



# Не все базы данных одинаково полезны



Аверин Сергей,  
Badoo



# Badoo — это:

- Социальная сеть для знакомств с новыми людьми
- В Top-200 Alexa с 2007 года
- 150+ миллионов зарегистрированных пользователей
- 150+ тысяч новых пользователей в день
- 3+ миллиона фотографий загружаются ежедневно
- 2+ тысячи серверов
- 30+ тысяч запросов в секунду к бекендам
- MySQL, PHP, C(++), Linux, nginx, PHP-fpm, memcache





# 7 советов стартапам





# 1. Масштабирование





# Масштабирование

- Стартап тратит кучу сил и времени на «готовность» к highload, большому масштабированию
- Тратим большие ресурсы без быстрой отдачи
- Сложные вопросы не рассматриваются по причине того, что мало опыта или проблемы еще непонятны



# Масштабирование

- Стартап тратит кучу сил и времени на «готовность» к highload, большому масштабированию
- Тратим большие ресурсы без быстрой отдачи
- Сложные вопросы не рассматриваются по причине того, что мало опыта или проблемы еще непонятны

*На самом деле, это предполагает, что ваши бизнес-метрики тоже вырастут в десятки и сотни раз, а архитектура сохранится*



# Масштабирование



Что имеем





# Масштабирование



Что рассчитываем получить







# Масштабирование



Способ масштабирования



# Масштабирование

- «Серебряной пули» масштабирования нет
- Проблемы будут уникальными для вашего проекта
- Понадобится творческое решение
- И многое придется переделывать



# Масштабирование

- Для стартапа главными ценностями являются быстрый старт и дешевизна изменений
- Начните с простых, быстрых и несложных решений «по рецепту»
- Клиенты → опыт → понимание, какая архитектура нужна



К. О. предупреждает: истиной для 100% случаев не является



## 2. Отказоустойчивость





# Отказоустойчивость

- При проектировании архитектуры проблемы нижних уровней во внимание не принимаются
- Железо, человеческий фактор, внешние риски и т. д.
- Взаимосвязанность сбоев
- В рамках одного сервера на практике не бывает



# Отказоустойчивость

Как это сделано в Бадю, на примере пользовательских данных:

Выделенные БД-серверы

- проверенного вендора
- резервирование по питанию
- RAID 1+0



# Отказоустойчивость

Как это сделано в Баду, на примере пользовательских данных:

Софт

- фаервол
- Persona Server
- разные права доступа
- chroot-окружение



# Отказоустойчивость

Как это сделано в Баду, на примере пользовательских данных:

## Архитектура

- запись в транзакции, на один сервер
- синхронизация с другим ДЦ через общую очередь







### **3. БД с запасом на вырост**





# БД с запасом на вырост

- Выбирается БД без большого запаса фич, которые могут понадобиться в будущем
- Ни один стартап не становился огромным в один день
- Узкоспециализированные БД → теряется гибкость
- NoSQL → нет возможности делать сложные вещи худо-бедно, но ценой малых затрат на кодирование



## 4. БД — хранилище событий





# БД — хранилище событий

Использование БД как хранилища событий чаще всего оправдано только ленью

Распространенные use case'ы:

- события, порожденные транзакциями
- события, которые должны надежно доставляться
- события, которые можно потерять



# БД — хранилище событий

Специализированный движок — RabbitMQ, Kestrel, Scribe, и даже Redis:

- скорость
- простота
- фичи
- масштабируемость



# БД — хранилище событий

В Баду для некоторых задач используем Scribe:

- своя обертка с агрегацией данных, вставкой в БД
- меньше сетевых соединений
- передаем данные между ДЦ
- гибкие настройки
- при сбоях сохраняет данные локально
- очень быстрый



# Старые песни о главном





## 5. Поиск







# Поиск

- Либо быстро, просто, плохо
- Либо используем бесплатный движок — Sphinx, Solr, Lucene/ElasticSearch



# Поиск

99% случаев — быстро, просто, плохо:

```
SELECT `id`, `body` FROM `entries` WHERE `body`  
LIKE '%one%'
```



# Поиск

99% случаев — быстро, просто, плохо:

```
SELECT `id`, `body` FROM `entries` WHERE `body`  
LIKE '%one%'
```

```
SELECT `id`, `body` FROM `entries` WHERE `body`  
RLIKE '[:<:]one[>:]'
```

<http://www.slideshare.net/billkarwin/practical-full-text-search-with-my-sql>



# Поиск

99% случаев — быстро, просто, плохо:

Some people, when confronted with a problem, think  
“I know, I’ll use regular expressions.”

Now they have two problems.

— Jamie Zawinsky



# Поиск

99% случаев — быстро, просто, плохо:

- потом используем MySQL FULLTEXT Index
- для простых решений прекрасно работает обратный индекс
- Но с полноценным поиском по тексту проблема в том, что просто плохо ищет =)
- а также: мало фич, медленно, хуже масштабируется



# Поиск

99% случаев — быстро, просто, плохо:

- а для каких-то задач просто неприменимо

Тест Percona: индекс по всем статьям Википедии.  
2,5 млн записей, 15 Гб текста на одном сервере

- Sphinx: 20 минут
- MySQL: админ уснул через 6 часов, так и не дождавшись

[http://www.percona.com/files//presentations/openssl2008\\_sphinx.pdf](http://www.percona.com/files//presentations/openssl2008_sphinx.pdf)



# Поиск

Используйте специализированный софт:

- проще в разработке
- быстрее
- больше возможностей
- масштабируется
- а главное, лучше ищет



## 6. Сильная consistency







# Сильная consistency

- Не всегда нужна в вебе
- Часто сложно достигается
- Особенно, когда данные в один сервер не помещаются и надо что-то придумывать



# Сильная consistency

- Eventual consistency рулит
- Можно писать в базу выборочно или писать агрегированные данные, не нагружая БД
- Денормализация может дать большой прирост производительности
- Важно знать меру, и что мы теряем, а что получаем



# Сильная consistency

Чтобы не получилось так:

SQL DB = ‘A consistent transactional datastore with schema guarantees that uses relational algebra to access normalized tables.’



# Сильная consistency

Чтобы не получилось так:

+ добавляем slave — репликация



# Сильная consistency

Чтобы не получилось так:

+ добавляем slave — репликация

+ мемкеш



# Сильная consistency

Чтобы не получилось так:

+ добавляем slave — репликация

+ мемкеш

+ добавляем еще slave'ов — репликация репликации



# Сильная consistency

Чтобы не получилось так:

+ добавляем slave — репликация

+ мемкеш

+ добавляем еще slave'ов — репликация репликации

+ шардинг



# Сильная consistency

Чтобы не получилось так:

+ добавляем slave — репликация

+ мемкеш

+ добавляем еще slave'ов — репликация репликации

+ шардинг

+ один столбец на таблицу, храним в нем сериализованный объект





# Сильная consistency

Чтобы не получилось так:

SQL DB = ‘A consistent transactional datastore with schema guarantees that uses relational algebra to access normalized tables.’



# Сильная consistency

Чтобы не получилось так:

SQL DB = ‘A consistent transactional datastore with schema guarantees that uses relational algebra to access normalized tables.’

*Много данных* ↓ *кривые руки*



# Сильная consistency

Чтобы не получилось так:

SQL DB = 'A consistent transactional datastore with schema guarantees that uses relational algebra to access normalized tables.'

*Много данных* ↓ *кривые руки*

~~'A consistent transactional datastore with schema guarantees that uses relational algebra to access normalized tables.'~~  
= datastore with access to data, лучше и не скажешь

<http://www.youtube.com/watch?v=zAbFRiyT3LU>



## **7. Используйте хорошо изученные инструменты**





# Используйте хорошо изученные инструменты

- Неизвестность → опасность
- Выше скорость разработки
- Не поддавайтесь просто так на моду NoSQL



# Используйте хорошо изученные инструменты

“Психологическая” популярность NoSQL:

- marketing hype
- мало знаний в области SQL: ACID, CAP, 3 NF, транзакции
- пытается сделать вид, что БД-специалист не нужен



# Используйте хорошо изученные инструменты

“Психологическая” популярность NoSQL:

Идеальная БД для программиста

- хранит объекты классов приложения (сериализация)
- работает быстро (чтобы можно было похвастаться друзьям)
- обо всем остальном заботится сама



# Используйте хорошо изученные инструменты

“Психологическая” популярность NoSQL:

Выбор БД

- техн. менеджмент спускает вопрос на тормозах, хотя это его задача
- БД выбирает тот самый программист
- Выбираете NoSQL — понимайте, почему вы это делаете



К. О. предупреждает: так бывает далеко не всегда





# Используйте хорошо изученные инструменты

NoSQL:

- запись в один поток
- memory-mapped files, IO scheduling не для БД
- один индекс на запрос
- не очень гибкий шардинг
- производительность тюнится только на уровне ОС
- нет атомарности на уровне одного запроса
- иногда скудный мониторинг, статистика





# Используйте хорошо изученные инструменты

NoSQL:

- зачастую, приходится писать кучу довольно скучного кода на уровне приложения
- + чаще всего быстрее SQL-баз
- + проще развертывать, особенно шардинг
- + нет схемы, ALTER TABLE забыто, как страшный сон



# Используйте хорошо изученные инструменты

SQL:

- медленнее
- сложнее
- (–) много каверзных настроек
- в редких случаях непредсказуемо работает
- (–) позволяет писать медленные/плохие запросы



# Используйте хорошо изученные инструменты

SQL:

- + более популярно, язык у всех на 80% совпадает
- + хорошо изучено, стабильно
- + оптимизировано хранение данных
- + куча рычагов оптимизации
- + constraint'ы, триггеры, хранимые процедуры
- + ACID
- + B-Tree, R-Tree, GIN, GIST, hash-индексы



# Используйте хорошо изученные инструменты

SQL:

- (+) Join'ы, которые зло, но иногда выручают
- + очень навороченный оптимизатор запросов
- + параллельное исполнение (под)запросов
- + многоуровневое кеширование
- + статистика, мониторинг
- + можно писать сложные запросы, не перенося логику в код приложения



# EVERYBODY LIES



Выводов нет, думайте своей головой!



## Вопросы?

**We're hiring!**



Аверин Сергей,  
Badoo

[@averin.ru](https://twitter.com/ryba_xeks)  
[averin.ru/slides/](http://averin.ru/slides/)