



Онлайн-образование

Проверить, идет ли запись!



Меня хорошо видно && слышно?

Ставьте +, если все хорошо

Напишите в чат, если есть проблемы

Redis, часть 2



Дмитрий Кириллов

Технический директор

1С-Старт

@esteps_kirillov

Правила вебинара



Активно участвуем



Задаем вопрос в чат или голосом



Off-topic обсуждаем в Slack #канал-группы или #general



Вопросы вижу в чате, могу ответить не сразу

Маршрут вебинара

01. Конфигурация



02. Персистентность



03. Масштабирование



04. Итоги

Цели вебинара | После занятия вы сможете

1

Настроить ОС и Redis
в зависимости от бизнес-задач

2

Выбрать оптимальную стратегию
резервного копирования

3

Развернуть Redis Sentinel
и Redis Cluster

Краткое содержание предыдущих серий

- Redis хранит все данные **только** в оперативной памяти
- мы обращаемся к ним по ключам
- эти данные представляют собой строки, списки, таблицы...
- для ключа можно указать **время жизни (TTL)**
- Redis работает **в одном потоке**
- для обеспечения атомарности есть **транзакции**,
но они работают специфическим образом

01. Конфигурация

Redis после запуска:

- работает на порту 6379
- доступен всем пользователям без аутентификации
- не шифрует трафик
- в определённых случаях не привязан
к конфигурационному файлу

Настройка безопасности

- создать и подключить конфигурационный файл

/etc/redis/redis.conf

Настройка безопасности

- подключить TLS

```
tls-cert-file /path/to/redis.crt  
tls-key-file /path/to/redis.key  
tls-ca-cert-file /path/to/ca.crt  
tls-dh-params-file /path/to/redis.dh
```

Настройка безопасности

- Redis < 6.0: включить аутентификацию по паролю

```
CONFIG SET requirepass <пароль>
AUTH <пароль>
```

- Redis >= 6.0: настроить ACL

```
ACL SETUSER <пользователь> [правила]
AUTH <пользователь> <пароль>
```

Базовые команды ACL

```
ACL LIST
```

Список пользователей и правил

```
ACL SETUSER dmitry  
on >123456  
~cached:*  
+get
```

Создание пользователя:
- активный
- доступ к ключам cached: *
- может использовать get

```
ACL GETUSER dmitry
```

Подробная информация

```
AUTH dmitry 123456
```

Аутентификация

Вопросы для обсуждения:

- по каким критериям вы будете **разделять** клиентов Redis на группы?
- какие команды вы будете **запрещать** – и почему?

Журналирование

Журналирование Redis

- logfile указывается в конфиге
- запросы туда **не пишутся**
- при необходимости можно включить syslog

Типичный лог:

/var/log/redis/redis-server.log

Отслеживание всех запросов

- используется команда MONITOR
- её вывод можно перенаправить в файл
- это **крайне** сильно замедляет систему

Отслеживание медленных запросов

- настраиваем Slow Log в конфиге

```
slowlog-log-slower-than 10000 # это микросекунды!  
slowlog-max-len 128
```

- смотрим результаты

```
SLOWLOG GET      # В ответе будут тоже микросекунды!  
SLOWLOG RESET
```

Статистика и конфигурация

Статистика Redis

- выводится как результат команды INFO
- можно отправить в систему мониторинга
- сбрасывается с помощью CONFIG RESETSTAT

Статистика и конфигурация

Конфигурация Redis

- выводится как результат команды CONFIG GET
- изменяется с помощью CONFIG SET
- изменения хранятся только в оперативной памяти, при необходимости их нужно сбрасывать на диск

Статистика и конфигурация

Основные команды

INFO

INFO CPU

CONFIG RESETSTAT

CONFIG GET port

CONFIG SET save "3600 1"

CONFIG REWRITE

Информация + статистика

Конкретный раздел

Сбросить статистику

Получить значение конфига

Установить значение конфига

Сохранить настройки на диск

Производительность

Оптимизации на уровне ОС

- отключить transparent_hugepage
- увеличить /proc/sys/fs/file-max
- увеличить net.core.somaxconn
- уменьшить vm.swappiness

Оптимизации на уровне клиентского кода

- использовать пул постоянных соединений
- использовать прокси (twemproxy, sentinel_tunnel...)
- оборачивать цепочки команд в pipeline
- использовать скрипты на языке Lua*

* позволяет организовывать на стороне Redis аналоги хранимых процедур

Оптимизации на уровне Redis

- использовать Insecticide
<https://github.com/city-mobil/insecticide>
- в общем случае анализировать результаты redis-benchmark

Управление памятью

Вопросы на засыпку:

- какие способы контроля над заполнением памяти предоставляют разные языки программирования?

Управление памятью

Действия при переполнении памяти

В Redis можно выбрать один из алгоритмов вытеснения "лишних" данных:

noeviction Не вытеснять (будет ошибка при переполнении)

allkeys-lru Удалять данные, которыми редко пользуются

volatile-ttl Удалять данные, которые скоро "прокиснут"

allkeys-random Удалять случайные данные

02. Персистентность

Персистентность

RDB (Redis Database)

Сохранение полных слепков (snapshots) данных

- вся база полностью сохраняется в один файл
- идеален для бэкапов с быстрым разворачиванием
- обычно создаётся раз в S секунд
или после N операций записи

Сценарии использования

- когда допустимо потерять часть данных (аналитика)

Персистентность

Конфигурация RDB

```
save 3600 1
```

```
# Сохранять слепок, если
# - за последние 3600 секунд
#   изменился хотя бы 1 ключ
# - за последние 300 секунд
#   изменилось хотя бы 100 ключей
#   и т.д.
```

```
save 300 100
```

```
dir "/var/lib/redis"
dbfilename "dump.rdb"
```

```
# Каталог и файл
# со слепком базы
```

Создание слепка RDB

BGSAVE

Асинхронное сохранение,
выполняется клоном процесса

SAVE

Синхронное сохранение,
выполняется в этом же процессе

Как Redis создаёт слепок RDB

- выполняется системный вызов `fork(2)`
- создаётся **полный клон** процесса Redis
- этот клон читает все данные из своей копии памяти и скидывает их на диск
- запись выполняется **атомарно**
(созданный файл не перезаписывается, а заменяется)

Персистентность

Как Redis создаёт слепок RDB

Родительский процесс Redis

site:name	Магазин игрушек		
user:1	uid	1	
	name	Елена	
	city	Москва	
user:2	uid	2	
	name	Иван	
	city	Воронеж	
orders:total	14800	user:2	
	12000	user:1	

Клон процесса

site:name	Магазин игрушек		
user:1	uid	1	
	name	Елена	
	city	Москва	
user:2	uid	2	
	name	Иван	
	city	Воронеж	
orders:total	14800	user:2	
	12000	user:1	

Персистентность

Вопросы на засыпку:

- есть ли нестыковки этого процесса с тем, как устроен Redis?
- о каких проблемах эксплуатации сразу нужно задуматься?

Персистентность

Экономим память: Copy-on-Write

Родительский процесс Redis

site:name	Магазин игрушек		
user:1	uid	1	
	name	Елена	
	city	Москва	
user:2	uid 2 name Иван city Саратов		
orders:total	14800	user:2	
	12000	user:1	

↗ Значение ключа изменилось

Клон процесса

site:name	Магазин игрушек		
user:1	uid	1	
	name	Елена	
	city	Москва	
user:2	uid 2 name Иван city Воронеж		
orders:total	14800	user:2	
	12000	user:1	

↗ Создаётся копия старой версии

Персистентность

Экономим память: Copy-on-Write

- установить `vm.overcommit_memory = 1`

AOF (Append Only File)

Инкрементальный лог изменений

- аналог WAL
- минимальная (или нулевая) потеря свежих данных
- более устойчивый к потере питания
- в целом медленнее, чем RDB

Сценарии использования

- когда важна целостность данных (event sourcing, streams...)

Конфигурация AOF

```
appendonly yes          # Включить AOF  
appendfilename "appendonly.aof" # Имя файла  
  
appendfsync everysec    # Частота выполнения fsync()
```

AOF и буферизация на стороне ОС

- если попросить ОС записать данные на диск, эти данные обычно сначала ложатся в буфер
- ОС выполнит реальную запись тогда, когда сочтёт нужным
- её можно поторопить, вызвав **блокирующую** операцию `fsync(2)`
- при этом даётся **почти 100%** гарантия того, что данные будут сохранены на носитель

Персистентность

Вопросы для обсуждения:

Redis:

- создаёт атомарные RDB-слепки
- постоянно дописывает инкрементарные AOF-файлы

Вопросы:

- с какими проблемами можно столкнуться при использовании персистентности Redis в облаке?
- предложите стратегию резервного копирования для одной ноды Redis под высокой нагрузкой

03. Масштабирование

Масштабирование

Три способа масштабирования Redis

Redis предлагает "из коробки" механизмы репликации, а также контроля над ними:

Репликация

Master + несколько slave

Redis Sentinel

Репликация
+ выборы нового master-а

Redis Cluster

Репликация
+ шардирование
+ выборы нового master-а для шарда

Репликация

Способ 1. Репликация

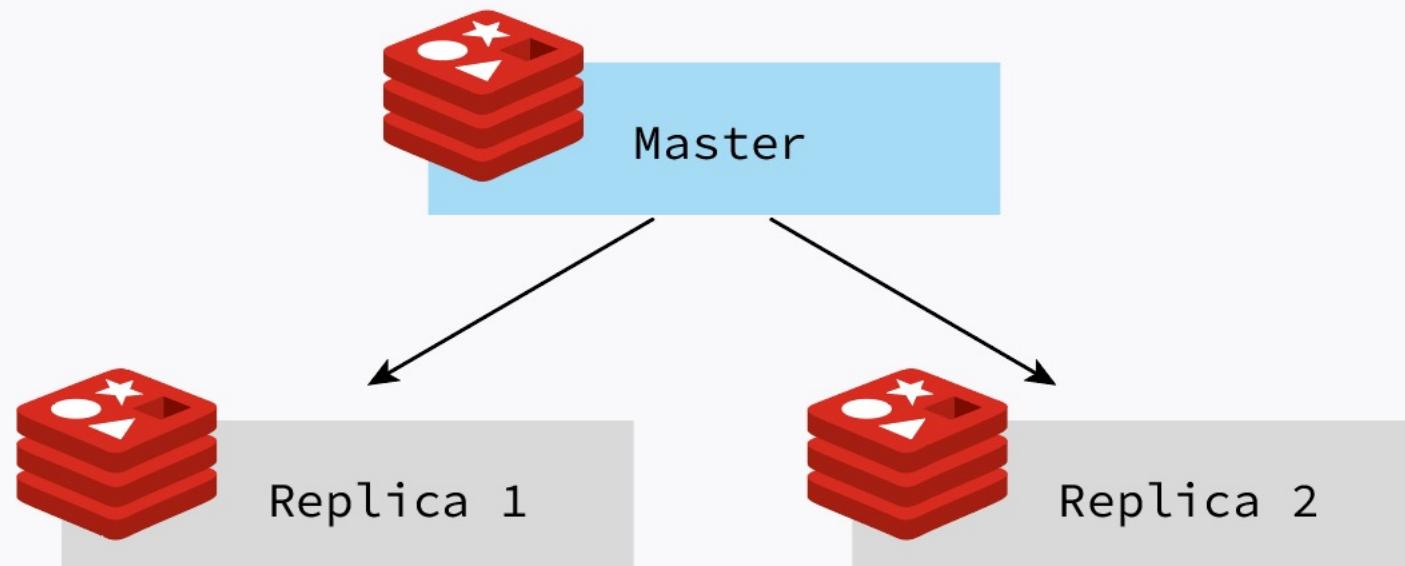
- архитектура master-replica
- реплики по умолчанию read-only
- при синхронизации реплики вызывают PSYNC и получают обновлённые данные от мастера
- как следствие, на репликах могут какое-то время быть доступны устаревшие или "прокисшие" данные

Важно помнить:

- Redis "из коробки" не обрабатывает отказы репликации

Репликация

Репликация



Репликация

Настройка репликации

- указываем мастера в конфигах реплик

```
replicaof 80.249.147.12 6379  
masterauth 1q2w3e4r5t
```

- проверяем результат

```
INFO replication
```

Способ 2. Redis Sentinel

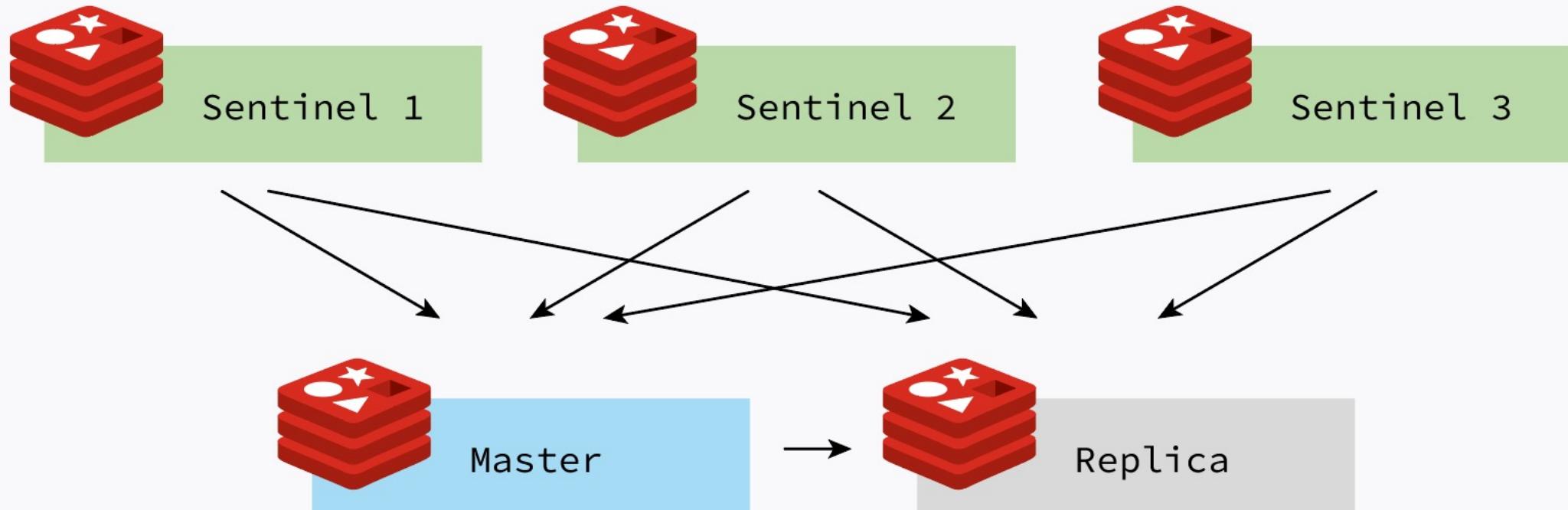
- аналог Patroni для PostgreSQL
- запускается несколько экземпляров Sentinel (минимум 3)
- запускается несколько серверов Redis (master + реплики)
- при отказе master производятся выборы нового master

Важно помнить:

- если вы используете Sentinel,
его должен поддерживать клиентский код
- при определённых условиях система может
прийти в неработоспособное состояние

Redis Sentinel

Redis Sentinel



Настройка Redis Sentinel

- создаём конфигурационный файл

/etc/redis/sentinel.conf

- добавляем в него настройки для master-a

```
daemonize yes
```

```
logfile "/var/log/redis/sentinel.log"
```

```
sentinel monitor mymaster 80.249.147.12 6379 2
```

```
sentinel auth-pass mymaster 1q2w3e4r5t
```

```
sentinel down-after-milliseconds mymaster 5000
```

Настройка Redis Sentinel

- устанавливаем нужные права

```
chown redis:redis /etc/redis/sentinel.conf  
touch /var/log/redis/sentinel.log  
chown redis:redis /var/log/redis/sentinel.log  
chmod 640 /etc/redis/sentinel.conf  
chmod 660 /var/log/redis/sentinel.log
```

Настройка Redis Sentinel

- запускаем Sentinel

```
/usr/bin/redis-server /etc/redis/sentinel.conf \
--sentinel
```

- подключаемся к нему (порт 26379)

```
redis-cli -p 26379
```

Основные команды Redis Sentinel

SENTINEL MASTERS	# Список мастеров
SENTINEL MASTER <u>mymaster</u>	# Информация о мастере
SENTINEL REPLICAS <u>mymaster</u>	# Информация о репликах
SENTINEL SENTINELS <u>mymaster</u>	# Информация о Sentinels
SENTINEL GET-MASTER-ADDR-BY-NAME <u>mymaster</u>	# Получить IP по имени мастера

Использование Redis Sentinel на клиенте

Пример на Java + jedis:

```
var masterName = "mymaster";
var masterPass = "1q2w3e4r5t";

Set<String> sentinels = new HashSet<>();
sentinels.add("192.168.0.10:26379");
sentinels.add("192.168.0.11:26379");

var pool = new JedisSentinelPool(masterName, sentinels);
var jedis = pool.getResource();
jedis.auth(masterPass);
```

Сценарий отказа: недоступность master

- по истечении down-after-milliseconds начинаются выборы
- все Sentinels голосуют
- если выборы прошли успешно, в конфиги репликации **автоматически** вносятся правки
- назначается новый мастер
- клиентский код при этом **не изменяется**

Способ 3. Redis Cluster

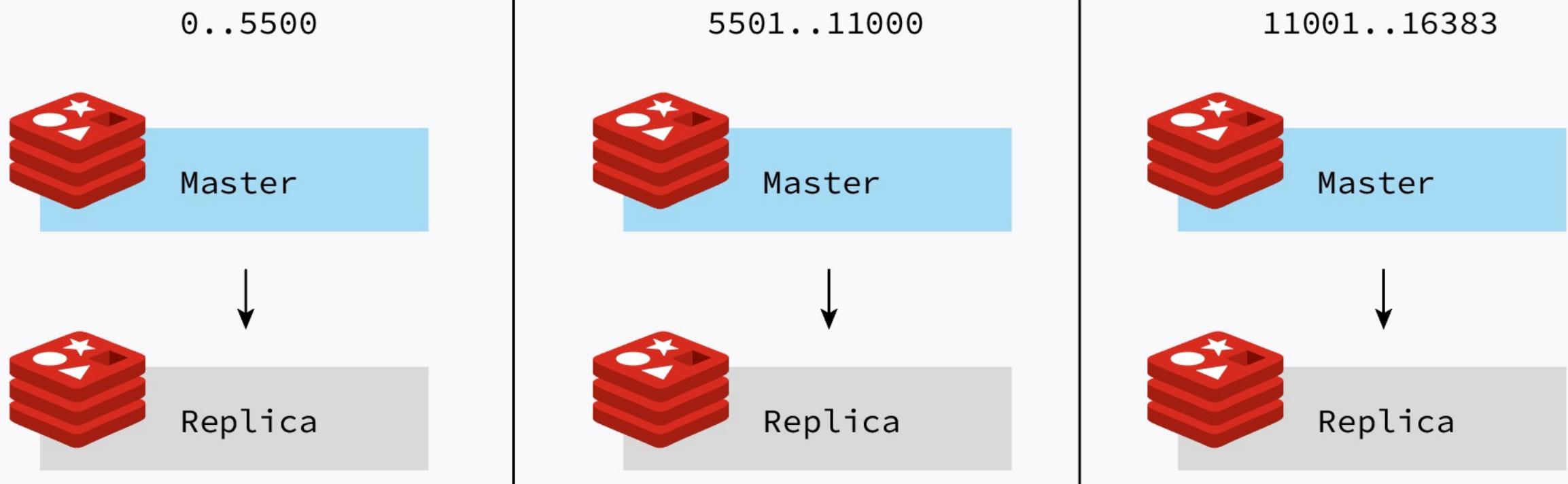
- автоматическое шардирование + репликация
- не требует Sentinel (выполняет в т.ч. его задачи)
- требуется минимум 3 ноды Redis
- желательно минимум 6 нод Redis для устойчивой работы

Важно помнить:

- если вы используете Redis Cluster,
его должен поддерживать клиентский код
- Redis Cluster **не гарантирует** ни доступность,
ни устойчивость

Redis Cluster

Redis Cluster



Шардирование

- база разбивается на непересекающиеся фрагменты – **шарды**
- шард обычно включает в себя мастера + его реплики
- ключи **автоматически** распределяются по шардам
- если клиент сделает запрос на "чужой" шард,
клUSTER подскажет редирект
- если **очень** нужно, чтобы какие-то ключи
хранились рядом, можно использовать тэги:

{user:1}:token

{user:1}:session

Настройка Redis Cluster

— создаём для каждой ноды свой конфигурационный файл

```
port 7000          # у каждой ноды будет свой порт
cluster-enabled yes
cluster-config-file nodes.conf
cluster-node-timeout 5000
appendonly yes      # Обязательно для работы кластера!
```

Настройка Redis Cluster

- запускаем все ноды по отдельности
- объединяем их в кластер

```
redis-cli --cluster create \
 127.0.0.1:7000 \
 127.0.0.1:7001 \
 127.0.0.1:7002 \
 127.0.0.1:7003 \
 127.0.0.1:7004 \
 127.0.0.1:7005 \
--cluster-replicas 1
```

Использование Redis Cluster на клиенте

Пример на Java + jedis:

```
Set<HostAndPort> clusterNodes = new HashSet<>();  
clusterNodes.add(new HostAndPort("127.0.0.1", 7000));  
  
var jc = new JedisCluster(jedisClusterNodes);  
  
jc.set("foo", "bar");
```

Redis Cluster

Вопросы для обсуждения:

Redis Cluster:

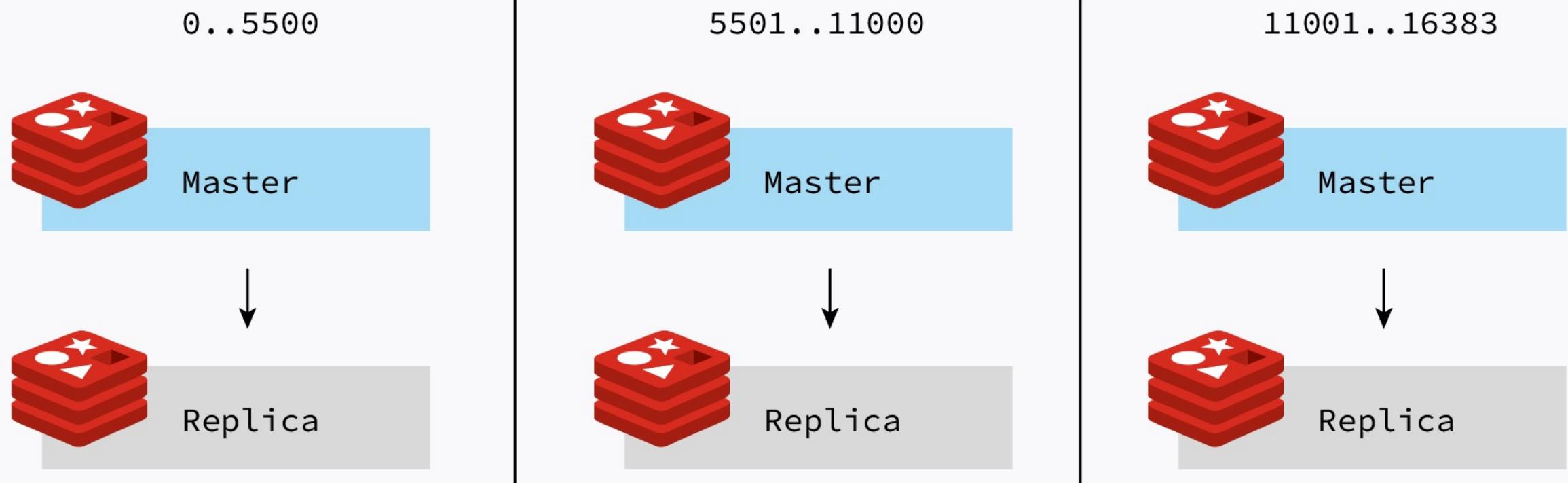
- разбивает базу на несколько шардов
- отвечает за доступность каждого из них

Вопросы:

- какие проблемы могут возникнуть при эксплуатации кластера?
- предложите оптимальную сетевую топологию, если у вас всего 3 физических сервера и 6 нод

Redis Cluster

Redis Cluster



Масштабирование

Три способа масштабирования Redis

Сценарии использования:

Redis Sentinel

- чаще читаем, чем пишем

Redis Cluster

- база не помещается в ОЗУ
- высокая нагрузка на CPU

04. Итоги

Redis: основные характеристики

Тип БД

Key-value

Ключевые "фишки"

Всё в ОЗУ, множество структур данных

Производительность

Крайне высокая: в основном $O(1)$ + ОЗУ

Типы данных

Строки + структуры данных

Ограничения данных

512 Mb на ключ/значение, ОЗУ на всю БД

Применимость

- кэширование
- "внешняя память" приложения
- быстрый брокер сообщений

Redis: эксплуатация

Мониторинг

Логи запросов

Бэкапы

Отказоустойчивость

Кластеризация

Redis_exporter (нет в коробке)

MONITOR + Slow Log

Слепки БД + AOF, автовосстановление

Гарантируется только для одной ноды

Шардирование + репликация

Redis: сравнение с другими NoSQL

Document Store

Redis может частично имитировать
документы с помощью структур данных

etcd / Consul

Несопоставимо (разные сценарии)

FoundationDB

Redis не заточен под транзакции

Tarantool, Aerospike

Redis менее функционален

Redis: выводы

- основной плюс — скорость
- основной недостаток — надёжность
- Redis существует на рынке много лет, поэтому легко найти как сотрудников, так и информацию
- облачные провайдеры предлагают базы, **полностью** совместимые с Redis, но более надёжные

Цели вебинара | Проверка достижения целей

1

Настроить ОС и Redis
в зависимости от бизнес-задач

2

Выбрать оптимальную стратегию
резервного копирования

3

Развернуть Redis Sentinel
и Redis Cluster

Домашнее задание (NoSQL)

- 1 Сохранить большой JSON (~20МБ) в виде разных структур
- 2 Протестировать скорость сохранения и чтения
- 3 Предоставить отчет



Срок: желательно сдать до 10.01

* для уверенных в себе студентов есть дополнение к д/з «со звёздочкой»

Следующий вебинар

Тема: у каждой группы своя



Дата: у каждой группы своя



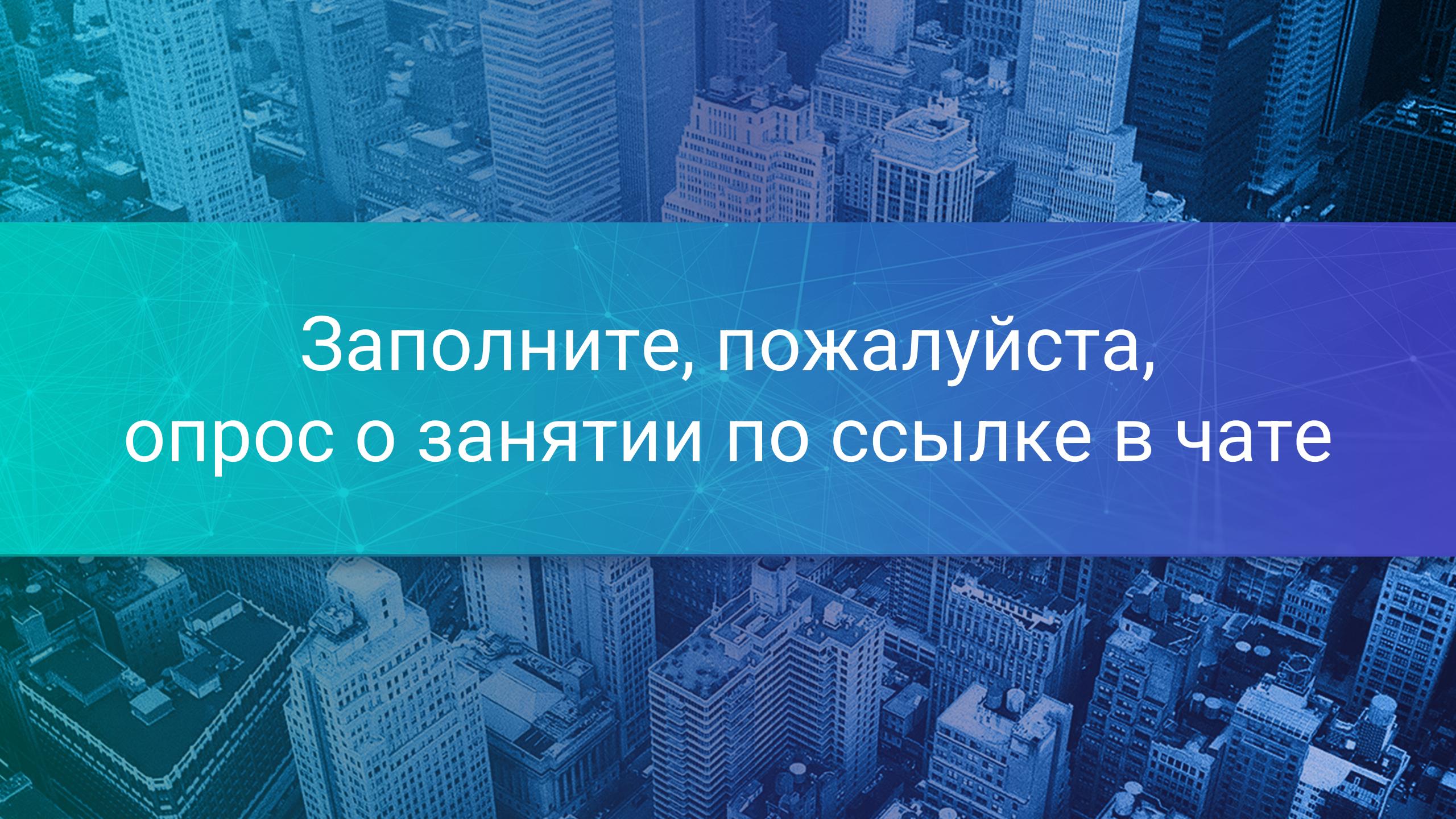
Ссылка на вебинар будет в ЛК за 15 минут



Материалы к занятию
в ЛК – можно
изучать



Обязательный
материал обозначен
красной лентой



Заполните, пожалуйста,
опрос о занятии по ссылке в чате

Спасибо за внимание!
Приходите на следующие вебинары



Дмитрий Кириллов

Технический директор

1С-Старт

@esteps_kirillov