**Важно.**

Я исхожу из следующей ситуации.

Пользовательский интерфейс в состоянии (т.е. физически рассчитан) отобразить увеличившийся список заказов. Если нет, ни каким кешированием без переработки интерфейса мы ситуацию не улучшим.

Второе. У нас уже реализовано реплицирование, есть быстрые реплики на чтение.

Если после этого ситуация требует кеширования (помним кеш – агрегированные данные и могут не соответствовать таблица БД), рассматриваем серверное кеширование.

В вариантах также будет клиентское кеширование.

И еще одна отмазка. Любое кеширование, какое бы не выбрала компания может привести к отображению пользователю устаревшую информацию. Так как у нас хотя и получение заказов (кол-во) через API контролируются Backpressure / Circuit Breaker , все равно список заказов будет меняться очень динамично.

Поэтому если репликация дала пристойный результат, я бы предложил «собрать обратную связь, провести исследование по фактическим результатам и определиться с необходимой детализации, поставить в план доработок и т.п.»

И только потом очень аккуратно приступить к кешированию.

1. **Проанализируйте диаграмму системы и её описание.**

У нас нет реплик и ни каких доработок.

Исходные данные

Нам нужен быстрый кеш на чтение, у нас много обновлений, обновлять придется кеш (так как запросы от операторов идут на постоянной основе и заказы сыпяться тоже) - Read-Through и Write-Through наш выбор.

По поводу Write-Behind. Мне не очень нравится писать в БД с задержками, по причине возможных проблем с падением кеша. Я понимаю, что Redis при рестарте поднимет все, что сможет. Но в данном случае риск больше, так как в кеше копится пул изменений. Но это мое личное мнение. На проектах видел использование этого подхода. Т.ч. повод для обсуждения с админом БД и другими коллегами.

Нам нужно закешировать весь список заказов для оператора, т.к. согласно заданию отображается весь список в интерфейсе, а уж потом на него накладывается фильтр. Т.е. массив данных для фронта все равно должен содержать все заказы.

Без системы кэширования, сервер MES приходится обращаться к базе данных для получения списка (в нашем случае всех заказов) заказов при каждом запросе оператора к главной странице интерфейса списку заказов. Это занимает время и ресурсы.

1. **Добавьте в документ раздел «Мотивация».**

Преимущества кэширования

Повышение производительности приложения в нашем случае MES

Сокращение затрат на базы данных

Прогнозируемая производительность

Решение проблем с заказами в компании.

1. **Добавьте раздел «Предлагаемое решение».**

Как уже писал выше.

Предложение внедрить серверное кеширование.

При последующих запросах сервер MES извлекает список из кэша, значительно ускоряя ответ, само собой отображение страницы для пользователей ускоряется и снижается нагрузка на базу данных.

Используем паттерны Read-Through и Write-Through.

Все запросы на чтение и запись идут через кеш.



1. **В блоке «Предлагаемое решение» опишите стратегию инвалидации кеша, которую вы планируете использовать.**

Предлагаю использовать инвалидацию, основанную на запросах.

Кеш инвалидируется каждый раз, когда приходит запрос на обновление данных. А по процессу у нас будут шаги (взял в работу, в работе, закончил работу и тп.), которые будут отправлять соответствующий запрос.

Способ обеспечивает надёжность и актуальность данных, основываясь на активности и запросах пользователей.

Это самый подходящий вариант.

**Дополнительное задание.**

Если вы считаете, что может быть несколько решений и вам сложно выбрать между ними, можете описать несколько вариантов.

**Варианты.**

Клиентская кеширование.

Исходные данные

Раньше страница показывала все заказы, но это тормозило загрузку. Команда сделала фильтр по статусам и пагинацию, но это не помогло.

Операторам важно видеть самые новые заказы, потому что от этого зависит их вознаграждение, — кто взял заказ, тот и получит оплату.

Вариант проверки на производстве.

1. На производстве на машине оператора в режиме разработчика проверить формирование списка заказов. Варианты (описан для web интерфейса).
   1. Запрос «все заказы» отрабатывает корректно и в умеренные сроки. После этого даже с фильтром веб интерфейс строит список дольше ожидаемого (нужно для рассмотрения клиентского кеширования, даже при условии, что это может привести к попыткам оператора взять устаревшую запись о заказе, которая будет в работе).

Что смущает. Я не профи во фронте. Но если интерфейс писал под все заказы, он и получает и обрабатывает массив всего списка, а уже потом на экран отображает с учетом всех фильтров и доп. настроек.

Поэтому ситуация гипотетическая – мы все проверили, проблему с потерями поправили например увеличив кол-во переотправлений сообщений в MQ. И у нас по факту проблема только с криво написанным интерфейсом.

Получаем подтверждение отр разработчиков.

Пользовательский интерфейс в состоянии (т.е. физически рассчитан) отобразить увеличившийся список заказов. Если нет, ни каким кешированием без переработки интерфейса мы ситуацию не улучшим.

Существует возможность ограничить / лимитировать вывод массива данных. Для ситуации, когда первое утверждение не верно.

После этого можно рассмотреть клиентское кеширование.

**HTTP-cache, или клиентское кеширование,** не требует установки и разработки дополнительных компонентов и работает прямо в браузере.

Все HTTP-запросы сначала направляются в кеш браузера, чтобы проверить, существует ли действительный кешированный ответ, который можно использовать для выполнения запроса. Если есть совпадение, ответ считывается из кеша, что устраняет как задержку в сети, так и затраты на передачу данных.

Это должно помочь с оптимизацией загрузки ресурсов, улучшает производительность приложений и уменьшает нагрузку на серверы.

Обязательно используем

Заголовок no-cache (запрос актуальности запрашиваемого ресурса).

Чтобы это сделать, клиент должен отправить условный запрос на проверку соответствия — conditional request — на сервер.

Вариант If-Modified-Since - сравнение времени модификации.