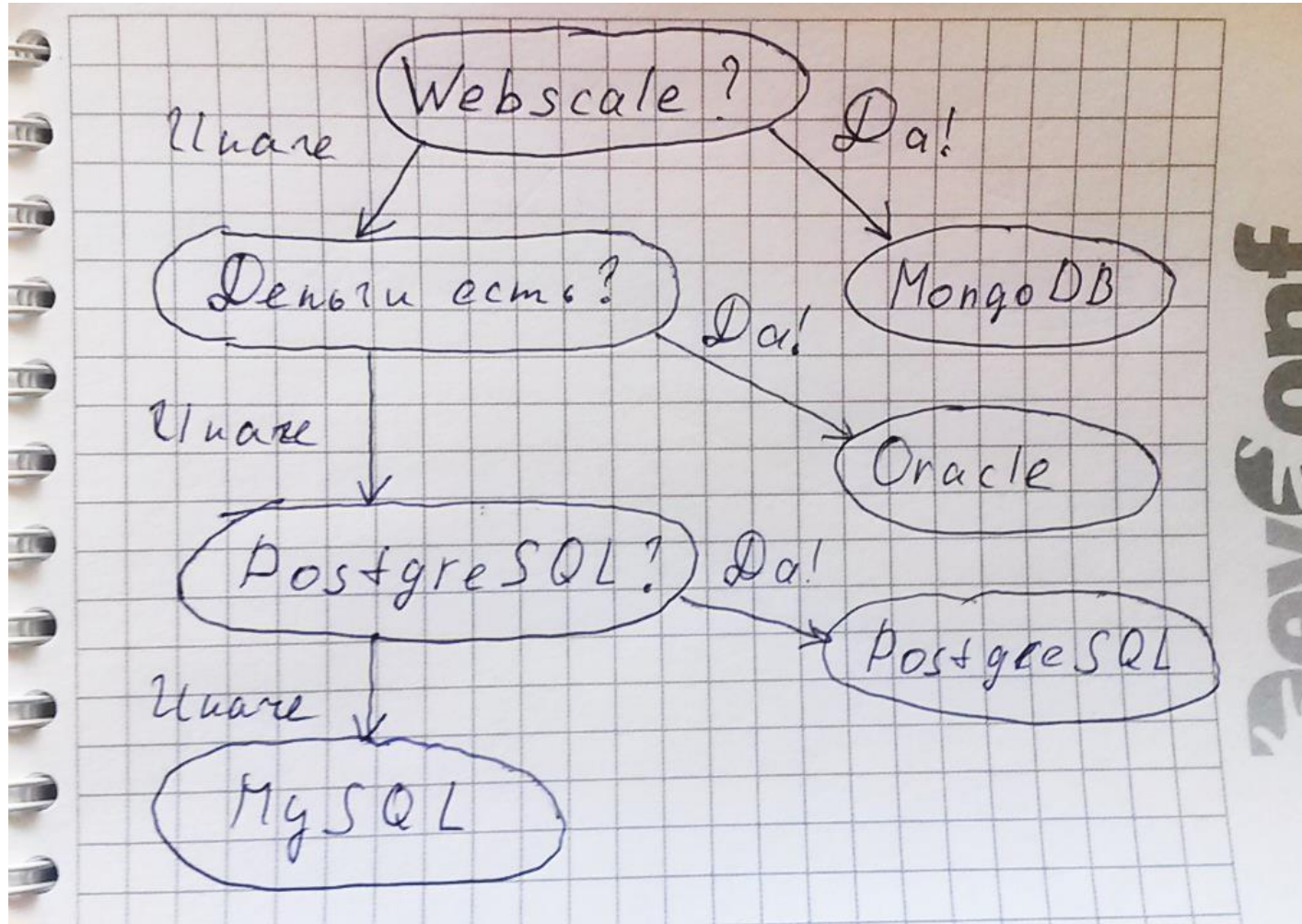
# Нужна ли база вообще?

---

- Зависит от того:
  - Как данные читаются
  - Как данные записываются
  - Какие выборки используются
  - Как данные масштабируются
  - Насколько данные ценны
  - Какая нужна скорость доступа
  - Насколько важна отказоустойчивость
  - Насколько значима безопасность
  - Есть ли географическая распределенность



# Почему мы используем MySQL?



# Что такое MySQL?

---

- Задуман и сделан как реляционная БД
- Придуман давно
- Умеет ACID

# ACID как это есть

---

- A – Atomicity
- C – Consistency
- I – Isolation
- D – Durability

# ACID как мы это понимаем

---

- A – Все или ничего. Даже если отключат свет.
- C – Данные логически непротиворечивы. Всегда.
- I – Изменения не видны пока не закончены.
- D – Записанное можно гарантированно прочитать.

# Когда ACID может быть плохим?

---



# Дополнительные затраты

---

- Сохранение корректного состояния в любой момент времени.
  - Блокировки
  - Мультиверсионность
- Восстановление состояния после отказов.
  - Сети
  - Дисков
  - Питания

# Зачем нужен такой тюнинг?

---



# Там, где данные критичны

---

- Финансовые приложения
- Торговля (магазины и склады)
- Учебные заведения
- Хранилища документов
- ... назовите сами



# Где не надо?

---



# Web 2.0. Весь.

---

- Никому не интересны посты Васи Пупкина
  - Кроме самого Васи Пупкина
    - В 95% случаях
- Дешевле вернуть ошибку
  - Вася Пупкин запостит 8915-го котика заново
- В крайнем случае восстановим из бекапа
  - Если он вдруг есть
- Или переключим чтение на другой датацентр
  - Если вы фейсбук

# Как с ЭТИМ жить?

---

- Конфигурировать
- Настраивать окружение
- Запускать и тюнить запросы
- Масштабироваться
- Использовать облако
  - или не использовать

# Конфигурация MySQL, шаг первый

---

- Конфигурация MySQL
  - 5% настроек обеспечивают 95% производительности
- Что тюнить
  - Буферы / Память
  - Тонкие настройки storage engine
  - Кеши
  - Сеть
  - Репликация

# Память

---

- innodb\_buffer\_pool\_size / key\_buffer
- max\_connections / max\_user\_connections
- Временные таблицы
  - max\_heap\_table\_size / tmp\_table\_size
- Трогаем только в крайнем случае:
  - sort\_buffer\_size, read\_buffer\_size, read\_rnd\_buffer\_size, thread\_stack
- Кэши
  - thread\_cache
  - query\_cache\_size

# InnoDB

---

- `innodb_buffer_pool_instances`
- `innodb_buffer_pool_size` Gb
- `innodb_log_file_size` Mb/Gb
- `innodb_flush_log_at_trx_commit` (0|1|2)
- `innodb_file_per_table`
- `innodb_flush_method=O_DIRECT`
- `innodb_log_buffer_size` Mb
- `innodb_thread_concurrency` (threads)

# Сеть

---

- `skip_name_resolv`
- `max_connect_errors`
  - FLUSH HOSTS
- `max_allowed_packet`
- `back_log`
- Тайм-ауты
  - `interactive_timeout`, `wait_timeout`,  
`connect_timeout`, net read/write timeouts

# Диски / IO

---

- innodb\_read\_io\_threads (Count)
- innodb\_write\_io\_threads (Count)
- innodb\_io\_capacity (IOPs)
- innodb\_flush\_log\_at\_trx\_commit



# Репликация: мастер

---

- `log_bin`
  - Для point-in-time recovery
  - Иногда желательно на другой диск
- `expire_logs_days`
- `max_binlog_size`
- `sync_binlog`
  - Если некуда девать производительность IO
  - Или если потеря данных критична
- `binlog_format` (Statement | ROW | Mixed)

# Репликация: общие настройки

---

- Формат и безопасность
  - `log_slave_updates`
  - `log_bin_trust_function_creators`
- Настройки реплики
  - `read_only`
  - `skip_slave_start`

# Тест «насколько админу скучно»

---

- `max_join_size` – два балла
- `sort_buffer_size` – пять баллов
- `tx_isolation` – десять баллов
- Правило 5/95 работает безукоризненно

# Ура, сконфигурили!

---

- Правильные настройки
  - не мешают работе MySQL
  - обеспечивают отказоустойчивость
    - только на уровне операционной системы!
  - не обеспечивают его производительности
- Допустим все работает

# Запросы

---

- Обеспечивают 99,99% тормозов
  - И головную боль админов
- Лучший MySQL запрос тот который до MySQL не дошел.
  - MySQL не сможет выполнить некоторые запросы быстро
  - Просто потому что так спроектирован

# А что с ним не так, турецкий?

---

- Ограничение дизайна MySQL
  - ACID медленный!
  - B-Tree работает только для спец. случаев
  - Запрос обрабатывается одним ядром
  - Есть Full-text индексы, но они медленные
  - Про репликацию можно говорить часами
    - Что и делаем на собеседованиях
  - Если вышеперечисленное для вас большая проблема – вы используете неправильный инструмент. Вам нужен не MySQL.
    - Если вы не FaceBook.

# Как написать тормозной запрос?

---

SELECT \* FROM table ...

- WHERE DAY(FROM\_UNIXTIME(`ts`)) = 205
- WHERE deleted != 1
- WHERE id NOT IN (1,2,3,...,10)
- WHERE url LIKE '%чтоототам%'
  - Не путать с LIKE 'чтоототам%' !
- ORDER BY RAND()

# Чем плох full table scan?

---

- Индексы не помогают!
  - Значение функции считается для всех строк
  - B-Tree не эффективен
- Читать всю таблицу долго
- Пока читаем, вымываем память базы ненужными данными
- Как бороться?
  - Пересматривать логику запросов
  - Использовать предварительно агрегированные данные вместо расчета функций «на лету»
  - Использовать `deleted = 0` вместо `deleted != 1`



# Высокоселективные запросы

---

- `SELECT * / COUNT(*)`  
`FROM users WHERE sex='female'`
  - Здравствуй, половина таблицы
- Как бороться?
  - Читаем только то, что показываем
    - Используем `LIMIT`
  - Делаем агрегацию для счетчиков
  - Кешируем все, что можем
    - Не в MySQL query cache!
  - Используем внешние инструменты (Mongo, Sphinx, etc)

# Временные таблицы

---

- Плохо
  - Если попадает на диск – очень плохо!
    - Если есть поля TEXT или BLOB, на диск таблица попадет
- Когда создаются
  - GROUP BY
  - Подзапросы
  - DISTINCT + ORDER BY

# Что делать?

---

- Тюнить буфферы временных таблиц
  - tmp\_table\_size
  - max\_heap\_table\_size
- Запускать «тяжелые» запросы на отдельной реплике
  - Например, для генерации отчетов

# Если нагрузка большая

---

- Query cache
- Thread cache
- Table cache
- Slow query log
- wait\_timeout
- connection pooling
- Репликация и шардинг

# От сервера кластеру

---

- Один сервер мало, два – плохо.
  - Репликация медленная
    - Работает в один поток (исправили в 5.6)
  - Репликация хрупкая (не исправили в 5.6)
    - Но можно использовать GTID
  - Доверия репликации нет
    - Консистентность данных требует проверок
      - Даже если видимых сбоев не было
      - `pt-table-checksum + pt-table-sync`
- Есть варианты
  - Tungsten и Galera

# Прелести кластера

---

- Если у тебя упал единственный сервер, это трагедия
  - А если один из десяти?
- Можно использовать реплики для бекапа
- Можно балансировать нагрузку
  - Можно (нужно) сделать реплику для генерации «тяжелых» отчетов

# Что еще умеет кластер

---

- Встать колом весь от вовремя запущенного ALTER TABLE
- Эффектно среплицировать команду DROP DATABASE
- С разной скоростью выполнять одинаковые запросы на разных нодах

# Ежедневная рутина

---

- Отказы железа
- Баги софта
- Надежные бекапы и восстановление
- Мониторинг
- Апгреды и настройка репликации
- Настройка сети и безопасности



# Что дальше?

---

- Зафолловить twitter @vfedorkov
- Посмотреть другие доклады на [astellar.com](http://astellar.com)
- Прочитать книги
  - “High Performance MySQL”
  - “MySQL Troubleshooting”

# ВОПРОСЫ!

**СПАСИБО!**