

Способ уменьшения задержки - кеширование

Кеширование — это метод ускорения доступа к данным путем их временного хранения в быстродоступных хранилищах (кеши).

Кеширование можно использовать на разных уровнях системы. На разном уровне уменьшается соответственно задержка либо сети, либо ответа сервера, либо ответа базы данных.

Кеширование на уровне CDN

CDN (Content Delivery Network) хранит копии контента с основного сервера для быстрой доставки клиенту (CDN сервера расположены по всему миру, и, если пользователь живёт далеко от ваших серверов - CDN значительно уменьшит задержку в его случае).

Как это работает?

Когда пользователь запрашивает контент, CDN сначала проверяет, есть ли у него закешированная версия. Если есть, контент доставляется из кеша, сокращая время загрузки и снижая нагрузку на основной сервер.

Преимущества и риски

- **Преимущества:** Ускорение загрузки контента, снижение нагрузки на основные серверы.
- **Риски:** Сложность инвалидации кеша, возможность хранения устаревших данных.

Кеширование на уровне балансировщика

Балансировщики нагрузки могут кешировать часто запрашиваемые данные для быстрого ответа, не перенаправляя запрос на бэкенд-серверы.

Как это работает?

Когда приходит запрос, балансировщик сначала проверяет, есть ли эти данные в кеше. Если да, он возвращает их, иначе перенаправляет запрос на один из бэкенд-серверов.

Преимущества и риски

- **Преимущества:** Снижение задержки и нагрузки на бэкенд.
- **Риски:** Возможность кеширования нежелательных или устаревших данных.

Кеширование на уровне фронтенда (браузера)

Браузеры кешируют статический контент (CSS, JS, изображения) для ускорения загрузки страниц.

Как это работает?

При первом посещении сайта браузер сохраняет статические файлы в локальном кеше. При повторных посещениях эти файлы загружаются из кеша.

Преимущества и риски

- **Преимущества:** Ускорение загрузки страницы, снижение трафика.
- **Риски:** Возможность хранения устаревшего контента, необходимость инвалидации кеша.

Кеширование на уровне бэкенд приложения сервера (RAM)

Приложения могут кешировать данные в оперативной памяти для быстрого доступа.

Как это работает?

Часто запрашиваемые данные или результаты сложных вычислений сохраняются в RAM, что обеспечивает быстрый доступ к ним.

Преимущества и риски

- **Преимущества:** Очень быстрый доступ к данным.
- **Риски:** Ограниченный объем оперативной памяти, возможность потери данных при перезагрузке.

Кеширование на уровне дополнительного хранилища (SSD)

Можно создать дополнительное хранилище с быстродействующим SSD для кеширования часто запрашиваемых данных.

Как это работает?

Часто запрашиваемые данные или таблицы хранятся на SSD, что обеспечивает более быстрый доступ по сравнению с HDD (основной базой данных).

Преимущества и риски

- **Преимущества:** Увеличение скорости доступа к данным.
- **Риски:** Высокая стоимость SSD, ограниченный объем хранилища.

Вот примеры, что можно хранить в кеше на разных уровнях веб-приложения:

CDN (Content Delivery Network)

1. **Статические ресурсы:** Файлы, которые редко изменяются, такие как CSS, JS и изображения.
2. **Медиа контент:** Видео, аудио файлы.
3. **Веб-страницы:** HTML страницы, которые не требуют частых обновлений.

Кеш балансировщика нагрузки

1. **Сессионные данные:** Информация о текущей сессии пользователя для быстрого доступа.
2. **SSL/TLS сессии:** Кеширование ключей для ускорения процесса рукопожатия.
3. **IP-адреса:** Чтобы быстро направлять повторные запросы с того же IP на тот же сервер.

Кеш фронтенда (Client-Side Caching)

1. **Локальное хранилище:** Кеширование пользовательских настроек, корзины покупок и другой некритичной информации.

2. **Service Workers:** Кеширование API-запросов и ресурсов для работы в оффлайн-режиме.
3. **HTTP Caching:** Использование заголовков HTTP для кеширования статических ресурсов.

Кеш бэкенда (Server-Side Caching)

1. **Результаты запросов к БД:** Кеширование результатов SQL-запросов, которые часто повторяются.
2. **Полные HTML страницы или их фрагменты:** Чтобы не генерировать их каждый раз.
3. **API-запросы:** Кеширование ответов от внешних сервисов.
4. **Данные из распределенных источников:** Например, результаты запросов к другим микросервисам.

Отдельная быстрая база данных для кеширования (например, Redis)

1. **Сессионные данные:** Хранение данных о пользовательской сессии для быстрого доступа.
2. **Ключ-значение:** Хранение промежуточных результатов вычислений, временных данных.
3. **Очереди и стеки:** Хранение данных, которые будут обработаны асинхронно.
4. **Топовые списки:** Например, топ статей, продуктов и так далее, чтобы не считать их каждый раз.