1. **Анализ схемы и описание системы.**

В задании описана цепочка работы над заказом (по статусам) только из Интернет магазина.

При создании заказа через API MES, заказ попадает в CRM через очередь MES.

Так же указано, что оператор сам выбирает поступивший заказ работу.

1. **Ограничения и отмазки**

Исхожу из того, что сами системы компании работают корректно, а теряются только сообщения между ними. Т.е. проблема интеграционная, ее и пытаемся решить.

Исхожу из того, что сами системы компании работают корректно, но после открытия доступа через API, не посчитав нагрузку, системы тормозят. Т.е вторая проблема –быстродействие и адаптация к возросшей нагрузке. Вторая проблема имеет прямое отношение к первой. Ниже поясню

Да. И я не рассматриваю инициативу переписать все, исхожу из того, что CRM, MES и интернет магазин должны меняться минимально для решения этих 2-х проблем.

1. **Анализ проблемы через логику.**

Если рассмотреть статусы по работе заказа, то работа по заявке от взятия в работу и до отправки, производятся на производстве и фиксируются в MES. Т.е. если оператор смог выбрать заказ и взять его в работу, то даже при потере промежуточных статусов, клиент не получит уведомлений, но курьер постучится к нему в дверь.

Проблемы с просрочкой были и раньше, до введения в действие API MES. Ввод API MES не является причиной проблем, но он усугубил проблему. И привел к затруднению работы оператора, увеличив нагрузку на MES.

Т.е. проблемное место с потерями цепочка заказа от интернет магазина через CRM в MES при большой загрузке MES.

Варианты.

1. Заказ создан в CRM, но не попал в MES (из-за большой нагрузки). Интеграция
2. Заказ создан в обеих системах, но оператор не может взять его в работу, из-за работы раздела список всех заказов. Нагрузка.
3. **Существующие и потенциальные проблемные места.**

Список проблем определенных заданием.

1. Софт MES.

Не оптимизированный расчет стоимости. Слишком большой диапазон времени расчета от 2-3 до 30 минут.

1. MES и CRM взаимодействуют друг с другом через RabbitMQ. Выбор, как всегда между скоростью RabbitMQ и надежностью Kafka (Если бы использовалась Kafka, потери заказов можно было бы попытаться определить по записям в БД Kafk-и.)
2. Нагрузка после ввода API растет не линейно. Но интернет магазин продолжает приносить на 100 заказов больше каждый месяц. Не продуманы возможные проблемы при увеличении нагрузки. API MES грузит MES систему, не проанализированы потоки и кол-во обращений.
3. Судя по всему проблемы с заказами были всегда, но когда заказов было меньше, не обращали на это внимание или разбирались в ручном режиме.
4. Есть еще одно возможное узкое место, но оно вне моей компетенции. Каналы связи на производстве. Возможна ситуация что каналы до рабочих мест операторов не рассчитаны на возросшую нагрузку. Тогда все нижеописанные действия не имеют смысла. Клиентское кеширование позволит улучшить ситуацию, но полностью не спасет.
5. **Инициативы, которые необходимы для устранения нежелательных ситуаций.**

Понятно, что по хорошему нужно собрать метрики, визуализировать состояние программного комплекса, провести анализ, после этого вносить правки.

Но по факту компания теряет клиентов, а значит деньги, и руководство не даст спокойно разобраться в ситуации.

Поэтому порядок действия примерно такое.

**Правки на лету (быстрые победы)**

1. Проверить раздел Limits в конфиге контейнеров. Узнать хватает ли выделенных лимитов для текущей работы системы. Увеличить лимиты (вертикальное масштабирование).
2. 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).
   2. Запрос «все заказы» отрабатывает очень долго. После отработки запроса интерфейс с фильтром в умеренные сроки строит отражение списка заказов. (Проблема на сервере. Клиентское кеширование не нужно, как факт).
   3. Желательно проверить сеть.
3. Включить автоматическое распределение заказов на свободных операторов. По идее такой функционал должен быть в MES. Это снизит необходимость работы с полным списком заказов. (Вопрос с премированием не в моей компетенции как Архитектора).
4. Молимся, чтобы версия PostgreSQL была выше 10-й, идем r DevOps-у скрестив пальцы, что он умеет настраивать репликацию а PostgreSQL. Нам нужна простая логическая, используем Паттерн read-replica. Т.е все чтение запустим через реплику. Разделим нагрузку чтения и записи. У нас контейнеры. Проблем быть не должно. Заодно сможем понять, что по нагрузке.

**Минимальный набор метрик (сбор в ручную).**

Метрики DAU (количество пользователей в день), RPS (количество запросов, которые сервер приложения обрабатывает в секунду) и QPS (количество запросов в секунду к базе данных).

Метрики собираются по статусам и записям баз данных и логам серверов приложения и api (рассмотреть монтирование отдельных сетевых дисков в контейнер для сохранения логов, если потребуется. )

В нашем случае DAU скорее всего превратиться в кол-во заказов, оформленных, взятых в работу, завершенных и т.п. так как кол-во пользователей известно заранее.

**Backpressure / Circuit Breaker**

Желательно после получения предварительного (ручного) представления об узких местах (т.е. станет известна нагрузка на API и MQ и остальные системы).

Рассмотреть Backpressure (лимитирование скорости запросов) и Circuit Breaker на основе полученной статистики.

Так как нагрузка растет постоянно (API статистика по логам, а магазин + 100 ежемесячно), т.е. ситуация с потерями будет усугубляться, и будет усугубляться работа оператора.

Мы сократим общее количество заказов в момент времени и поставим лимиты на скорость запросов, что позволит дальше разобраться в ситуации.

Важно, это не решение проблемы на данный момент, а борьба с последствиями. На этапе тюнинга настроек после реализации «Наблюдаемости» может стать решением.

**Быстродействие MES**

1. Рассмотреть ограничение сложности загружаемых моделей, подобрать необходимую сложность модели для качественного и быстрого расчета стоимости для снижения нагрузки на MES (это так же снизит нагрузку на интернет магазин, но скорее всего не значительно).
2. Кеширование. Рассмотреть использование серверного кеширования для решения проблемы быстродействия сервера.

Клиентское кеширование рассмотреть (при условии 1-го варианта проверки формирования списка заказов на компьютере оператора).

Паттерны кеширования будут рассмотрены на более детальном этапе.

1. Быстрые реплики БД MES на чтение. Уже реализовано по плану.

Поясню - быстрая реплика ускорит чтение данных из БД, а кэшироваться будет хранить уже подготовленный / агрегированный набор данных для работы MES.

Решение принимать после анализа первичных метрик, полученных в ручную.

**Определиться с Observability.**

Идеальный вариант, если предыдущие действия возымели результат.

Мы все еще не решили проблем, но заказы создаются, операторы работают, дата Х отодвинута.

Теперь можно спокойно (в режиме ужаленного зайца) приступить к анализу и реализации «Наблюдаемости» системы, а по ее результатам приступить к дотюниванию всего программного комплекса.

1. Мониторинг — сбор бизнесовых и инфраструктурных метрик (metrics).

Метрики RED измеряют показатели, которые важны для конечных пользователей ваших сервисов.

Мне больше нравится RED , так как можно использовать для мониторинга веб-сервисов, а главное запросов к базам данных (PostgreSQL), очередей (RabbitMQ)

Однако понимаю, что сбор метрик и построение систем мониторинга отдельная серьезная задача (по факту мне мониторинг namespace в кубере месяц делали – и это только контроль namespace, без разбивки по подам).

Подробнее в разделе мониторинг, там определимся точно.

1. Логирование — сбор логов (logs).
2. Трейсинг — сбор трейсов внутренних вызовов (traces).

Визуализировать с помощью разных инструментов. Например, Grafana (мониторинг) или Kibana (ELK).

**Дальние перспективы**

После создания комплекса мониторинга и логирования дотюнить настройки Backpressure / Circuit Breaker до приемлемых.

Провести проверку других возможных узких мест, составить план по их преодолению.

В перспективе стоит масштабироваться — добавить больше серверов или отправить часть систем на ресурсы в облаке (гибридный подход), чтобы справляться с возросшей нагрузкой.

Рассмотреть вариант. Вынести расчет стоимости в отдельный сервис с возможностью горизонтального масштабирования, взаимодействие с сервером back построить на основе уже существующего API. Это позволит снять нагрузку расчёта с MES, а горизонтально масштабирование расчетных сервисов позволит управлять нагрузкой при резких скачках заказов (скидки, пятницы и т.п.)

1. **Инициативы в порядке приоритета.** Опишите ход своих рассуждений и ответьте на вопросы:
   * Какой вы видите целевую архитектуру через полгода?

Предполагаю гибридный облачный подход. (Так как стоимость инфраструктуры растет).

Используется серверное кеширование + реплицирование.

Клиентское кеширование рассмотрю в task5 Архитектурное решение по кешированию.

* + Если бы у вас была возможность выполнить только три пункта из списка инициатив в ближайшие полгода, что бы вы выбрали и почему? Не обязательно добавлять в список только эпики. Вы можете включить в план как крупные изменения, так и локальные задачи.

1. Минимальный набор метрик (сбор в ручную).
2. Backpressure / Circuit Breaker
3. Быстродействие MES

По факту мы остаемся без мониторинга, но скорее всего выживем.