МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА – Российский технологический университет»
РТУ МИРЭА

ЛЕКЦИОННЫЕ МАТЕРИАЛЫ

Технологии хранения в системах кибербезопасности

(наименование дисциплины (модуля) в соответствии с учебным планом)

Уровень	бакалавриат
	(бакалавриат, магистратура, специалитет)
Форма обучения	очная
	(очная, очно-заочная, заочная)
Направление(-я) подготовки	10.05.04 Информационно-аналитические системы безопасности
	(код(-ы) и наименование(-я))
Институт	Кибербезопасности и цифровых технологий (ИКБ)
	(полное и краткое наименование)
Кафедра	КБ-2 «Прикладные информационные технологии»
	(полное и краткое наименование кафедры, реализующей дисциплину (модуль))
Лектор	к.т.н., Селин Андрей Александрович
	(сокращенно – ученая степень, ученое звание; полностью – ФИО)

Используются в данной редакции с учебного года 2024/2025

(учебный год цифрами)

Проверено и согласовано «____» _____2024 г. _____ А.А. Бакаев

(подпись директора Института/Филиала с расшифровкой)

Москва 2024 г.

Технологии хранения в системах кибербезопасности

2024 год

Лекция 10.

БД Redis

Учебные вопросы лекции:

1. **Традиционный подход к использованию Redis**
2. **Архитектура Redis**
3. **Примеры использования Redis**

Введение

Redis – это БД, размещаемая в памяти, используемая, в основном, в роли кэша, находящегося перед другой, «настоящей» БД, вроде MySQL или PostgreSQL. Кэш, основанный на Redis, помогает улучшить производительность приложений. Он эффективно использует скорость работы с данными, характерную для памяти, и смягчает нагрузку центральной БД приложения, связанную с обработкой следующих данных:

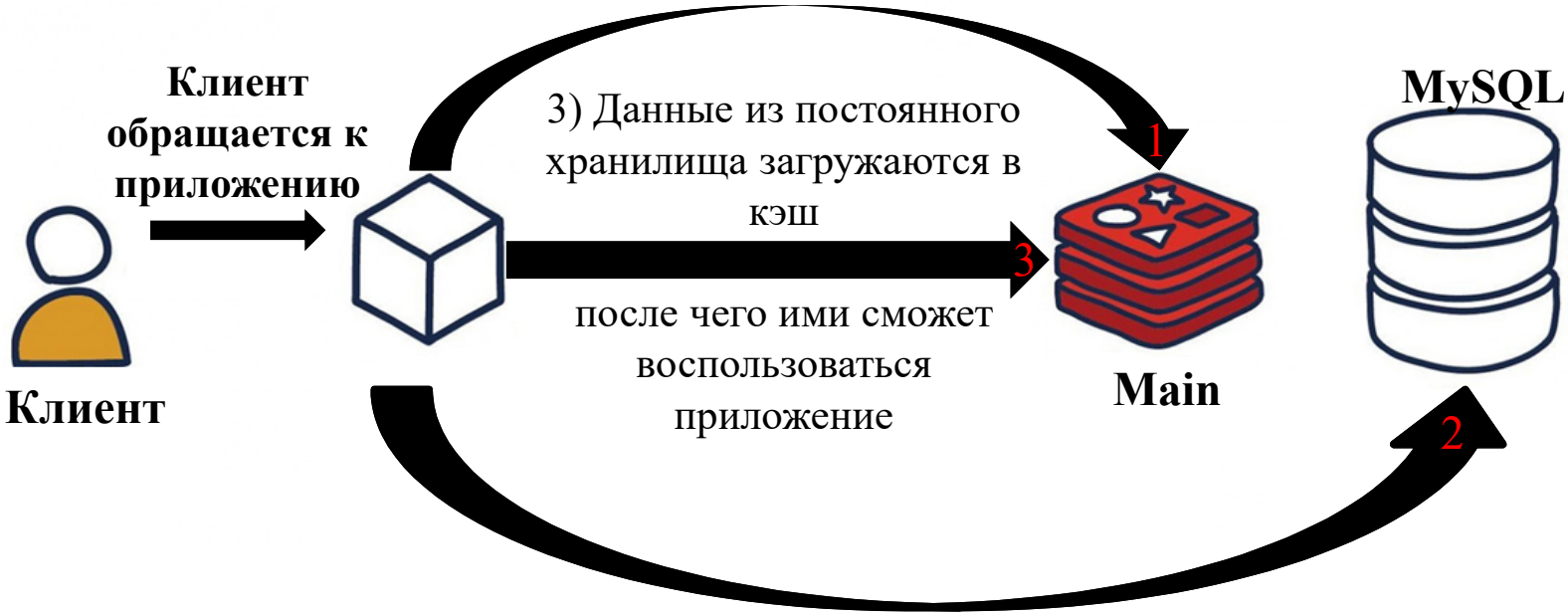
- данные, которые редко меняются, к которым часто обращается приложение;
- данные, не относящиеся к критически важным, которые часто меняются.

Примеры таких данных могут включать в себя сессионные кэши или кэши данных, а также содержимое панелей управления – вроде списков самых популярных новостей и отчётов, включающих в себя данные, агрегированные из разных источников.

Точнее всего описать Redis можно, сказав, что это – сервер структур данных. Уникальные особенности сервера Redis являются основной причиной его популярности.

Традиционный подход к использованию Redis

1) Приложение обращается к кэшу Redis, представленному главной базой данных. Если данные в кэше есть, произошло попадание кэша, выполняется обычный возврат данных.



2) Если произошёл промах кэша система обращается к постоянному хранилищу

Ещё одна важная особенность Redis заключается в том, что эта СУБД размывает границы между кэшем и хранилищем данных. Чтение данных из памяти и работа с данными, находящимися в памяти, гораздо быстрее чем те же операции, выполняемые традиционными СУБД, использующими обычные жёсткие диски (HDD) или твердотельные накопители (SSD).

Архитектура Redis. Единственный экземпляр Redis

Различные варианты развёртывания этого хранилища:

1. Единственный экземпляр Redis.
2. Redis HA.
3. Redis Sentinel.
4. Redis Cluster.

1. Единственный экземпляр Redis

Самый простой вариант развёртывания Redis – это использование одного экземпляра системы. При таком подходе в распоряжении пользователя оказывается небольшое хранилище, которое поможет проекту расти и развиваться и ускорит сервисы этого проекта.

Запросы, поступающие к базе данных Redis, обрабатываются путём работы с данными, хранящимися в памяти. Если используемый экземпляр Redis предусматривает применение постоянного хранения данных, в системе будет форк процесса. Он использует RDB (Redis Database) для организации постоянного хранения снепшотов данных – это компактное представление данных Redis на определённый момент времени.

Вместо RDB могут использоваться файлы, предназначенные только для добавления данных (Append-Only File, AOF).

Архитектура Redis. Redis HA

Другой популярной конфигурацией Redis является система, состоящая из ведущего и подчинённого узлов, состояние которых синхронизируется посредством репликации. По мере того, как данные записываются в ведущем экземпляре Redis, копии соответствующих команд отправляются в выходной буфер для подчинённых узлов, что обеспечивает выполнение репликации данных.

«Высокая доступность» (HA, High Availability) – это характеристика системы, которая нацелена на обеспечение согласованного уровня показателей её деятельности (обычно – времени безотказной работы системы) на временных интервалах, превышающих средние.

В системах высокой доступности важно отсутствие единой точки отказа. Это позволяет им корректно и быстро восстанавливать нормальную работу после сбоев.

Конфигурации высокой доступности предусматривают наличие надёжных линий связи, что исключает потерю данных при их передаче от ведущего узла к подчинённому. В таких системах осуществляется автоматическое обнаружение сбоев и восстановление после них.

Репликация данных в Redis

Каждый ведущий узел Redis имеет идентификатор (ID) и смещение репликации (offset). Эти два показателя нужны для установления момента времени, когда подчинённый узел может продолжать процесс репликации, или чтобы определить, что необходимо выполнить полную синхронизацию данных. Смещение инкрементируется при выполнении любого действия, которое происходит в ведущем узле Redis.

Replication ID, offset

Когда подчинённый узел Redis отстаёт лишь на несколько шагов смещения от ведущего узла, он получает оставшиеся необработанными команды от ведущего узла, эти команды применяются к данным, так происходит до момента синхронизации узлов.

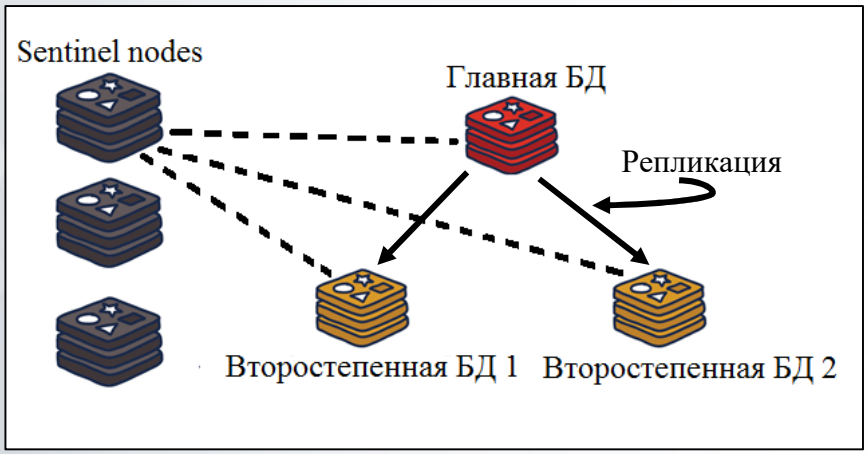
Пример: два экземпляра Redis, ведущий и подчинённый, имеют один и тот же ID репликации, а их смещения отличаются на несколько сотен команд. То есть – если на «отстающем» экземпляре воспроизвести соответствующие команды, в распоряжении обоих экземпляров будет идентичный набор данных.

Предположим, что ID репликации экземпляров различаются и неизвестен предыдущий ID (у экземпляров нет общего предка) узла, уровень которого недавно понижен до подчинённого (он подключён к ведущему узлу). В такой ситуации нужно выполнить ресурсозатратную операцию полной синхронизации данных.

Иначе, если предыдущий ID репликации известен – мы можем синхронизировать данные двух узлов. Так как известен общий предок узлов, то они хранят общие данные, а значит, воспользовавшись смещением, можно провести частичную синхронизацию данных.

Архитектура Redis. Redis Sentinel

Redis Sentinel – это сервис, обеспечивающий создание распределённых систем. Как и в случае со всеми распределёнными системами, у Sentinel имеются и сильные, и слабые стороны. В основе Sentinel лежит кластер Sentinel-процессов, работающих совместно.



Они координируют состояние системы, реализуя конфигурацию высокой доступности Redis. Sentinel – это сервис, защищающий хранилище Redis от сбоев. Поэтому логично то, чтобы этот сервис не имел бы единой точки отказа.

Сервис Sentinel решает несколько задач.

Во-первых – он обеспечивает работоспособность и доступность текущих ведущих и подчинённых узлов.

Благодаря этому текущий Sentinel-процесс (вместе с другими подобными процессами) может отреагировать на ситуацию, когда теряется связь с ведущими и/или подчинёнными узлами.

Во-вторых – он играет определённую роль в деле обнаружения сервисов. То есть – когда новый клиент пытается что-то записать в хранилище Redis, Sentinel сообщит клиенту о том, какой экземпляр Redis в этот момент является ведущим.

Архитектура Redis. Redis Sentinel

Кворум – это минимальное число голосов, которое нужно получить распределённой системе для того, чтобы ей было позволено выполнять определённые операции, наподобие восстановления после сбоя. Это число поддаётся настройке, но оно должно отражать количество узлов в рассматриваемой распределённой системе. Размеры большинства распределённых систем равняются трём или пяти узлам, в них, соответственно, кворум равен двум или трём голосам. Нечётные количества узлов предпочтительны в случаях, когда системе необходимо разрешать неоднозначности.

3. Управление восстановлением системы после отказа. Узлы Sentinel могут запустить процесс восстановления системы после сбоя в том случае, если ведущий экземпляр Redis недоступен и достаточное количество (кворум) узлов согласно с тем, что это так.

4. Управление конфигурацией. Узлы Sentinel, кроме того, играют роль системы, позволяющей обнаруживать текущий ведущий экземпляр Redis.

Использование Redis Sentinel для решения вышеописанных задач позволяет обнаруживать сбои Redis. Процедура обнаружения сбоя включает в себя получение согласия нескольких узлов Sentinel с тем, что текущий ведущий экземпляр Redis недоступен. Процесс получения такого согласия называют **кворумом (quorum)**. Это позволяет повысить надёжность системы, защититься от ситуаций, когда один из процессов ведёт себя неправильно и не может подключиться к ведущему узлу Redis.

Архитектура Redis. Redis Sentinel

Sentinel можно запустить и на тех же машинах, на которых работают экземпляры Redis, или даже в виде независимых узлов, но это, в различных формах, усложняет ситуацию. Рекомендуется применять как минимум три узла с кворумом, как минимум, из двух.

Количество серверов	Кворум	Количество допустимых отказов
1	1	0
2	2	0
3	2	1
4	3	1
5	3	2
6	4	2
7	4	3

Есть несколько способов уменьшить масштабы потерь данных в том случае, если принудить ведущий экземпляр Redis к репликации операций записи на как минимум один подчинённый экземпляр. Репликация в Redis выполняется асинхронно, и у неё есть свои недостатки. Поэтому необходимо независимо отслеживать подтверждения получения данных, а при невозможности получить подтверждение от хотя бы одного подчинённого узла, главный узел должен прекратить принимать запросы на запись данных.

Redis

Достоинства Redis:

Высокая производительность – данные хранятся в оперативной памяти сервера, что значительно ускоряет работу с ними.

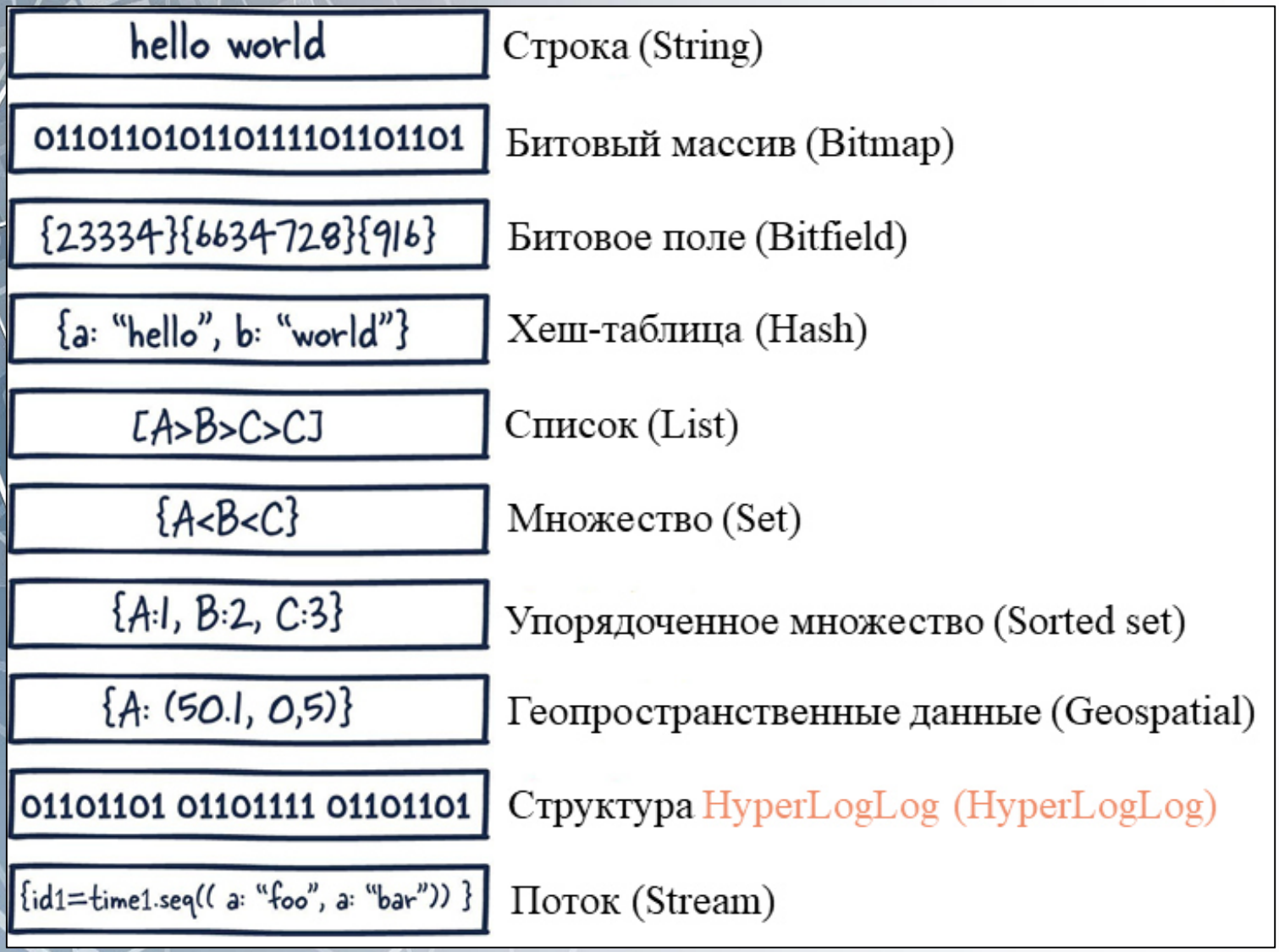
Персистентность – есть возможность сохранить снимки БД на диск в зависимости от количества обновлённых значений. Также есть режим AOF – сохранение на диск журнала операций с возможностью восстановить из него данные при следующем запуске.

Удобство – есть встроенная поддержка широко используемых структур данных (строки, списки, хеши, множества, сортированные множества).

Масштабируемость – поддерживается шардирование, репликация, отказоустойчивость и другие возможность распределённых систем.

Redis и типы данных

Redis хранит пары «ключ и значение». Ключ – строковый литерал, а в качестве значения могут выступать:



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.

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.

Игровые таблицы лидеров

Разработчики игр 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.

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.

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

Примеры использования Redis

Работа с геопространственными данными

Redis предлагает специально разработанные операторы и структуры данных в памяти для управления поступающими в режиме реального времени геопространственными данными в нужном масштабе и с высокой скоростью. Такие команды, как GEOADD, GEODIST, GEORADIUS и GEORADIUSBYMEMBER, предназначенные для хранения, обработки и анализа геопространственных данных в режиме реального времени, позволяют Redis просто и быстро выполнять геопространственные операции.

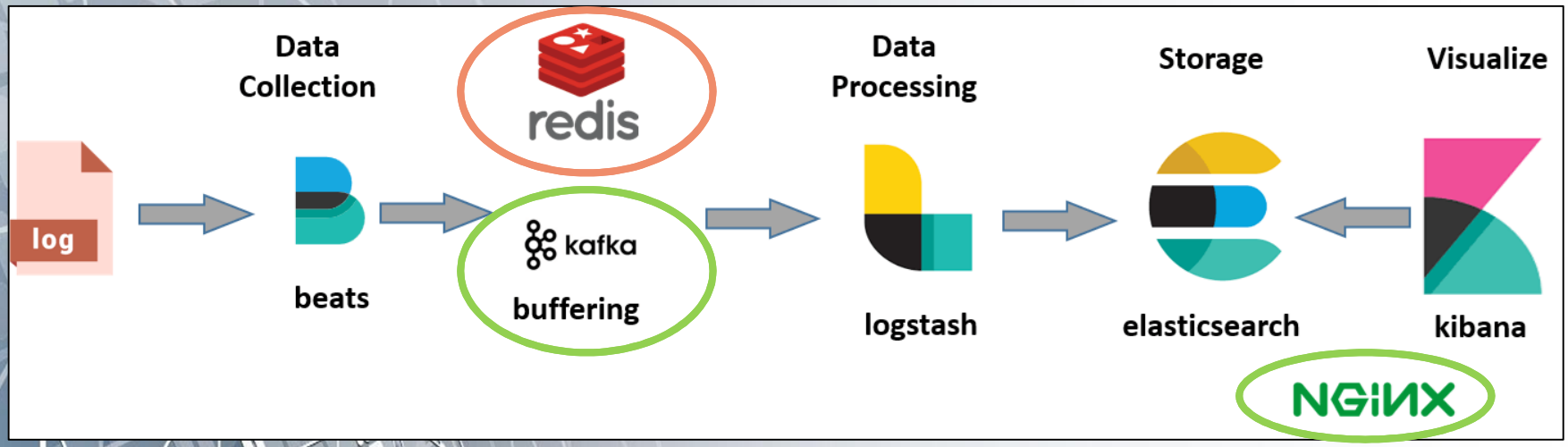
Машинное обучение (Machine Learning)

Чтобы быстро обрабатывать огромные объемы разнообразных данных, передаваемых на большой скорости, и автоматизировать принятие решений, современным приложениям, управляемым данными, требуется машинное обучение. Redis предоставляет скоростное хранилище данных в памяти, обеспечивающее быстрое создание, обучение и развертывание моделей машинного обучения.

Аналитика в режиме реального времени

Redis может использоваться с решениями потоковой передачи, такими как Apache Kafka и Amazon Kinesis, в качестве хранилища данных в памяти для сбора, обработки и анализа данных в режиме реального времени с задержкой на уровне долей миллисекунды. Redis – идеальный выбор для аналитики в режиме реального времени в таких примерах использования, как аналитика в социальных сетях, рекламный таргетинг, персонализация контента и IoT.

Примеры использования Redis



Redis достаточно легко встраивается в качестве дополнительного инструмента в стек ELK и используется, в основном, в роли кэша. Кэш, основанный на Redis, помогает повысить производительность

Ключевая задача, которую решает Kafka - организация потока данных из одной точки в другую и их опциональная обработка. Он может работать с разными видами событий – метрики, логи, данные систем мониторинга и другое.

NGINX используется:

для обработки запросов с сайтов, с большим количеством статического контента;

обслуживания серверов, на которые поступает много запросов одновременно;

в качестве прокси, почтового сервера или для распределения нагрузки на серверную часть.

Языковая поддержка Redis

Redis поддерживает большинство ведущих языков программирования и протоколов:

Python;

Java;

PHP;

Perl;

Go;

Ruby;

C/C#/C++;

JavaScript;

Node.js.

Заключение

Redis редко используется как основное хранилище в крупных системах: оптимальной производительности можно добиться при хранении всего объёма данных в оперативной памяти, а для значительного числа хранилищ это невозможно из-за технических ограничений на её размер.

Кроме того, стоимость хранения в оперативной памяти всё ещё дороже стоимости хранения на диске. Наибольшая производительность – а это главное преимущество Redis – достигается при выключенной персистентности, что создаёт риск потерять данные. Асинхронная репликация же, которая применяется в Redis и позволяет быстрее записывать данные, даёт некоторую постоянную задержку достижения консистентности данных между мастером и репликой, что допустимо не во всех сценариях использования.

БД Redis используют социальные сети, сервисы путешествий, различные форумы и электронная коммерция (Twitter, Uber, Slack, Airbnb, Pinterest…и др.)

СПАСИБО ЗА ВНИМАНИЕ!