Базы данных
Лекция 8

Redis

Мгер Аршакян

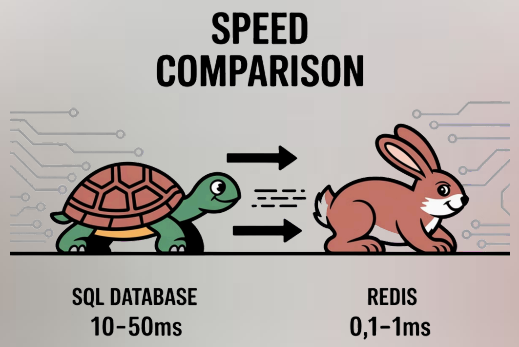
План лекции

1. Введение в Redis
2. Структуры данных Redis
3. Транзакции и атомарность
4. Паттерны использования Redis

Введение в Redis

Что такое Redis

1. Redis (Remote Dictionary Server) - быстрое in-memory хранилище данных типа "ключ-значение"
2. Поддерживает множество типов данных и операций над ними



Почему Redis так популярен?

1. Исключительная скорость (>100,000 операций/сек)
2. Атомарные операции и поддержка транзакций
3. Расширенная функциональность: Pub/Sub, Lua-скриптинг
4. Идеален для кеширования и временных данных

Архитектура Redis

1. Однопоточная модель на основе цикла событий (event loop)
2. Все данные хранятся в оперативной памяти
3. Опциональная персистентность данных
4. Поддержка репликации master-slave
5. Кластеризация для масштабирования

Структуры данных Redis

Ключи в Redis

1. Строки (максимум 512 MB)
2. Принятые соглашения: 'object-type:id' (user:1000)
3. Команды: KEYS, SCAN, EXISTS, DEL, EXPIRE

Значения в Redis

1. **Strings** - Самый простой тип данных в Redis (512 МБ)
2. **Lists** - Связанные списки строк, упорядоченные по порядку вставки, вставка/удаление с обоих концов O(1) (**FIFO, LIFO**)
3. **Sets** - Неупорядоченные коллекции уникальных строк, проверка на наличие (O(1))
4. **Bitmaps** - бинарные данные

Управление временем жизни (TTL)

1. Автоматическое удаление ключей по истечении времени.
2. Ключи с временем жизни vs без него, это разделение важно для некоторых политик эвакуации
3. Использование с политиками эвакуации

Стратегии эвакуации

1. **noeviction** — при переполнении памяти ничего не удаляет, записывающая команда получает ошибку `ERR maxmemory`
2. **allkeys-lru** — удаляет самый давно неиспользованный ключ (Least Recently Used) среди **всех** ключей
3. **volatile-lru** — удаляет самый давно неиспользованный ключ **только из тех, у кого установлен TTL**
4. **allkeys-random** — удаляет произвольный ключ, выбранный случайно, из всех ключей
5. **volatile-random** — случайно удаляет ключ **лишь среди ключей с TTL**
6. **volatile-ttl** — удаляет ключ с ближайшим истечением времени жизни (наименьший TTL)

Персистентность в Redis

1. **RDB (Redis Database)** - снапшоты через заданные интервалы
2. **AOF (Append Only File)** - журналирование каждой модифицирующей операции

Транзакции и атомарность

Транзакции в Redis

1. **MULTI** переводит клиента в режим «собираю пакет».
2. **EXEC** запускает *последовательное* выполнение пакета, и никакие другие клиенты не могут вклиниться между командами.
3. **Rollback’а нет**: если одна команда из пакета падает, предыдущие уже отработали, последующие всё-равно пойдут.

```
127.0.0.1:6379> MULTI                  # открываем транзакцию
OK
127.0.0.1:6379(TX)> INCR page:1:views      # увеличиваем первый счётчик
QUEUED
127.0.0.1:6379(TX)> INCR page:2:views      # и второй
QUEUED
127.0.0.1:6379(TX)> EXEC                # запускаем пакет
```

Atomic Counters: INCR / DECR

1. **INCR / DECR** - Увеличивает или уменьшает строку-число на 1
2. **INCRBY / DECRBY** - Прибавляет/вычитает произвольное целое
3. **Автосоздание ключа** - Если ключа нет, Redis считает его «0» и сразу меняет

```
> INCR page:123:views         # 1
> INCRBY page:123:views 5     # 6
> DECR user:42:tokens         # -1   (создался сам)
```

Pub/Sub: мгновенные уведомления

1. **Модель Publish / Subscribe** — издатели (PUBLISH) посылают сообщения в *каналы*, подписчики (SUBSCRIBE) мгновенно их получают
2. **Fire-and-forget** — сообщения **не хранятся**: если подписчика нет в момент публикации, событие теряется
3. **Нулевые накладные расходы** — нет очередей, ack или курсоров, Redis просто рассылает копии в сетевые сокеты

```
# Терминал A — подписчик
> SUBSCRIBE chat:general
Reading messages... (press Ctrl-C to quit)
1) "message"
2) "chat:general"
3) "Hello, Redis!"

# Терминал B — издатель
> PUBLISH chat:general "Hello, Redis!"
(integer) 1        # число активных получателей
```

Паттерны использования Redis

Кэш (Cache-Aside / Read-Through)

Храним часто-читаемые данные (HTML-фрагменты, результаты SQL-запросов, внешних API). При промахе читаем из основной БД, кладём в Redis с TTL. Уменьшаем нагрузку на «тяжёлые» ресурсы, сокращаем задержку страницы до < 1 мс.

Хранение сессий

`SET session:<token> {json} EX 1800` — прикладные сервера (Django, Express, Spring) быстро читают/пишут сессионное состояние, не блокируя СУБД. TTL автоматически удаляет «потухшие» сессии; горизонтальное масштабирование приложений упрощается

Pub/Sub уведомления

Мгновенная доставка событий: чат, live-обновления дашбордов, «горячая» перезагрузка конфигурации микросервисов. `PUBLISH` в канал — подписчики получают сообщение без хранения на сервере

Rate Limiting

`INCR user:42:hits; EXPIRE 60` — считаем запросы пользователя в минуту

Спасибо за внимание!

Беседа курса в Telegram