MongoDB is a web-scale

Курс «Базы данных»

Антон Волохов

Yandex

7 октября 2013 г.

Содержание

Содержание

Терминология

- **Документ** - Ряд в RDBMS
- **Коллекция** - Таблица в RDBMS
- **База данных** - Базы данных в MySql
- **Пространство имён** - Пара <база данных, коллекция>
- **Нода** - Процесс базы данных, осуществляющий хранение и поиск пользовательских данных на диске
- **Реплика** - Множество **Node**, обладающее полной копией одних и тех же данных
- **Шард** - Единица масштабирования БД

Disclaimer

- Если есть вопрос, меня можно и нужно перебивать
- Не на каждый вопрос я смогу ответить по ходу лекции
- Не на каждый вопрос я смогу ответить

Декларированные цели

- Производительность
- Простота использования
- Горизонтальное масштабирование
- Обработка эволюционирующих данных
- Конкуренция с RDBMS

MongoDB features

- BSON протокол
- JS консоль
- Хранимые процедуры
- Master-slave репликация
- Шардирование без даунтайма
- Система хранения файлов GridFS

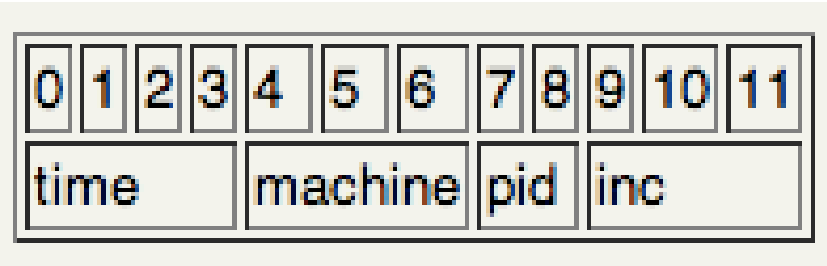
API

- Хранимые процедуры

```
> db.system.js.insert({"_id" : "inc", "value" : function(x)
    {return x+1}})
> db.eval("inc(1)")
2
```

- Индексы
- MapReduce
- Уникальный первичный ключ

```
> db.books.insert({foo:"bar"})
> db.books.find()
{ "_id" : ObjectId("5251664b629201a4d4164ec0"), "foo" : "bar" }
```



Примеры запросов

- Документы

```
isbn = {
_id: "0-321-34960-1",
author: "B. Goetz",
title: "Java Concurrency In Practice",
ISBN :{
"ISBN-10": "0-321-34960-1",
"ISBN-13": "978-0-321-34960-6"
},
"authors-list":["Brian Goetz", "Tim Peierls", "Doug Lea", "Joseph
    Bowbeer", "David Holmes"]
}
```

- Запросы

```
db.test.insert(isbn)
db.test.find({ "title" : /Java/ })
db.test.update(isbn, { $push : { "authors-list" : "Joshua Bloch" }})
db.test.insert({ "foo" : "bar" })
db.test.find({ "title" : { $gte : "Java" }, "title" : { $lt : "Python"
    }})
```

Вложенные документы

```
> db.items.insert({ foo : "foo", "bar" : { "first" : 1, "second" : 1 }})
> db.items.insert({ foo : "foo", "bar" : { "first" : 1, "second" : 2 }})
> db.items.insert({ foo : "foo", "bar" : { "first" : 2, "second" : 1 }})
```

- Запрос на точное совпадение

```
> db.items.find({ bar:{ "first" : 1 , "second" : 1 }})

{ "_id" : ObjectId("525170906292O1a4d4164ec1"), "foo" : "foo",
"bar" : { "first" : 1, "second" : 1 } }
```

- Запрос на диапазон значений

```
> db.items.find({ bar : { $gte : { "first" : 1, "second" : "" }},
"bar" : { $lt : { "first" : 2, "second" : "" }}})

{ "_id" : ObjectId("525170906292O1a4d4164ec1"), "foo" : "foo",
"bar" : { "first" : 1, "second" : 1 } }

{ "_id" : ObjectId("5251709e629201a4d4164ec2"), "foo" : "foo",
"bar" : { "first" : 1, "second" : 2 } }
```

Работа с файловой системой

- Аллокейтим файлы на диске отдельно для каждой базы данных

```
-rw------- 1 mongodb nogroup 16M test.ns
-rw------- 1 mongodb nogroup 64M test.0
-rw------- 1 mongodb nogroup 128M test.1
...
-rw------- 1 mongodb nogroup 2.0G test.5
-rw------- 1 mongodb nogroup 2.0G test.6
```

- Создаём служебную базу данных ($\sim$ 5% дискового пространства)

Кеширование

- Не in-memory database
- Не занимаемся memory management
- Memory-mapped files
- File system cache

Файлы данных

- Экстент
  - Указатели на первую и последнюю запись
  - Положение
  - Длина
  - Пространство имён
- Запись
  - Длина
  - Смещение внутри экстента
  - Смещения предыдущей и последующей записей
  - Документ
  - Padding

В свете этого

Удаление

- Помечаем пространство как свободное
- Не пишем на диск

Добавление

- Лениво
- В первый найденный свободный блок

Обновление

- Документ увеличился?
- Влезает ли обновлённый элемент в свой блок?

Администрирование

Проблемы

1. Фрагментация данных
2. Параллельное изменение одних и тех же файлов
3. Неконтролируемое разрастание файлов данных

Решения

1. compact
2. глобальные блокировки на уровне базы данных
3. repair базы

Резюме

Ни одного адекватного решения

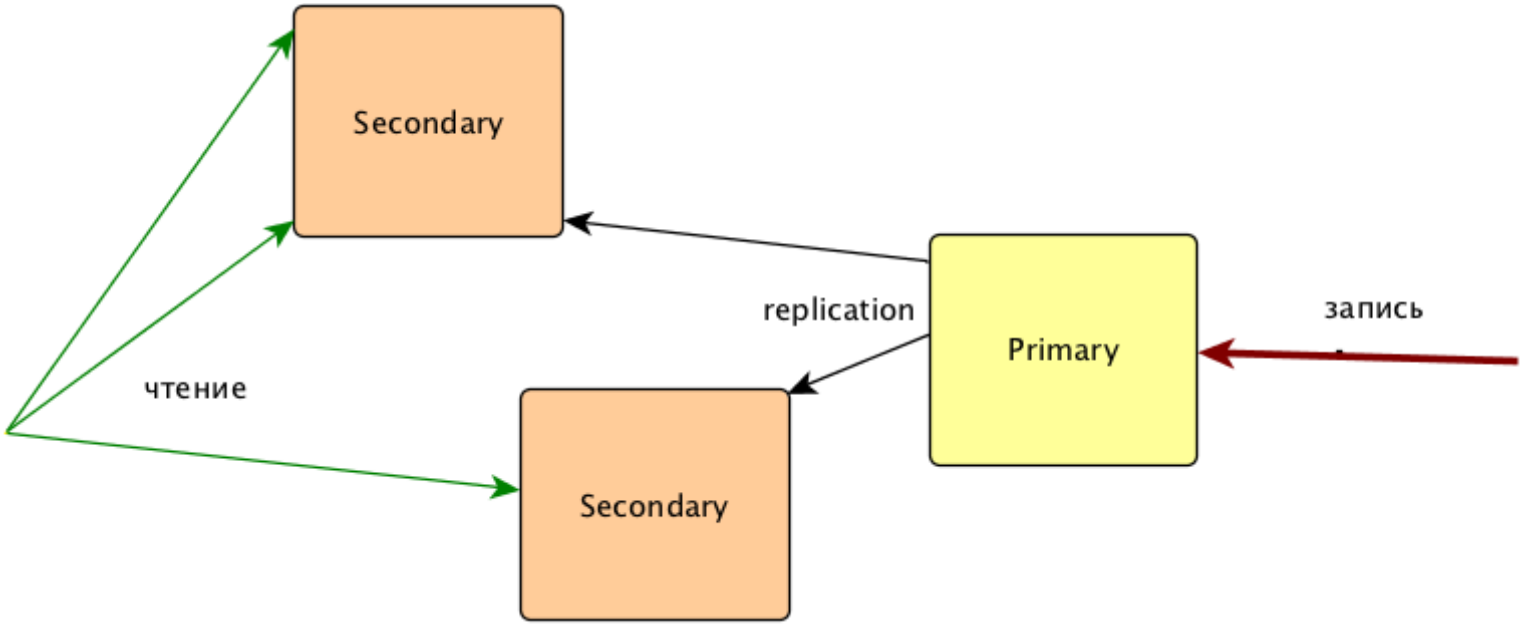
Пространство имён

- Индексы
- Коллекции
- Экстенты
- Статистика

Индексы

- B-Дерево
- Композитный индекс
- По массиву и поддокументу
- Запоминаем оптимальный план запроса
- .explain()

Master - Slave



- Запись только на одну машину
- Чтение со всех машин
- Настраиваемая консистентность

Консистентность

- Не проверяем, удалась ли запись
- Ошибки записи в текущем соединении

getLastError()

- Ждёт, когда W машин запишут все изменения в текущем соединении с момента предыдущего getLastError()
- По-сути, точка синхронизации

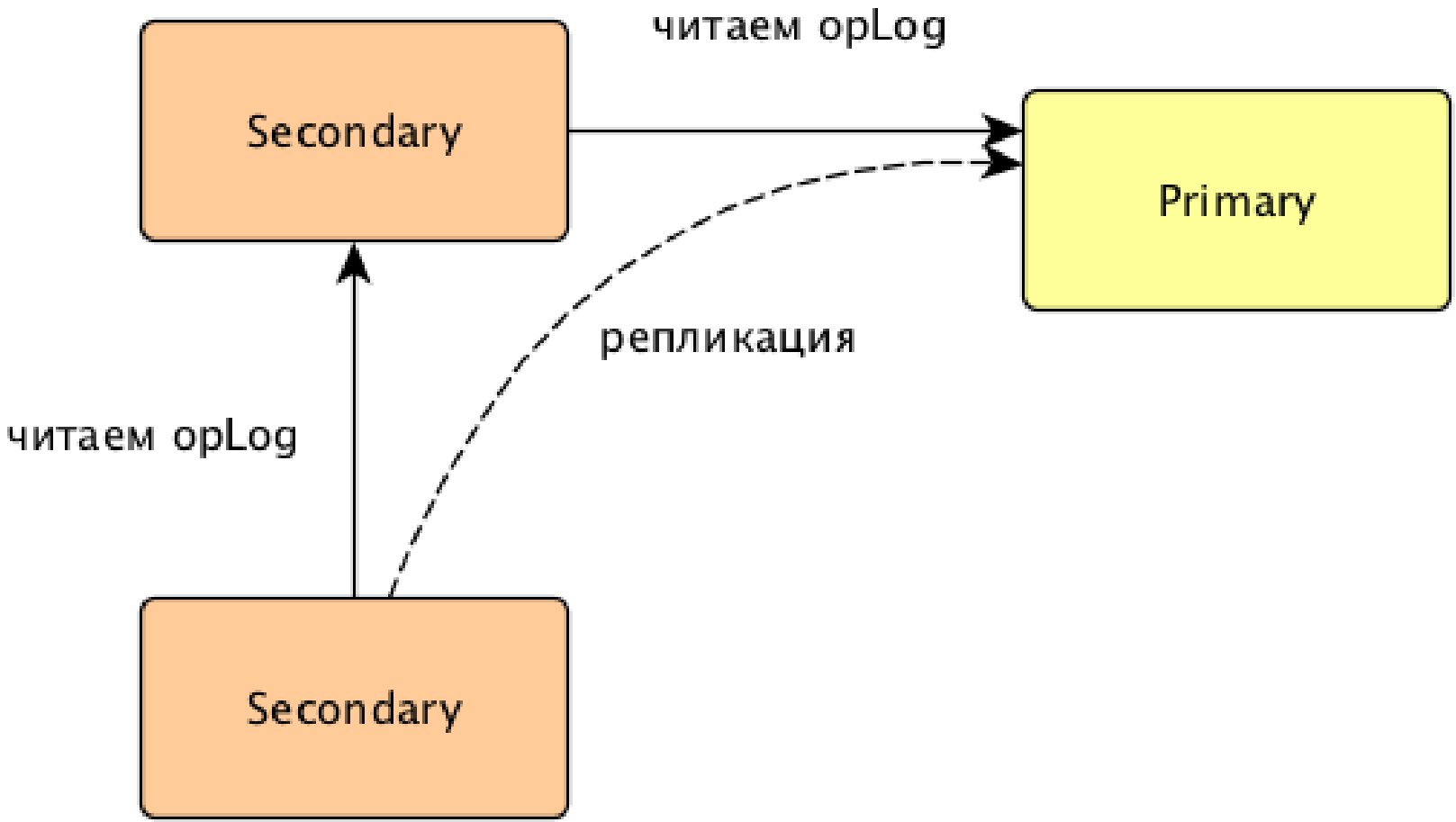
Не верьте бенчмаркам!

Если не вызывать getLastError(), клиент будет писать в монгу, не зная, прошла ли запись. Результат - over9000 rps, при неопределённом числе удачных записей.

Replication internals

- Лог операций
- Простейшие преобразования запросов
- Транзитивность репликации
- Текущий мастер выбирается голосованием

Replication internals: oplog polling



Replication internals: voting

- Все машины реплики знают друг о друге
- Мастер пингует все машины своей реплики
- Если мастер не видит кворум, он снимает с себя полномочия
- Процесс голосования инициирует машина, которая не может найти мастера
- Голосование начинается в случае, если мастер не виден кворумом

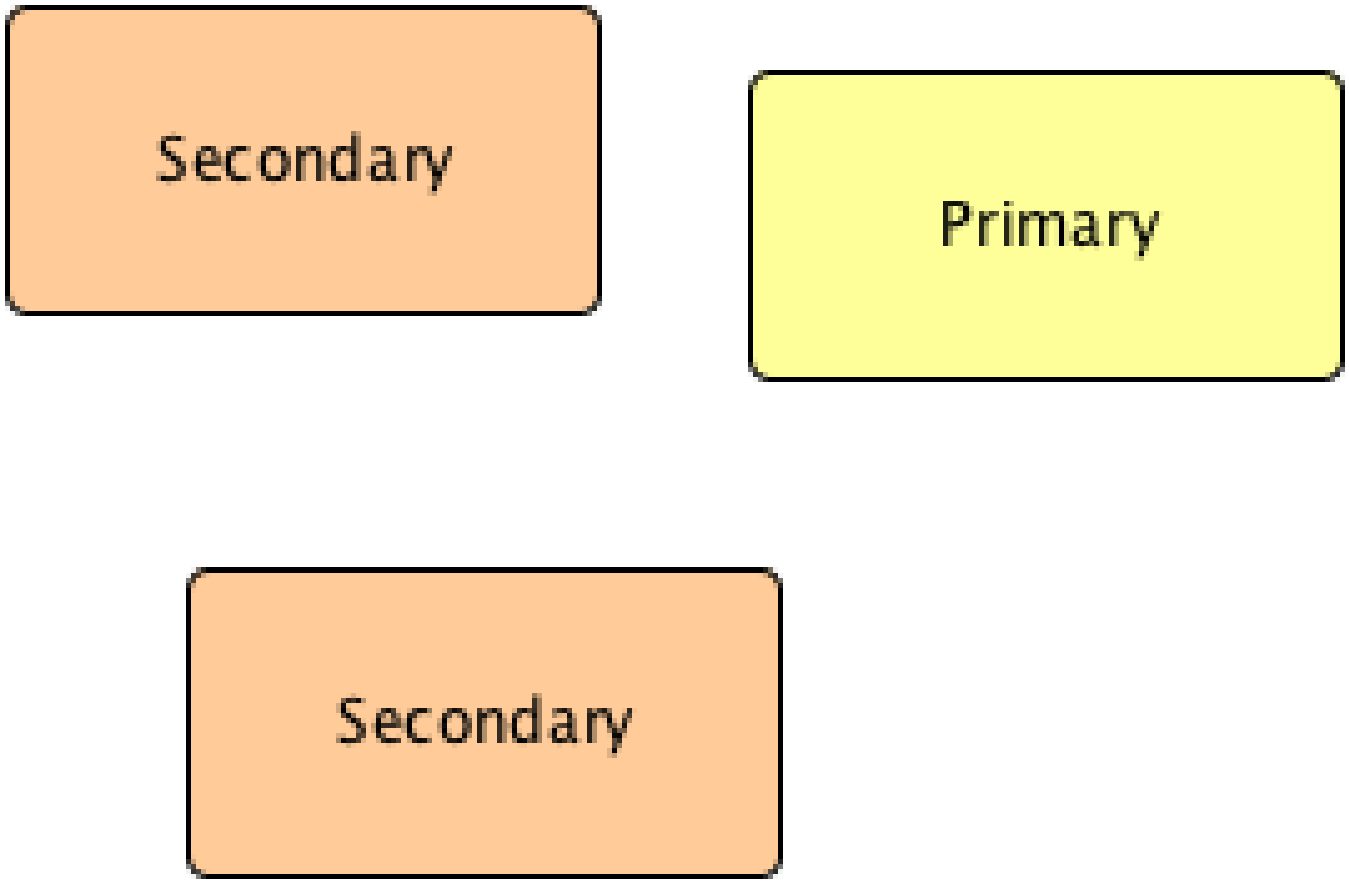
Replication internals: Rollback and Recovery

1. Откатываются все изменения, которых нет на мастере
2. Накатываются все изменения нового мастера

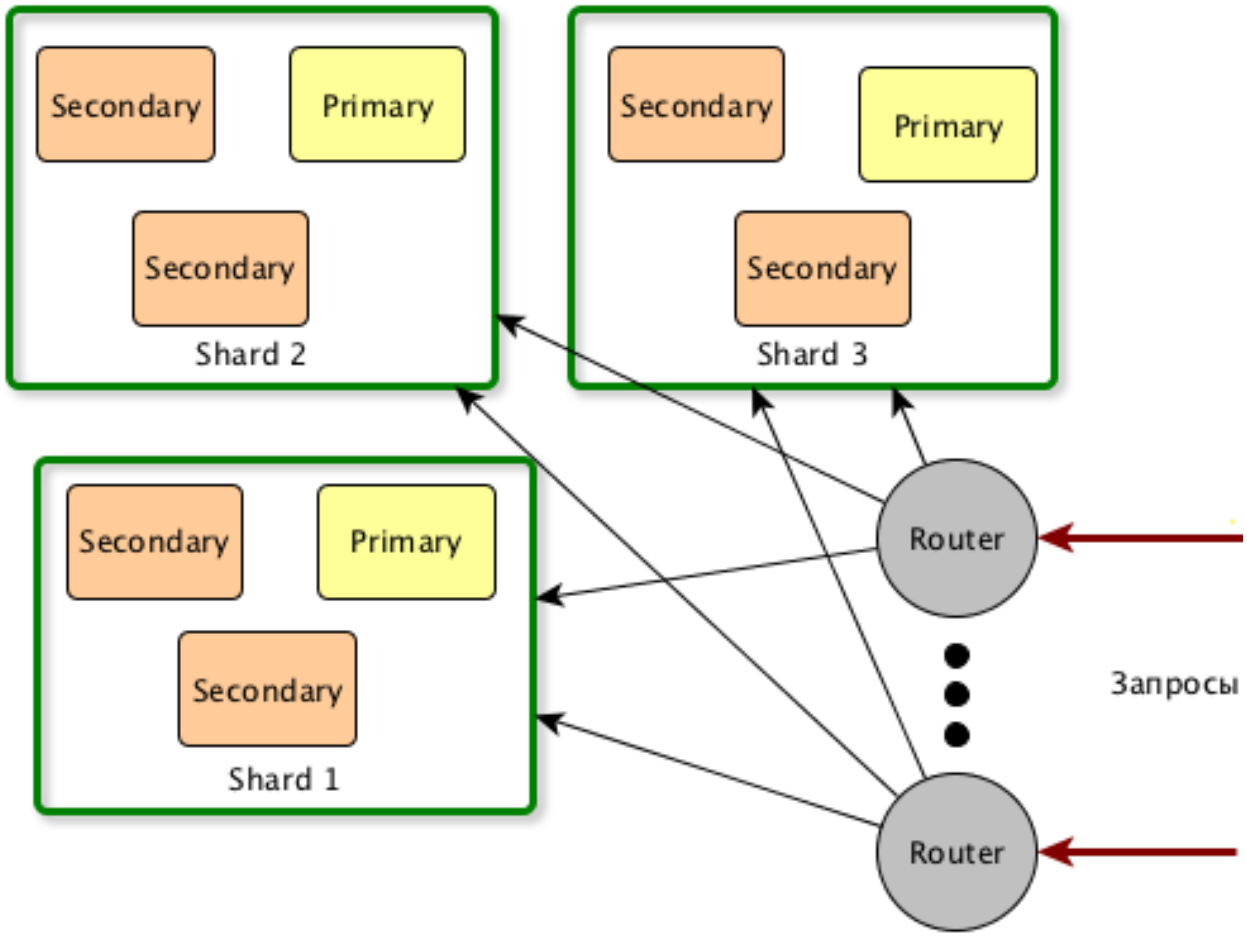
NB!

- Нет автоматического разрешения конфликтов
- Нет внятного оповещения о существовании конфликтов

Overview



Overview


Secondary
Primary
Secondary
Shard 2
Secondary
Primary
Secondary
Shard 3
Secondary
Primary
Secondary
Shard 1
Router
Router
Запросы

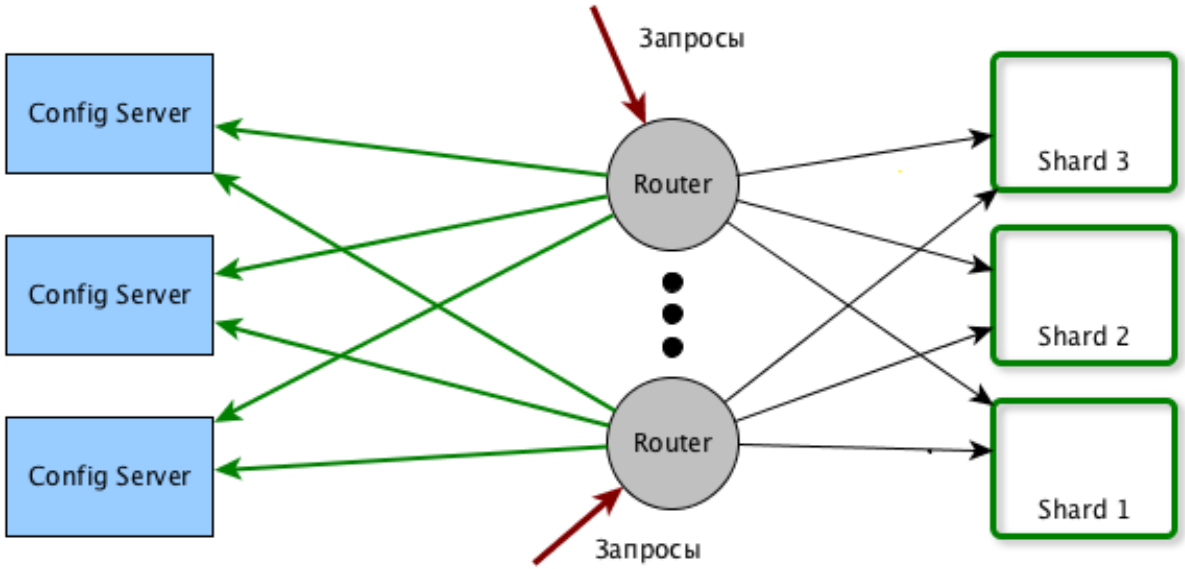
Sharding meta

- Точечные запросы по шардам
- Автоматическая балансировка
- Локализация данных

Sharding meta

А что внутри?

- Где хранить метаданные?
- Как обеспечить надёжность?
- Как сгруппировать данные для шардирования?



Configuration Server

- Нет репликации
- Синхронная запись
- Two phase commit
- Read-Only при отказе одной из машин

Шардируем

1. Объявляем коллекцию как шардированную
2. Выбираем поле, по которому будут строится чанки

Чанк

- Начало диапазона
- Конец диапазона
- Объем
- Принадлежность к шарду

Балансируем

- Периодичный бэкграунд процесс роутера

Workflow

1. Пытаемся взять lock
2. Если удалось, опрашиваем каждое пространство имён
3. Смотрим на распределение и заполненность чанков
4. Определяем, нужно ли делить или перемещать чанки
5. Осуществляем миграцию

Решаем проблемы

Плохо

Ходить на каждый запрос в амстердам с западного побережья - не комильфо.

- Кеширование на стороне роутера

Всё ещё плохо

После каждой миграции, кешированная информация становится протухшей.

- Оповещаем роутеры, они обновляют кеш

На хлеб уже мажется.

Решаем проблемы

Но ещё пахнет

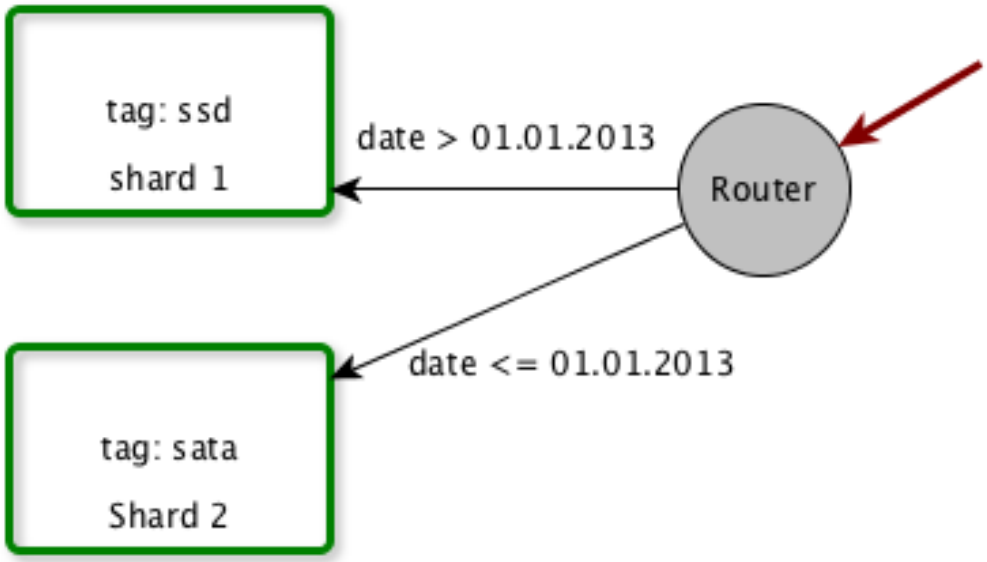
Не можем контролировать распределение чанков; нет заявленной локализации данных

Тэги

- Свойство шарда
- Отображение из диапазона ключей в тэг

Тэги

```
{ ns : "test.books", minKey : { date : 01.01.2013 }, maxKey : {date : $MaxKey
    }, tag : "ssd" }
```



Не решаем проблем

Не предусмотренное

- DDL-убивашка
- Нельзя так просто взять и поменять ключ шардирования
- Если поздно объявить коллекцию шардированной, все существующие данные навсегда окажутся в одном чанке
- Чанки не умеют склеиваться и не умирают при опустении

Выбираем ключ шардирования

Схема:

- 100 000 000 записей
- `{url: "http://www.imdb.com/title/tt0795176/", domain : 1, title: "Planet Earth", "air date": 2007, seasons:[1], rating:9.5 cast:<...>, similar:<...>, description:<...> }`
- Полный урл уникален
- Некоторые домены могут содержать десятки тысяч документов
- Список доменов меняется редко, информация по урлам меняется часто, домены имеют возрастающий идентификатор

Запросы

- Достать одну сущность по урлу
- Достать 1000 сущностей по урлам
- Достать всё из www.imdb.com
- Отгрузить всю базу

Выборы ключа шардирования (0)

Возможные варианты

- ObjectId()
- Поле с рандомным значением
- Урл
- Хеш от урла
- Пара <хеш от домена, хеш от урла>

Выборы ключа шардирования (1)

ObjectId

Преимущества

- Монотонно возрастает - легко собрать чанк

Недостатки

- Монотонно возрастает - запись в один шард
- Монотонно возрастает - пустые чанки
- Невозможно поддержать уникальность урлов в разных шардах

Выборы ключа шардирования (2)

Случайное значение

Преимущества

- Равномерная распределённость - запись во все шарды
- Равномерная распределённость - нет пустых чанков

Недостатки

- Равномерная распределённость - очень долгая миграция, стынет кеш
- Невозможно поддержать уникальность урлов в разных шардах

Выборы ключа шардирования (3)

Урл

Преимущества

- Уникальность урлов
- Локализация запросов
- Запись во все шарды
- Мало пустых чанков

Недостатки

- Тяжеловесный индекс
- Сложно собрать чанк - долгая миграция чанка

Выборы ключа шардирования (4)

Пара id домена- хеш от урла

Преимущества

- Те же, что и от урла
- Плюс бесплатный индекс по домену
- Быстрая миграция, если обновления по домену приходят батчами
- Легковесный индекс

Недостатки

- Долгая миграция, если гипотеза о батчах неверна
- Иногда чанки пустеют

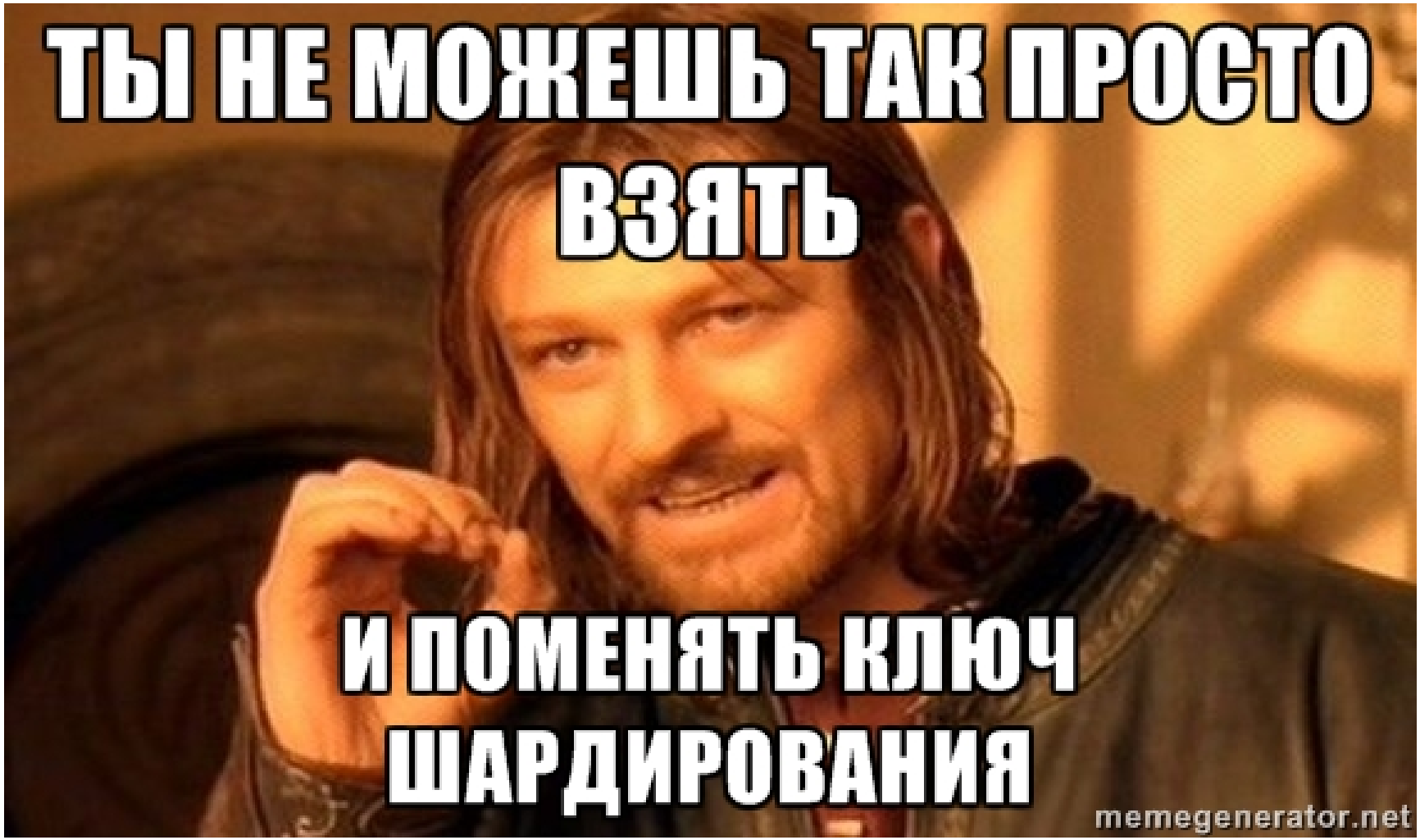
Выборы ключа шардирования (5)

Резюме

- На какие запросы хотим отвечать?
- Монотонно возрастающий ключ - плохо
- Полностью рандомный ключ - плохо
- Шардовый ключ участвует в запросе - дополнительный профит

Выборы ключа шардирования. Итоги

Студент! Помни!



Почему MongoDB не production-ready?

- Писалась исходя из того, что она работает
- Только SocketException в логах
- Совершенно не описано поведение в неблагоприятных условиях
- Некачественные мониторинги

Кстати о мониторингах

db.serverStatus()

- Возвращает кучу данных о базе
- Все показатели - в миллисекундах от старта сервера
- Мониторинг дёргает serverStatus

А теперь представим, что система перегружена

1. Дёргаем serverStatus
2. Через 20 секунд получаем ответ
3. Через минуту дёргаем ещё раз
4. Через 20 секунд получаем ответ
5. Разность значений может быть статистикой за любой промежуток времени от 40 до 80 секунд

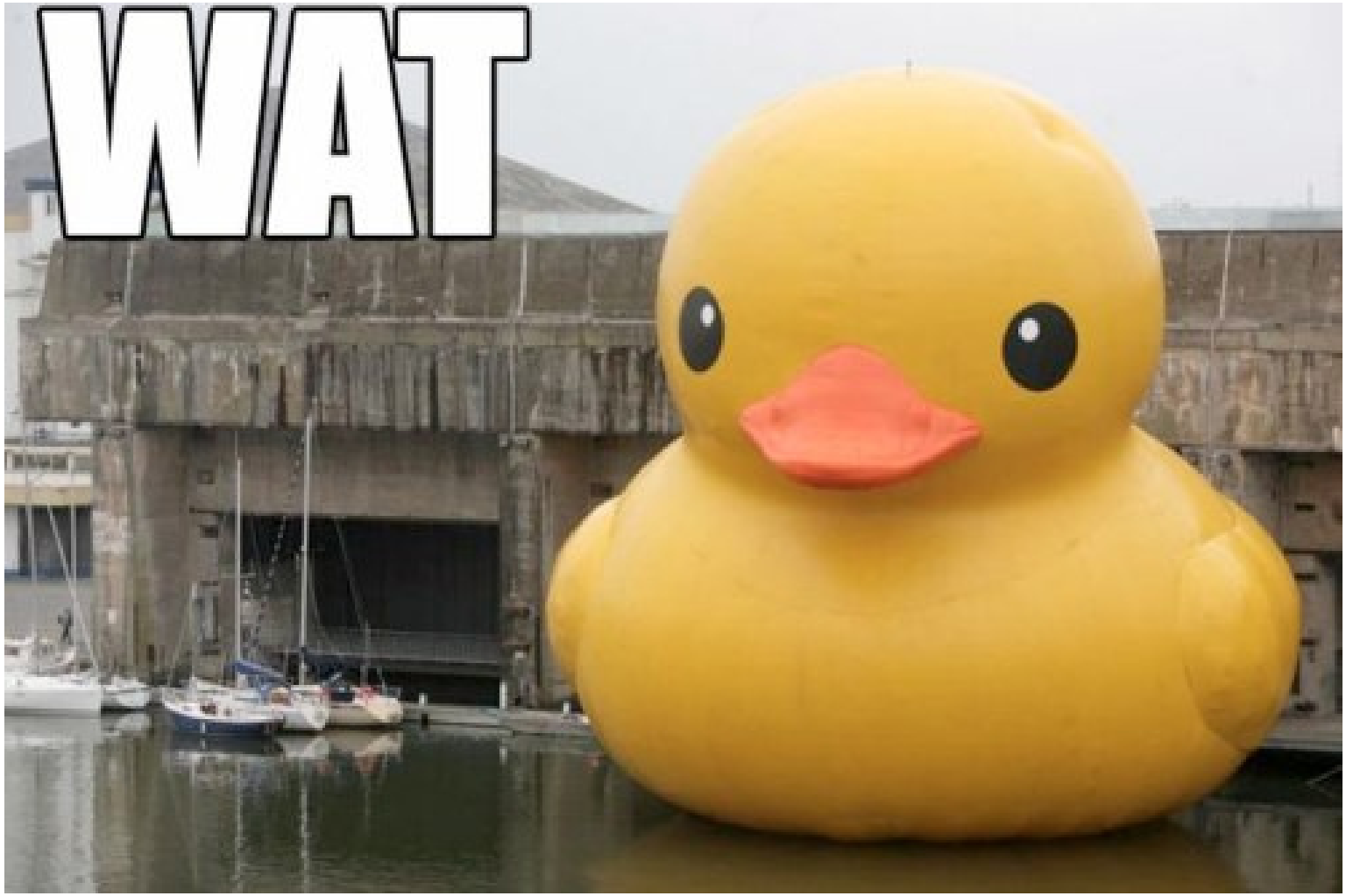
Кстати о мониторингах

А теперь делим разность из предыдущего слайда на 60

И получается, что за последнюю минуту половина баз была заблокирована на чтение в течение не меньше чем 90 секунд.

- Не вся статистика разделена по пространствам имён
- Не знаем, какие именно запросы перегружают базу

WAT?



Если действительно нужен функционал

1. Предельно уяснить, какие запросы важные
2. Описать ограничения уникальности
3. Построить ключ шардирования
4. Не выполнять DDL на боевых базах без крайней необходимости
5. Всегда вручную передёргивать роутеры после DDL
6. Заблаговременно масштабировать
7. Не тянуть с шардированием

Сухой остаток

Почему MongoDB - это действительно круто

- Однокнопочные шардирование и репликация
- Богатый язык запросов
- Вторичные индексы

Почему нужно трижды подумать

- Нечитаемые логи
- Непонятно, где у монги болит
- При неосторожном обращении, легко превратить всё в тыкву
- Не жмутся названия полей

Осталось за кадром

- `http://www.kchodorow.com/blog/2010/08/23/history-of-mongodb/` - История создания продукта
- Авторизация
- GridFS
- Профилирование
- Aggregation framework
- MapReduce

Литература

- http://www.amazon.com/MongoDB-Definitive-Guide-Kristina-Chodorow/dp/1449344682
- http://docs.mongodb.org/manual/
- http://www.kchodorow.com/blog/index.php?s=mongodb

Вопросы?

- Почта a.v.volokhov@gmail.com