Настройка MySQL под высокие нагрузки

Владимир Федорков

http://twitter.com/vfedorkov
http://astellar.com/

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
  - У высоконагруженных проектов простых не бывает
- Специализируюсь на MySQL
  - И полнотекстовом поиске
- Работаю с разными клиентами по всему миру
  - Рассказываю о том, что знаю на конференциях
- 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?

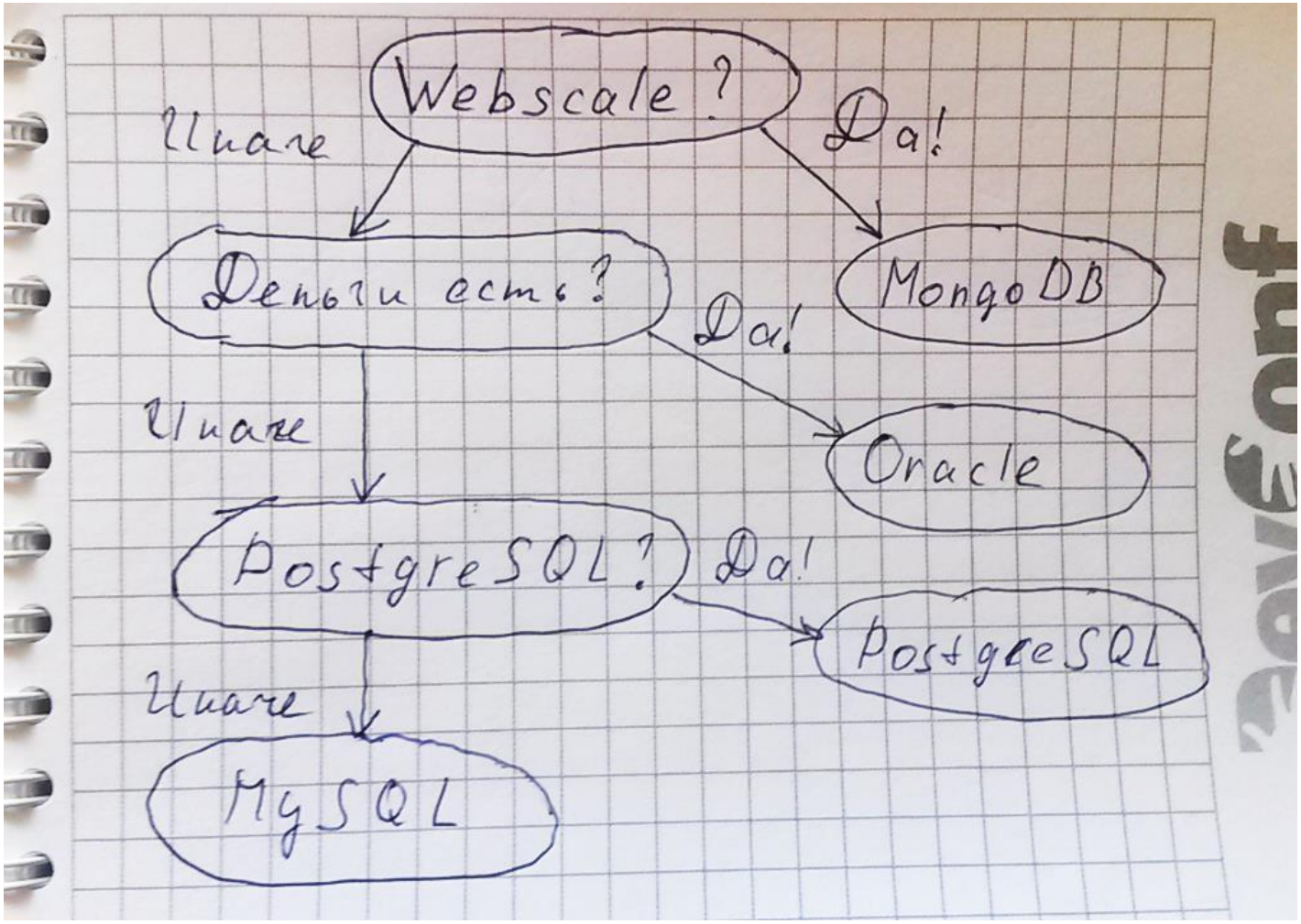
Где: определяем цели

- 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?



А где MySQL вообще работает?

- FaceBook, Twitter и несколько других малоизвестных компаний
- Сотни тысяч запросов в секунду
  - На тех инсталляциях, с которыми работаю лично я
- Наверное работает у кого-то из вас?
- Почему выбирают именно MySQL?

Что такое MySQL?

- Задуман и сделан как реляционная БД
- Открытые исходники
- Многопоточный
- Много документации
- Много людей которые знают «Как»
- 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

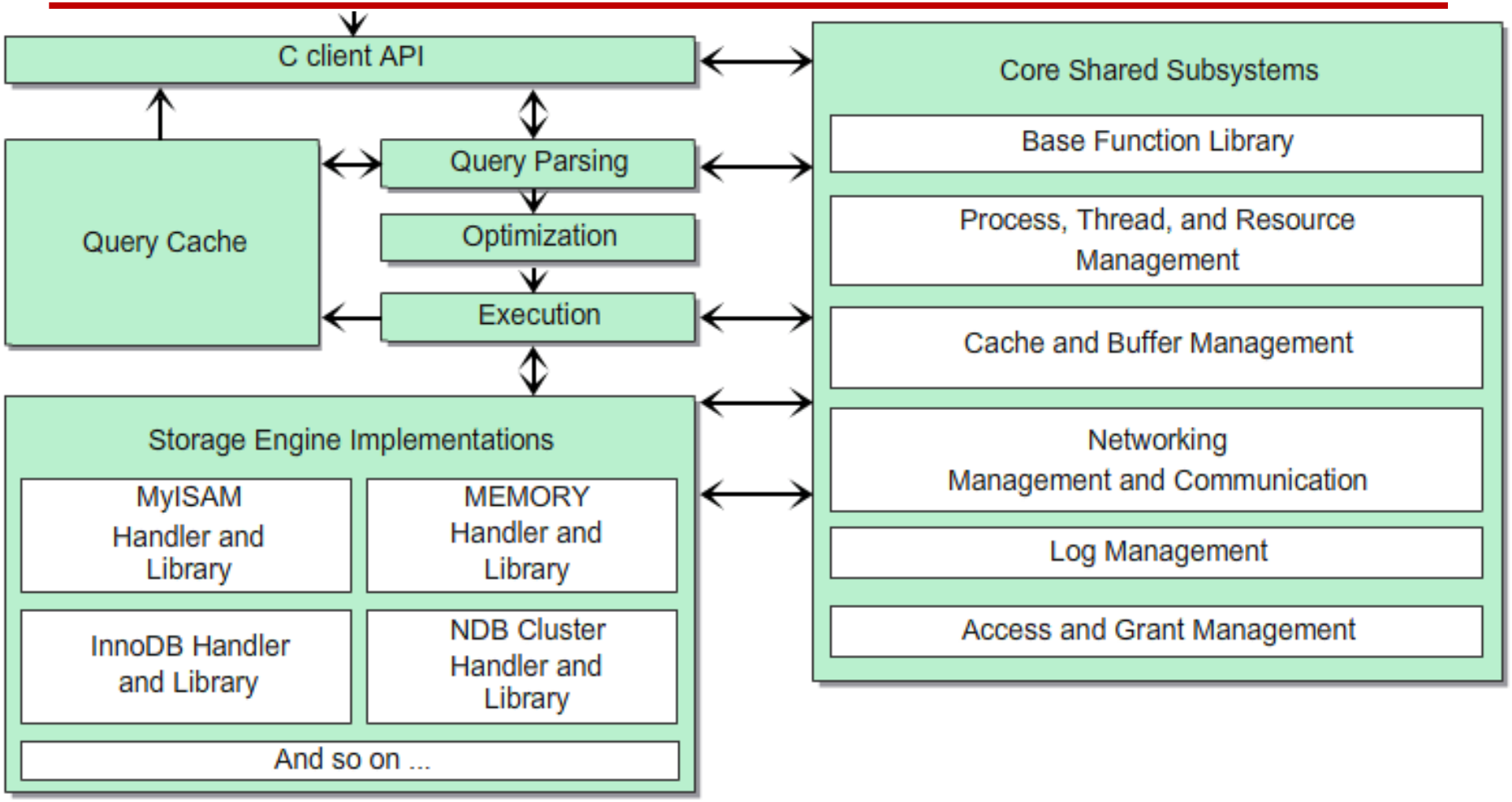
Как с этим жить?

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

Если все-таки угораздило

- Какой Storage Engine выбрать?
- Сколько серверов поставить?
  - А может в облако?!
- Какое железо выбрать?
- И как это вообще все работает?!!!

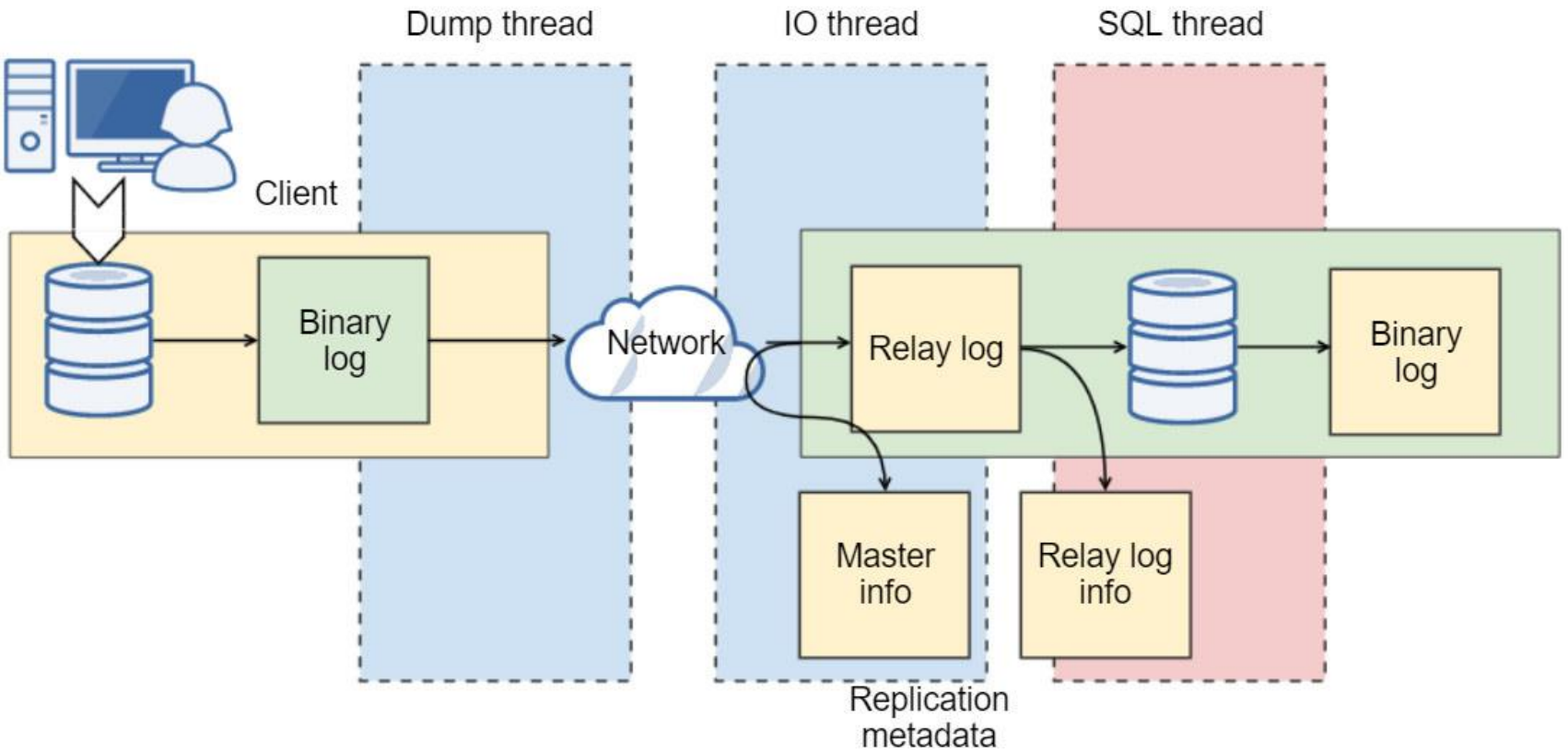
Как это вообще работает?



Бинлоги

- Все изменения можно писать в отдельный файл
  - Для того, что передать на другую ноду (репликация)
  - Для того, что бы «догнать» данные восстановленные из бекапа
  - Для того что бы найти кто и когда сделал DROP DATABASE

Репликация


Dump thread
IO thread
SQL thread
Client
Binary log
Network
Relay log
Binary log
Master info
Relay log info
Replication metadata

Какой Storage Engine выбрать?

- InnoDB
- TokuDB
- MyRocks
- Все три:
  - Транзакнционные
  - Мультиверсионные
  - ACID

Какой Storage Engine выбрать?

- InnoDB
- TokuDB
- MyRocks
  - Write and space optimized
  - Great compression support
  - Good for SSD
  - LSM Tree
- Storage engines details
  courtesy of Sveta Smirnova / Percona

LSM Tree

- Write and space optimized
  - Options for bulk operations
  - Compression
- All writes go to MemTable and WAL first
- Data files are immutable
- Compaction
- Designed for small transactions

RocksDB limitations

- Two transaction isolation levels
  - READ COMMITTED
  - RPEATABLE READ
- No gap locking
- No support for
  - Foreigh Keys
  - Full Text Keys
  - Spatial Keys

TokuDB

- Fractal Tree
- Space optimized
- Optimizations for reads
  - Secondary Clustered Indexes
- Optimizations for writes
  - Fast inserts
  - Bulk loader
  - Compression

MyISAM

- Блокировки на уровне таблиц
- Нет поддержки транзакций
- Сортированный Primary Key
- Быстрое чтение
  - И выборка DESC/ASC по первичному ключу

Archive

- Нет
  - B-Tree индексов
  - Транзакций
  - Foreign Keys
  - DELETE / UPDATE
- Есть
  - SELECT (Full table scan)
  - INSERT / REPLACE
  - Компрессия

Другие

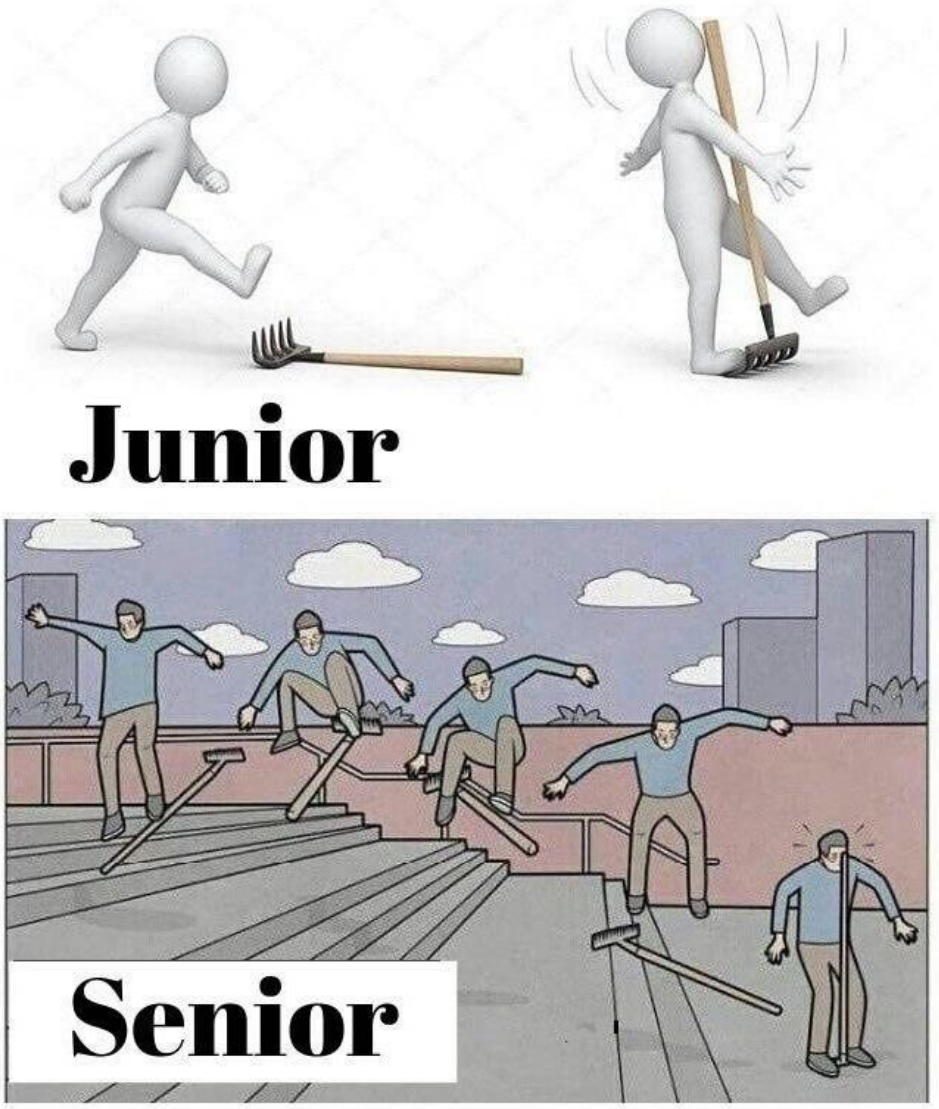
- Memory
- CSV
- Blackhole
- Merge
- Federated
- Example
- NDB Cluster?
- InnoDB?

InnoDB

- Подавляющее большинство инсталляций
- Немного истории
- Выпьем кофе и приступим к высоким материям!
- Потому что …

Знания – Сила!!!

- Перерыв 15 минут
- Потом InnoDB
- Потом железо
- И немного облаков



Конфигурация MySQL

Серверная конфигурация

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
- Кеши
  - 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 безукоризнено

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 ‘%чтототам%’
  - Не путать с LIKE ‘чтототам%’ !
- 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 аггрегированные 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, Galera (XtraDB Cluster), Group Replication

Прелести кластера

- Если у тебя упал единственный сервер, это трагедия
  - А если один из десяти?
  - Сколько их надо-то?!
- 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

- Можно ли это все автоматизировать?

От кластера к облаку

- На примере Amazon RDS
  - Relational database service
- MySQL, Oracle & MSSQL
  - MySQL (InnoDB) и аврора
- Что позволяет?
  - Создавать/удалять ноды и реплики
  - Выделять реплики из кластера
  - Автоматизировать развертывание БД и фронтов
    - Вплодь до полного скриптования

Что это значит для нас?

- Полностью автоматизированное развертывание приложение с использованием RDS и EC2
- Гибкий контроль производительности и стоимости с помощью добавления и удаления машин
  - В зависимости от времени дня
  - В зависимости от текущей нагрузки
- Изменение параметров кластера на лету

Что сделать не получиться?

- Зайти на RDS инстанс по SSH
- Починить репликацию
- Сделать бекап с помощью xtrabackup
  - Только mysqldump, только хардкор

Что мы можем контролировать?

- Выбирать регион для инстанса
- Конфигурировать MySQL
- Выбирать тип инстанса
- Открывать/закрывать доступ по сети
- Выбирать параметры дисковой подсистемы

Регионы и availability zones

- US
  - East 1 (Northern Virginia)
  - West 1 (Northern California)
  - West 2 (Oregon)
- EU: Ireland, Frankfurt
- China: Beijing
- Asia Pacific: Singapore, Tokyo, Sydney
- South America: São Paulo
- И другие

Availability zone

- В каждом регионе нескольк AZ
- Можно сделать AZ мастер
  - Поможет в случае креша основного мастера
    - Репликация может быть сломана

Storage

- SSD & Magnetic диски
  - Полностью разные архитектурно и физически
- Вы в облаке

Настройка MySQL: Parameter groups

- Содержит все возможные настройки MySQL
- Дефолтные настройки не оптимальны
  - Нужно создать свою группу и поменять как надо
  - Число PG не бесконечно
- Некоторые настройки поменять нельзя
- Статические и динамические настройки почти как в «родном» MySQL

Добро пожаловать в облако!

- Железо можеть быть разное
  - Даже на каждом запуске бенчмарков
- IO зависит от сети
- Стоят ограничители CPU & IO
- Ты не знаешь своих соседей

Как жить?

- Всегда ориентируемся на худший случай
- Максимально уменьшить нагрузку на IO
  - Аккуратно выбирать диски
- Агрессивный шардинг наш друг и верный друг и товарищ
- Внимательно следить за
  - нагрузкой на CPU/IO
  - Использованием памяти
  - Вовремя обнаруживать битые реплики

Что есть хорошего?

- Можно смотреть логи
- Точность в логах запросов до микросекунд
- Есть зачаточный мониторинг
- Есть попытки автоматического восстановления репликации

Когда использовать облако?

- Когда не очень много данных
  - Или когда вы можете их зашардить
- Когда сложно предугадывать нагрузку
- Когда быстро нужно много ресурсов
  - Или когда ресурсы нужны ненадолго
- RDS, когда не хватает рук для администрирования

Как с этим всем живут гиганты?

- Строят свои датацентры и каналы
- Содержат команды инженеров и программистов
- Переделывают MySQL под свои нужды
- Меняются патчами и публикуют их
  - Можно их найти и использовать
- Но если Вы не Твиттер, не Фейсбук и не Google, переживать Вам пока рано!

Что дальше?

- Зафолловить twitter @vfedorkov
- Посмотреть другие слайды на http://astellar.com/about/talks/
- Прочитать книги
  - “High Performance MySQL”
  - “MySQL Troubleshooting”

ВОПРОСЫ!

СПАСИБО!