

# MySQL от домашней странички до Facebook

---

Владимир Федорков  
Владивосток 2015  
лекторий Луч-5

# 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
- 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?

# Пережить первую стадию

---

- Придумать и сделать что-то абсолютно новое
- Сделать что говорит дядя
- Грамотно адаптировать зарубежный сервис
- Запустить в интернет уже работающий offline бизнес

# Вторая стадия: Как?

---

- На каком языке писать?
- Какую платформу / фреймворк использовать?
- Где хранить данные?

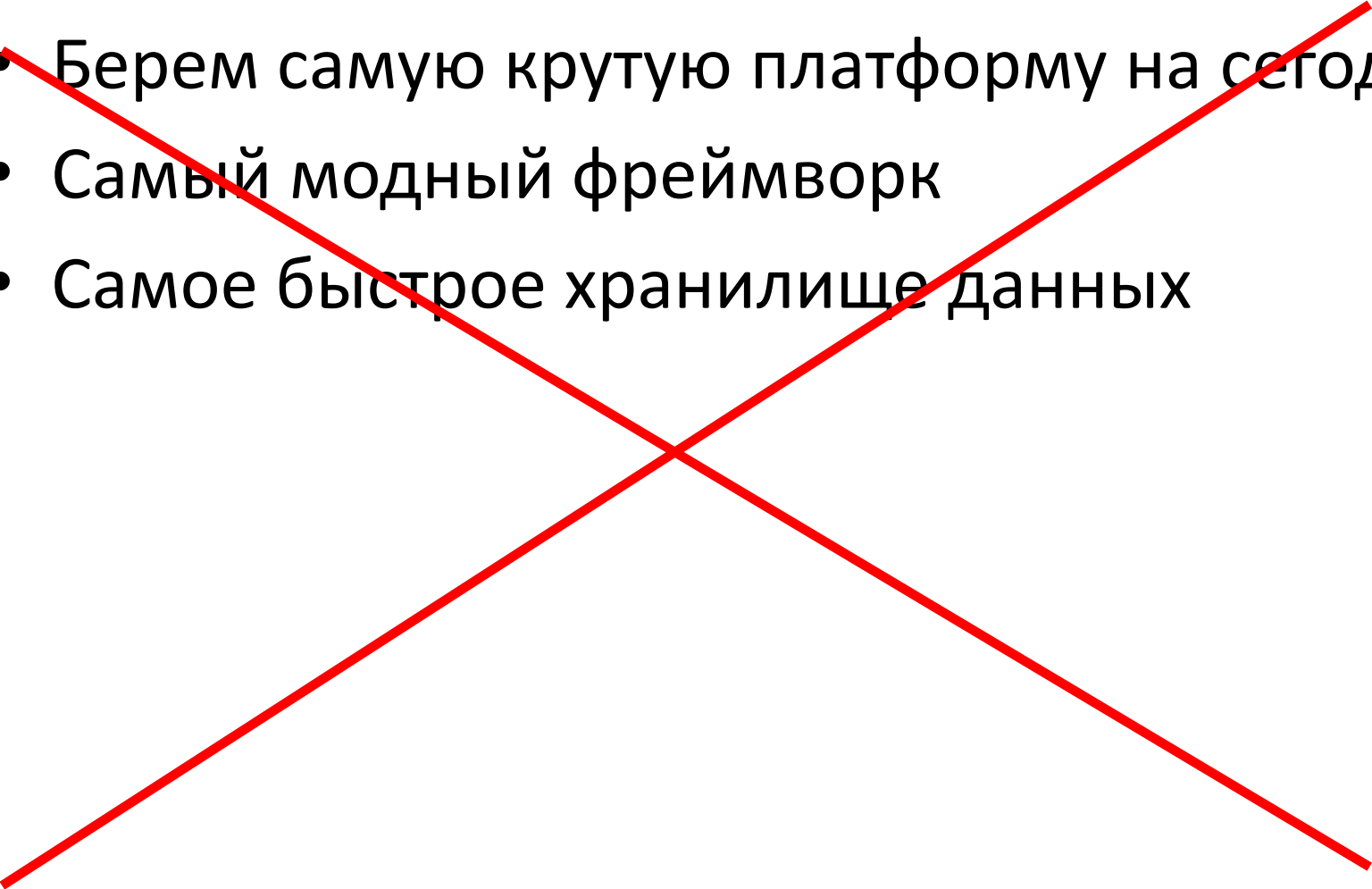
# Да вот так!

---

- Берем самую крутую платформу на сегодня
- Самый модный фреймворк
- Самое быстрое хранилище данных

# Нет, не так!

---

- Берем самую крутую платформу на сегодня
  - Самый модный фреймворк
  - Самое быстрое хранилище данных
- 



# Определяем цели

---

- Научиться использовать крутую технологию?
- Или сделать что-то максимально быстро?

# Смотрим вперед, ставим задачи

---

- Делаем первый релиз максимально быстро
  - можно из глины и соломы
- На известных, хорошо документированных технологиях
- Думаем о том как будем:
  - монетизировать трафик
  - справляться с ростом нагрузки
  - расширять команду
- ... но не очень активно, потому что может не взлететь.

# Решаем простые вопросы.

---

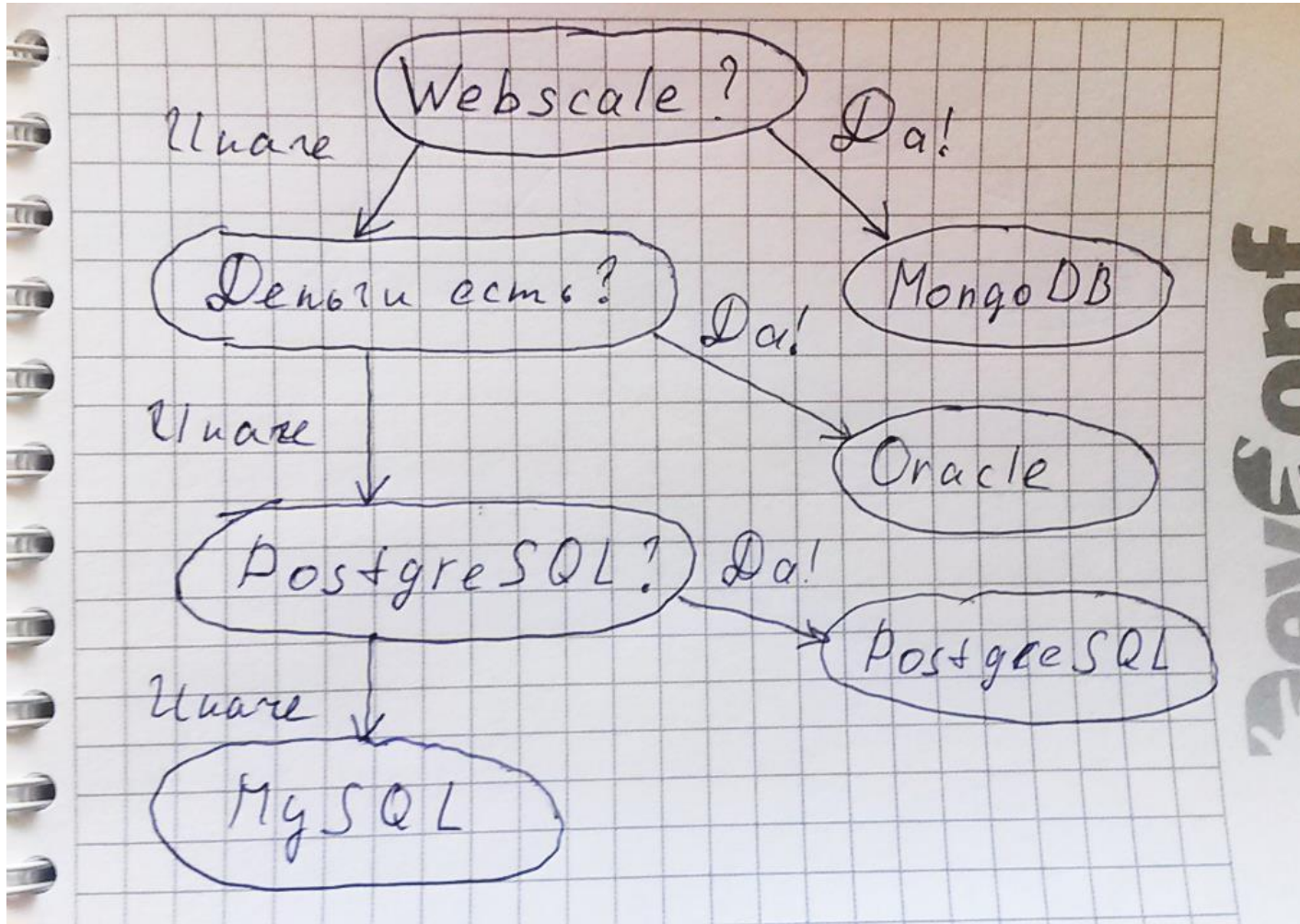
- Фокусируемся на самом больном.
- Как выбрать технологию хранения данных?
- Где хостить проект?
- Облачные вычисления или ржавые железки?

# Где хранить данные?

---

- 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

# Где хранить данные?



# 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

# Почему же все-таки СУБД?

---

- MySQL одна из хорошо знакомых технологий
  - Условно бесплатна
  - Стабильна
  - Хорошо документирована
  - Есть у кого узнать как правильно
  - Есть у кого купить поддержку
  - Это относится и к выбору платформы

# Как с ЭТИМ жить?

---

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



# Конфигурация, шаг первый

---

- Конфигурация MySQL
  - 5% настроек обеспечивают 95% производительности
  - InnoDB наше все, MyISAM уже не наше!
- Что тюнить сразу
  - innodb\_buffer\_pool\_size
  - innodb\_file\_per\_table
  - key\_buffer
  - server-id

# Конфигурация, шаг второй

---

- Сеть
  - skip\_name\_resolv
  - max\_connect\_errors
    - FLUSH HOSTS
  - max\_allowed\_packet
- IO
  - innodb\_flush\_log\_at\_trx\_commit
  - innodb\_log\_file\_size

# Тест «насколько админу скучно»

---

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

# Путь самурайской базы

---

- Готовим MySQL к смерти
  - log\_bin
    - Для point-in-time recovery
    - Желательно на другой диск
  - expire\_logs\_days
  - max\_binlog\_size
  - sync\_binlog
    - Если некуда девать производительность В/В
    - Или если данные очень нужны

# 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

# КАК ПЕРЕЖИТЬ АПОКАЛИПСИС?



Владивосток 2015

[ASTELLAR.COM](http://ASTELLAR.COM)

# 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)
  - 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
- 
- Можно ли это все автоматизировать?

# БЕЛЫЕ ОБЛАКА ПРОТИВ РЖАВОГО ЖЕЛЕЗА



# От кластера к облаку

---

- На примере Amazon RDS
  - Relational database service
- MySQL, Oracle & MSSQL
  - Был MySQL 5.5 и есть 5.6
- Что позволяет?
  - Создавать/удалять ноды и реплики
  - Выделять реплики из кластера
  - Автоматизировать развертывание БД и фронтов
    - Вплоть до полного скриптования

# Что это значит для нас?

---

- Полностью автоматизированное развертывание приложения 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 получится?

---

- 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



# Availability zone

---

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

# Storage

---

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

# RDS Sizes

---

Instance Type	vCPU	Memory (GiB)	PIOPS-Optimized	Network Performance
<b>Standard - current generation</b>				
db.m3.medium	1	3.75	-	Moderate
db.m3.large	2	7.5	-	Moderate
db.m3.xlarge	4	15	Yes	Moderate
db.m3.2xlarge	8	30	Yes	High
<b>Memory optimized - current generation</b>				
db.r3.large	2	15	-	Moderate
db.r3.xlarge	4	30.5	Yes	Moderate
db.r3.2xlarge	8	61	Yes	High
db.r3.4xlarge	16	122	Yes	High
db.r3.8xlarge	32	244	-	10 Gigabit
<b>Burstable performance instances</b>				
db.t2.micro	1	1	-	Low to Moderate
db.t2.small	1	2	-	Low to Moderate
db.t2.medium	2	4	-	Low to Moderate

# 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 может 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
  - Перейти на PIOPS (SSD) где возможно
- 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 посмотреть логи
- Точность в логах запросов до микросекунд
- Есть MySQL 5.6
- 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
- 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?

---

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

# ВОПРОСЫ!

**СПАСИБО!**