**Системная и программная инженерия**

Лабораторная работа №2 «Знакомство с Redis»

**Введение**

**Redis** (от англ. remote dictionary server) — резидентная система управления базами данных класса NoSQL с открытым исходным кодом, работающая со структурами данных типа «ключ — значение». Используется как для баз данных, так и для реализации кэшей, брокеров сообщений.

Ориентирована на достижение максимальной производительности на атомарных операциях.

Все данные Redis хранит в виде словаря, в котором ключи связаны со своими значениями. Одно из ключевых отличий Redis от других хранилищ данных заключается в том, что значения этих ключей не ограничиваются строками. Поддерживаются следующие абстрактные типы данных: строки, списки, множества, хеш-таблицы, упорядоченные множества.

**Преимущества Redis**

* **Производительность**

Все данные Redis хранятся в памяти, что обеспечивает низкую задержку и высокую пропускную способность доступа к данным. В отличие от традиционных баз данных, хранилища данных в памяти не требуют перемещения на диск, что сокращает задержку ядра до микросекунд.

* **Гибкие структуры данных**

В отличие от других хранилищ на основе пар «ключ – значение», которые поддерживают ограниченный набор структур данных, Redis поддерживает огромное разнообразие структур данных, позволяющее удовлетворить потребности разнообразных приложений.

Типы данных Redis включают:

* строки – текстовые или двоичные данные размером до 512 МБ;
* списки – коллекции строк, упорядоченные в порядке добавления;
* множества – неупорядоченные коллекции строк с возможностью пересечения, объединения и сравнения с другими типами множеств;
* сортированные множества – множества, упорядоченные по значению;
* хэш‑таблицы – структуры данных для хранения списков полей и значений;
* битовые массивы – тип данных, который дает возможность выполнять операции на уровне битов;
* структуры HyperLogLog – вероятностные структуры данных, служащие для оценки количества уникальных элементов в наборе данных;
* потоки – очереди сообщений со структурой журналов данных;
* пространственные данные – записи карт на основе долготы/широты, «поблизости»
* JSON – полуструктурированный объект со вложенной структурой из именованных значений с поддержкой чисел, строк, булевских значений, массивов и других объектов

**Простота и удобство**

Redis позволяет писать такой же сложный код с меньшим количеством простых строк. Redis позволяет писать меньше строк для хранения, использования данных и организации доступа к данным в приложениях. Разница в том, что, в отличие от языков запросов традиционных баз данных, с Redis разработчики могут использовать простую структуру команд. Например, вы можете задействовать структуру хэш-данных Redis, чтобы перемещать данные в хранилище только одной строкой кода. Решение подобной задачи с использованием хранилища данных, не поддерживающего структуры хэш‑таблиц, потребует написания серьезного объема кода для преобразования данных из одного формата в другой. Redis уже оснащен встроенными структурами данных и предоставляет множество возможностей их комбинирования и взаимодействия с данными клиента. Разработчикам под Redis доступны более ста клиентов с открытым исходным кодом. Поддерживаемые языки программирования включают Java, Python, PHP, C, C++, C#, JavaScript, Node.js, Ruby, R, Go и многие другие.

**Репликация и постоянное хранение**

В Redis применяется архитектура узлов «ведущий‑подчиненный» и поддерживается асинхронная репликация, при которой данные могут копироваться на несколько подчиненных серверов. Это обеспечивает как улучшенные характеристики чтения (так как запросы могут быть распределены между серверами), так и ускоренное восстановление в случае сбоя основного сервера. Для обеспечения постоянного хранения Redis поддерживает снимки состояния на момент времени (копирование наборов данных Redis на диск).

**Высокая доступность и масштабируемость**

Redis предлагает архитектуру «ведущий‑подчиненный» с одним ведущим узлом или с кластерной топологией. Это позволяет создавать высокодоступные решения, обеспечивающие стабильную производительность и надежность. Если требуется настроить размер кластера, доступны различные варианты вертикального и горизонтального масштабирования. В результате можно наращивать кластер в соответствии с потребностями.

**Инструменты с открытым исходным кодом**

Redis – проект с [открытым исходным кодом](https://ru.wikipedia.org/wiki/%D0%9E%D1%82%D0%BA%D1%80%D1%8B%D1%82%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5), поддерживаемый активным сообществом.

**Распространенные примеры использования Redis**

* **Кэширование**

Redis прекрасно подходит для организации высокодоступного кэша в памяти, который уменьшает задержку доступа, увеличивает пропускную способность и снижает нагрузку на реляционную базу данных или базу данных NoSQL и на приложение. Redis может обеспечить доступ к часто запрашиваемым данным с задержкой в доли миллисекунды и позволяет с легкостью выполнять масштабирование, справляясь с повышением нагрузок без дорогостоящего наращивания мощности БД на уровне сервера. Типовые примеры использования Redis – это кэширование результатов запросов к базе данных, долговременных сессий, веб‑страниц или часто используемых объектов, таких как изображения, файлы и метаданные.

* **Чат, обмен сообщениями и очереди**

Redis поддерживает системы «издатель – подписчик» с заданными шаблонами и различные структуры данных, такие как списки, сортированные множества и хэш‑таблицы. Это позволяет использовать Redis для создания высокопроизводительных комнат чата, лент комментариев, работающих в режиме реального времени, лент новостей в социальных сетях и систем взаимодействия серверов. Структура данных список сервиса Redis позволяет легко создавать упрощенные очереди. Списки Redis List обеспечивают выполнение элементарных операций, а также возможности блокировки, поэтому подходят для различных приложений, в которых требуется надежный брокер сообщений или циклический список.

* **Игровые таблицы лидеров**

Разработчики игр нередко применяют Redis для создания таблиц лидеров в режиме реального времени. Достаточно просто использовать структуру данных Redis Sorted Set, которая обеспечивает уникальность элементов и сортировку списка по результатам пользователей. Создание ранжированного списка в режиме реального времени в итоге требует лишь обновления результата пользователя при его изменении. Можно также применять структуры Sorted Set для обработки временных данных с использованием в качестве результата временных меток.

* **Хранилище сессий**

Redis как хранилище данных в памяти с высокой доступностью и долговременным хранением широко применяется для хранения данных сессий в приложениях, работающих в масштабе всего Интернета, а также для управления такими данными. Redis обеспечивает задержку на уровне долей миллисекунды, масштабируемость и отказоустойчивость, необходимые для управления такими данными сессий, как профили пользователей, учетные данные, состояние сессий и индивидуальные пользовательские настройки.

* **Потоковая передача мультимедиа**

Redis предлагает быстрое хранилище данных в памяти для примеров использования с потоковой передачей в режиме реального времени. Redis можно использовать для хранения метаданных профилей пользователей и историй просмотров, данных и токенов аутентификации миллионов пользователей, а также файлов манифеста. Это позволяет сетям CDN одновременно выполнять потоковую передачу видео миллионам пользователей мобильных и настольных компьютеров.

**Практическая часть**

**Установка**

В данной лабораторной работе для удобства развертывания Redis будет использоваться Docker. Предварительно необходимо скачать образ Redis (можно не использовать средства контейнеризации Docker, а установить Redis непосредственно на вашу рабочую машину)

Скачаем последнюю доступную версию, применив команду *docker pull redis:*

*Изображение выглядит как текст

Автоматически созданное описание*

Рис. 1. Выполнение команды «*docker pull redis*»

Убедимся, что все сделали правильно: запустим контейнер с образом Redis и перейдем в него.

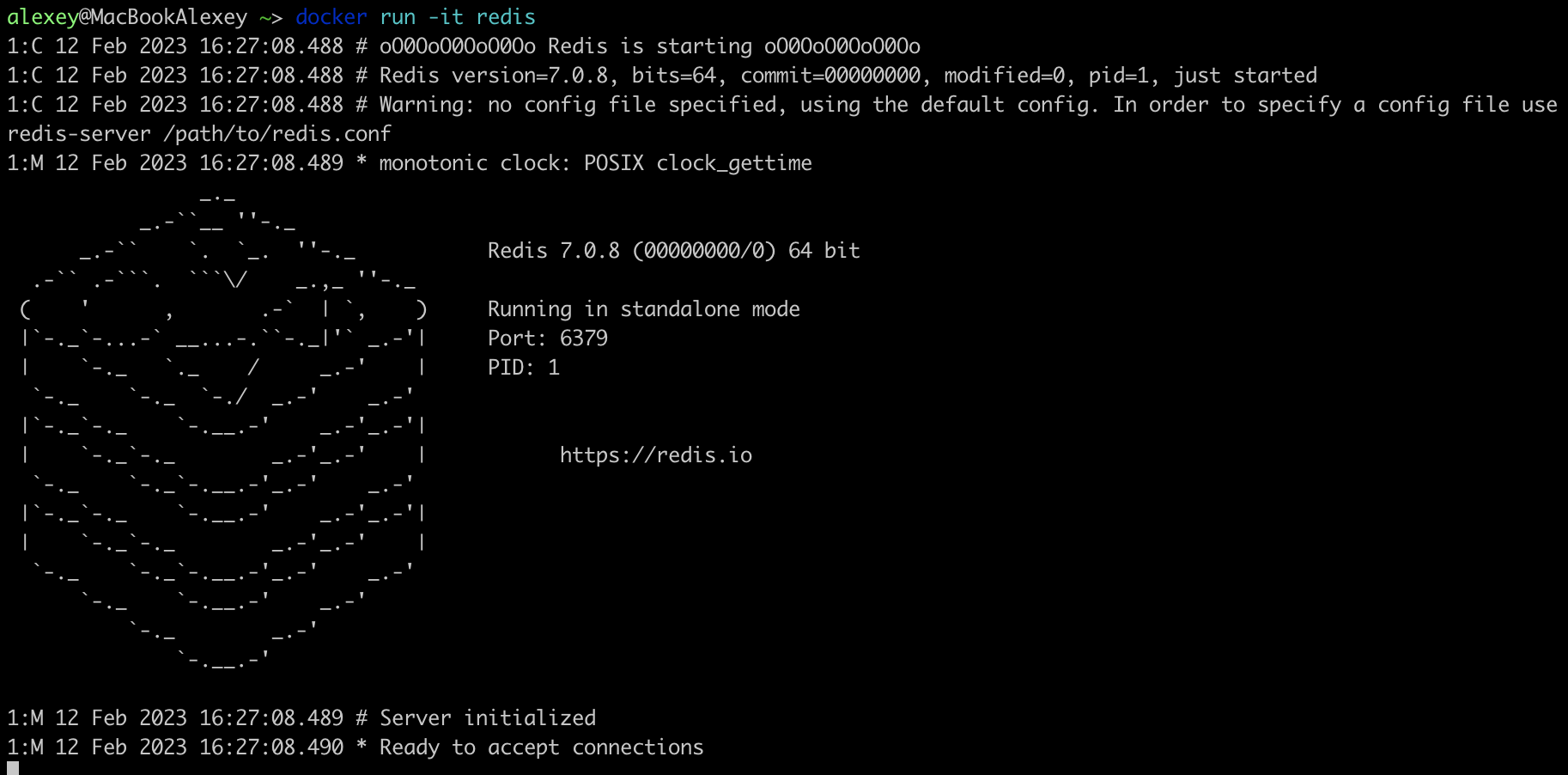


Рис. 2. Выполнение команды «*docker run -it redis*»

Внешние программы взаимодействуют с Redis, используя сокет TCP и специальный протокол Redis. Этот протокол реализован в клиентских библиотеках Redis для разных языков программирования. Однако, чтобы упростить работу с Redis, он предоставляет утилиту командной строки, которую можно использовать для отправки команд в Redis. Эта программа называется redis-cli.

Чтобы запустить redis-cli в интерактивном режиме необходимо перейти в запущенный контейнер и прописать в консоли команду *redis-cli*

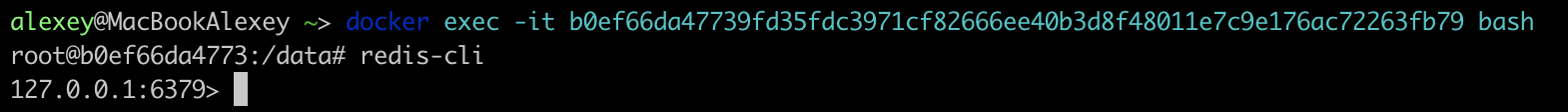


Рис. 4. Выполнение команды «*redis-cli*»

**Ключи**

Ключи Redis безопасны для двоичных файлов, это означает, что вы можете использовать любую двоичную последовательность в качестве ключа, от строки, такой как «foo», до содержимого файла JPEG. Пустая строка также является допустимым ключом.

**Еще несколько правил о ключах:**

* Очень длинные ключи - не лучшая идея. Например, ключ размером 1024 байта — плохая идея не только с точки зрения памяти, но и потому, что поиск ключа в наборе данных может потребовать нескольких дорогостоящих сравнений ключей.
* Очень короткие ключи часто не являются хорошей идеей. Нет смысла писать «u1000flw» в качестве ключа, если вместо этого вы можете написать «user:1000:followers». Последний более удобочитаем, а добавленное пространство незначительно по сравнению с пространством, используемым самим ключевым объектом и объектом-значением. Хотя короткие ключи, очевидно, будут потреблять немного меньше памяти, ваша задача — найти правильный баланс.
* Старайтесь придерживаться схемы. Например, "object-type:id" - хорошая идея, например, "user:1000". Точки или тире часто используются для полей, состоящих из нескольких слов, например, «comment:4321:reply.to» или «comment:4321:reply-to».
* Максимально допустимый размер ключа составляет 512 МБ.

**String (строка)**

Тип String — это самый простой тип значения, который можно связать с ключом Redis.

C помощью команд SET и GET мы можем устанавливать и получать строковые значения. Обратите внимание, что это SET заменит любое существующее значение, уже сохраненное в ключе, в случае, если ключ уже существует, даже если ключ связан с нестроковым значением.

Значения могут быть строками (включая двоичные данные) любого типа, например, внутри значения можно хранить изображение в формате jpeg. Значение не может превышать 512 МБ.

Изображение выглядит как текст

Автоматически созданное описание

Рис. 5. Выполнение команд «*set*» и «*get*»

Возможность установить или получить значение нескольких ключей в одной команде также полезна для уменьшения задержки. По этой причине существуют команды MSET и MGET:

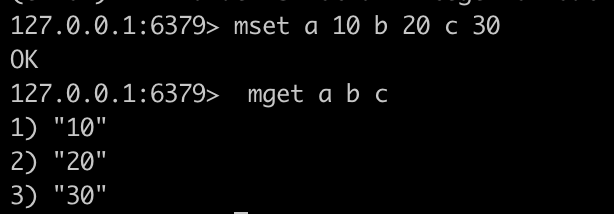


Рис. 6. Выполнение команд «m*set*» и «m*get*»

Команда INCR анализирует строковое значение как целое число, увеличивает его на единицу и, наконец, устанавливает полученное значение в качестве нового значения.

Изображение выглядит как текст

Автоматически созданное описание

Рис. 7. Выполнение команд «*incr*»

Существуют и другие подобные команды, такие как INCRBY, DECR и DECRBY . Внутри это всегда одна и та же команда, действующая немного по-разному.

Изображение выглядит как текст

Автоматически созданное описание

Рис. 8. Выполнение команд «*incrby*»

INCR является атомарным, это значит, что даже несколько клиентов, выдающих INCR для одного и того же ключа, никогда не попадут в состояние гонки. Например, никогда не случится так, что клиент 1 прочитает «10», клиент 2 одновременно прочитает «10», оба увеличиваются до 11 и устанавливают новое значение 11. Окончательное значение всегда будет 12, а прочитанное операция increment-set выполняется, пока все остальные клиенты не выполняют команду одновременно.

**Команды, не привязанные к конкретному типу**

Есть команды, которые не определены для конкретных типов, но полезны для взаимодействия с пространством ключей и, таким образом, могут использоваться с ключами любого типа.

Например, команда EXISTS возвращает 1 или 0, чтобы указать, существует ли данный ключ в базе данных или нет, в то время как команда DEL удаляет ключ и связанное с ним значение, каким бы оно ни было.

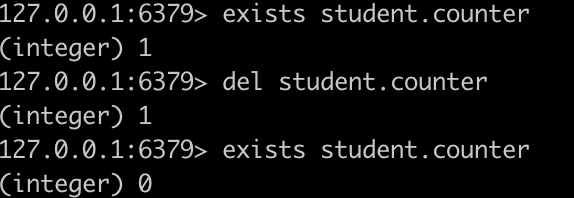


Рис. 9. Выполнение команды «*exists*» и «*del*»

Существует множество команд, связанных с пространством ключей, но две приведенные выше являются основными вместе с командой TYPE, которая возвращает вид значения, хранящегося в указанном ключе:

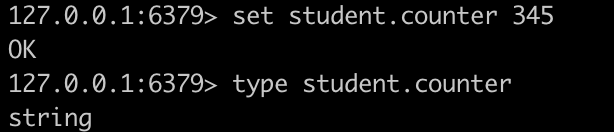
****

Рис. 10. Выполнение команды «type»

Чтобы перечислить ключи в хранилище данных Redis, используйте команду KEYS, за которой следует определенный шаблон. Redis выполнит поиск всех ключей, соответствующих указанному шаблону.

Мы можем использовать звездочку (\*) для поиска всех ключей в хранилище данных.

Изображение выглядит как текст

Автоматически созданное описание

Рис. 11. Выполнение команды «keys»

**Срок действия ключа**

Срок действия ключа позволяет установить время ожидания для ключа, также известное как «время жизни» или «TTL». Когда время жизни истекает, ключ автоматически уничтожается.

Несколько важных замечаний по поводу истечения срока действия ключа:

* Они могут быть установлены как с точностью до секунд, так и с точностью до миллисекунд.
* Информация об истечении срока действия реплицируется и сохраняется на диске, время фактически проходит, когда ваш сервер Redis остается остановленным (это означает, что Redis сохраняет дату истечения срока действия ключа).

Используйте команду EXPIRE для установки срока действия ключа:

Изображение выглядит как текст

Автоматически созданное описание

Рис.12. Выполнение команды «*expire*» для установки срока действия ключа

Также мы можем создавать ключи с истекающим сроком действия, используя команду SET  <key> <value> EX <seconds>. Позже c помощью TTL <key> вызывается команда для проверки оставшегося времени жизни ключа:

Изображение выглядит как текст

Автоматически созданное описание

Рис. 13. Выполнение команды «*ttl*» для проверки оставшегося времени жизни ключа

**Hashes (хэши)**

Hashes Redis — это типы записей, структурированные как коллекции пар поле-значение.

Количество полей, которые вы можете поместить в хэш, не имеет ограничений (кроме доступной памяти), поэтому вы можете использовать хэши по-разному в своем приложении.

Команда HSET устанавливает несколько полей хеша, а HGET извлекает одно поле. HMGET аналогичен, HGET, но возвращает массив значений:

Изображение выглядит как текст

Автоматически созданное описание

Рис. 14. Выполнение команд «*hset*», «*hget*», «*hgetall*»

Изображение выглядит как текст

Автоматически созданное описание

Рис. 15. Выполнение команд «*hmget*»

Существуют команды, которые также могут выполнять операции с отдельными полями, например HINCRBY:

Изображение выглядит как текст

Автоматически созданное описание

**Примеры использования:**

* Представьте базовый профиль пользователя в виде хэша:
* Хранить счетчики количества раз, когда пользователь заходил на сервер

**Производительность**

Большинство хэш-команд Redis имеют сложность O(1).

Некоторые команды, такие как HKEYS, HVALS, и HGETALL, имеют сложность O(n), где n — количество пар поле-значение.

**Практическое задание**

1. Продемонстрировать основные строковые операции:
   1. В качестве ключа необходимо указать свою группу и уникальный номер студента (например, m2000001) в формате group:<groupName>:sudent:<studentNumber>, а в значении строку в формате JSON-объект с следующими полями: Фамилия, Имя, Год рождения.
   2. Получить значение по ранее созданному ключу.
   3. Узнать тип, хранящийся в ключе
   4. Задать время жизни ключа равное 600 секундам.
   5. Вывести оставшееся время жизни ключа.
   6. Не дожидаясь автоматического удаления удалить запись по ключу вручную.
2. Продемонстрировать основные операции с хэшем:
   1. В качестве ключа необходимо указать свою группу и уникальный номер студента (например, m20000001) в формате group:<groupName>:sudent:<studentNumber>, в хэше необходимо задать следующие поля: name, surname, birthyear, заполнив их в соответствии со своим именем, фамилией, годом рождения;
   2. Вывести все поля и значения по созданному ключу;
   3. Узнать тип, хранящийся в ключе;
   4. Увеличить поле birthyear на 2;
   5. Выставить время жизни ключа равное 40 секундам

**Теоретические вопросы**

1. Что такое Redis?
2. Для каких задач используется redis?
3. Что такое redis-cli?
4. Что такое ключ в Redis?
5. Как получить список всех ключей?
6. Какой командой можно задать «время жизни» ключа в Redis? Каким аргументом можно задавать время жизни?
7. Какой командой можно узнать оставшееся «время жизни» ключа в Redis?
8. Что такое строки (Strings) в Redis? Перечислите основные команды для работы со строками.
9. Какие команды используются для инкрементации и декрементации в Redis?
10. Что такое хэши (hashes) в Redis? Перечислите основные команды для работы с хэшем.