Работа очередей сообщений, особенности запросов и ответов

Очереди предоставляют буфер, обеспечивающий временное хранение сообщений, и конечные точки, позволяющие в свою очередь подключаться к очереди в целях отправки/получения сообщений в асинхронном режиме. В таких сообщениях могут быть:

— запросы;

— ответы;

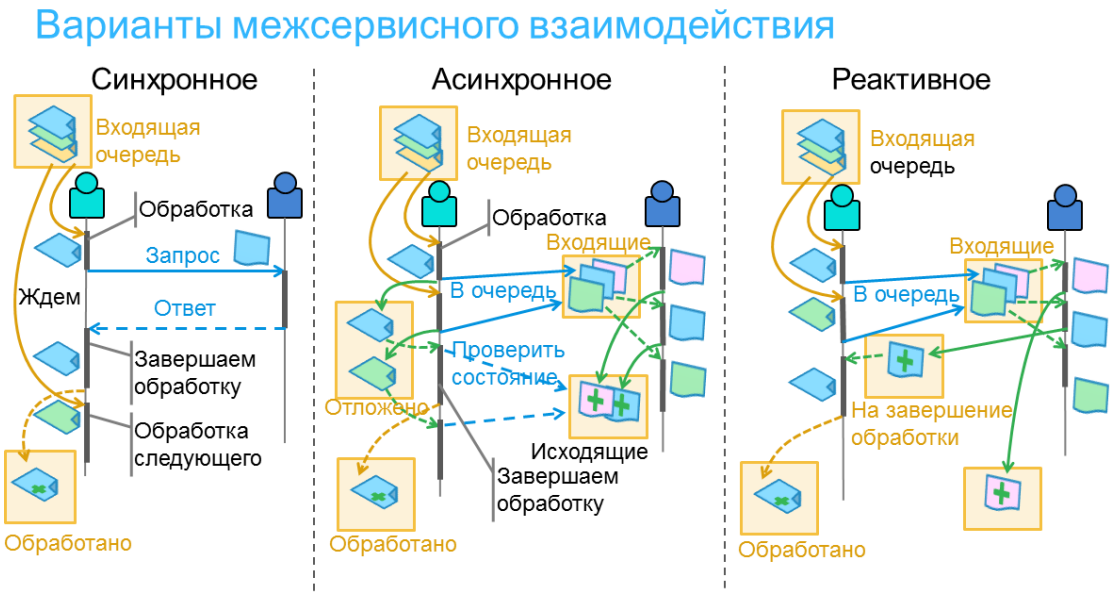
— ошибки;

— прочие данные, которые передаются между программными компонентами.

В очередях сообщений существует компонент, который называют производителем — **Producer**. Он служит для добавления сообщения в очередь, где оно станет храниться до тех пор, пока другой компонент с именем потребитель (**Consumer**) не извлечет это сообщение и не выполнит с ним нужную операцию.

Сообщения хранятся в очереди пока не будут обработаны и удалены. Каждое сообщение обрабатывается только один раз и только одним потребителем.

На



схеме вы можете увидеть все три варианта взаимодействия. Важно то, что схема описывает обработку не одного сообщения, а нескольких из входящей очереди. Потому что для анализа производительности и устойчивости важна не просто обработка одного сообщения — важно, как экземпляр сервиса обрабатывает поток сообщений из очереди, переходя из одного к другому и откладывая незавершенную обработку.

Синхронное взаимодействие — самое простое. Оно скрывает все детали удаленного вызова, что для вызывающего сервиса превращается в обычный вызов функции с получением ответа. Для его организации есть множество протоколов — например, давно известные RPC и SOAP. Но очевидная проблема синхронности в том, что удаленный сервис может отвечать не очень быстро даже при простой операции — на время ответа влияет загруженность сетевой инфраструктуры, а также другие факторы. И все это время вызывающий сервис находится в режиме ожидания, блокируя память и другие ресурсы (хотя и не потребляя процессор). В свою очередь, блокированные ресурсы могут останавливать работу других экземпляров сервиса по обработке сообщений, замедляя тем самым уже весь поток обработки. А если в момент обращения к внешнему сервису у нас есть незавершенная транзакция в базе данных, которая держит блокировки в БД, мы можем получить каскадное распространение блокировок.

Асинхронное и реактивное взаимодействие

Асинхронное взаимодействие предполагает, что вы посылаете сообщение, которое будет обработано когда-нибудь позднее. И тут возникают вопросы — а как получать ответ об обработке? И нужно ли вообще его получать? Потому что некоторые системы оставляют это пользователям, предлагая периодически обновлять таблицы документов в интерфейсе, чтобы узнать статус. Но достаточно часто статус все-таки нужен для того, чтобы продолжить обработку — например, после полного завершения резервирования заказа на складе передать его на оплату.

Для получения ответа есть два основных способа:

- обычное **асинхронное** взаимодействие, когда передающая система сама периодически опрашивает состояние документа;
- и **реактивное**, при котором принимающая система вызывает callback или отправляет сообщение о результате обработки заданному в исходном запросе адресату.

Оба способа вы можете увидеть на схеме вместе с очередями и логикой обработки, которая при этом возникает.

Какой именно способ использовать — зависит от способа связи. Когда канал однонаправленный, как при обычном клиент-серверном взаимодействии или http-протоколе, то клиент может запросить сервер, а вот сервер не может обратиться к клиенту — взаимодействие получается асинхронным.

Вот как можно представить очередь сообщений:

На практике очереди могут поддерживать получение сообщений и посредством метода **Push**, и посредством метода **Pull**. При этом:

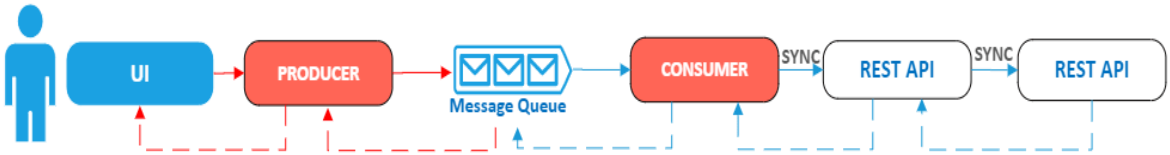
• в случае с **Pull** подразумевается периодический опрос очереди получателем на предмет наличия новых сообщений;

• в случае с **Push** подразумевается отправка уведомления получателю в случае прихода сообщения. Также здесь реализуется модель «Издатель/Подписчик» (Publisher/Subscriber).

Как известно, очереди могут работать одновременно с несколькими производителями и потребителями. По этой причине очереди, как правило, реализуют посредством дополнительной системы, которую называют **брокер**.

Message Broker — это брокер сообщений, который занимается как сбором, так и маршрутизацией сообщений, используя для этого предопределенную логику. При этом сообщения можно передавать с некоторым ключом — как раз таки по этому ключу брокер и понимает, в какую именно очередь (в одну либо в несколько) должно попасть нужное сообщение.

Вот как выглядит один из вариантов асинхронного взаимодействия, построенный на основе очереди сообщений:



Выводы

Итак, применение очередей сообщений позволит решить две задачи одновременно: — сокращение времени ожидания пользователя благодаря асинхронной обработке; — предотвращение потери информации при сбоях.

Польза и преимущества очередей сообщений в микросервисной архитектуре.

Используя очереди сообщений в качестве основного средства взаимодействия микросервисов (Microservices Communication), можно добиться следующих преимуществ:

Отделение логически независимых компонентов друг от друга (Decoupling).
Отличительная черта микросервисов — их автономность. И очереди во многом помогают уменьшить зависимости между ними. Каждое сообщение, передаваемое в очереди, — это всего лишь массив байтов с некоторыми метаданными. Метаданные нужны для направления в конкретную очередь, а информация, содержащаяся в основной части (теле) сообщения,

может быть практически любой. Брокер не анализирует данные, он выступает лишь в качестве маршрутизатора. Это позволяет настроить взаимодействие между компонентами, работающими даже на разных языках и платформах.

Улучшение масштабируемости
Очереди сообщений упрощают независимое масштабирование микросервисов. Наблюдая за состоянием очередей, можно масштабировать те сервисы, на которые приходится большая часть нагрузки. Кроме этого, очереди легко позволяют не только увеличивать число экземпляров существующих сервисов, но и добавлять новые с минимальным временем простоя. Все, что для этого требуется, — добавить нового потребителя, прослушивающего события в очереди. Однако сами очереди также необходимо масштабировать, и это может создать дополнительные сложности.

Балансировка нагрузки
Если один из сервисов не справляется с нагрузкой, требуется возможность запускать больше его экземпляров быстро и без дополнительных настроек. Обычно для этих целей используют балансировщик нагрузки, интегрированный с сервером обнаружения служб и предназначенный для распределения трафика. При использовании очередей сообщений сам брокер по умолчанию является балансировщиком нагрузки. Если несколько потребителей слушают очередь одновременно, сообщения будут распределяться между ними в соответствии с настроенной стратегией.

Повышение надежности
Выход из строя одного из компонентов не сказывается на работе всей системы: при восстановлении он обработает сообщение, находящееся в очереди. Ваш веб-сайт по-прежнему может работать, даже если задерживается часть обработки заказа, например, из-за проблем с сервером БД или системой электронной почты. Правда, при этом очередь сама приобретает статус SPoF (Single Point Of Failure), поэтому необходимо заранее предусмотреть действия на случай ее аварийного отключения.

Безопасность
Большинство брокеров выполняют аутентификацию приложений, которые пытаются получить доступ к очереди, и позволяют использовать шифрование сообщений как при их передаче по сети, так и при хранении в самой очереди. Таким образом, очередь снимает с ваших сервисов бремя организации авторизации запросов.

Варианты использования очередей сообщений
Очереди сообщений полезны в тех случаях, где возможна асинхронная обработка. Рассмотрим наиболее частые сценарии использования очередей сообщений (Message Queue use Cases):

Фоновая обработка долгосрочных задач на веб-сайтах
Сюда можно отнести задачи, которые не связаны напрямую с основным действием пользователя сайта и могут быть выполнены в фоновом режиме без необходимости ожидания с его стороны. Это обработка изображений, преобразование видео в различные форматы, создание отзывов, индексирование в поисковых системах после изменения данных, отправка электронной почты, формирование файлов и так далее.

Буферизация при пакетной обработке данных
Очереди можно использовать в качестве буфера для некоторой массовой обработки, например пакетной вставки данных в БД или HDFS. Очевидно, что гораздо эффективнее добавлять сто записей за раз, чем по одной сто раз, так как сокращаются накладные расходы на

инициализацию и завершение каждой операции. Но для стандартной архитектуры может стать проблемой генерация данных клиентской службой быстрее, чем их может обработать получатель. Очередь же предоставляет временное хранилище для пакетов с данными, где они будут храниться до завершения обработки принимающей стороной.

Отложенные задачи
Многие системы очередей позволяют производителю указать, что доставка сообщений должна быть отложена. Это может быть полезно при реализации льготных периодов. Например, вы разрешаете покупателю отказаться от размещения заказа в течение определенного времени и ставите отложенное задание в очередь. Если покупатель отменит операцию в указанный срок, сообщение можно удалить из очереди.

Сглаживание пиковых нагрузок
Помещая данные в очередь, вы можете быть уверены, что данные будут сохранены и в конечном итоге обработаны, даже если это займет немного больше времени, чем обычно, из-за большого скачка трафика. Увеличить скорость обработки в таких случаях также возможно — за счет масштабирования нужных обработчиков.

Гарантированная доставка при нестабильной инфраструктуре
Нестабильная сеть в сочетании с очередью сообщений создает надежный системный ландшафт: каждое сообщение будет отправлено, как только это будет технически возможно.

Упорядочение транзакций
Многие брокеры поддерживают очереди FIFO, полезные в системах, где важно сохранить порядок транзакций. Если 1000 человек размещают заказ на вашем веб-сайте одновременно, это может создать некоторые проблемы с параллелизмом и не будет гарантировать, что первый заказ будет выполнен первым. С помощью очереди можно определить порядок их обработки.

Сбор аналитической информации
Очереди часто применяют для сбора некоторой статистики, например использования определенной системы и ее функций. Как правило, моментальная обработка такой информации не требуется. Когда сообщения поступают в веб-службу, они помещаются в очередь, а затем при помощи дополнительных серверов приложений обрабатываются и отправляются в базу данных.

Разбиение трудоемких задач на множество маленьких частей
Если у вас есть некоторая задача для группы серверов, то вам необходимо выполнить ее на каждом сервере. Например, при редактировании шаблона мониторинга потребуется обновить мониторы на каждом сервере, использующем этот шаблон. Вы можете поставить сообщение в очередь для каждого сервера и выполнять их одновременно в виде небольших операций.

Прочие сценарии, требующие гарантированной доставки информации и высокого уровня отказоустойчивости
Это обработка финансовых транзакций, бронирование авиабилетов, обновление записей о пациентах в сфере здравоохранения и так далее.

Сложности использования и недостатки очередей сообщений: как с ними справляться

Несмотря на многочисленные преимущества очередей сообщений, самостоятельное их внедрение может оказаться довольно сложной задачей по нескольким причинам:

- По сути, это еще одна система, которую необходимо купить/установить, правильно сконфигурировать и поддерживать. Также потребуются дополнительные мощности.
- Если брокер когда-либо выйдет из строя, это может остановить работу многих систем, взаимодействующих с ним. Как минимум необходимо позаботиться о резервном копировании данных.
- С ростом числа очередей усложняется и отладка. При синхронной обработке сразу очевидно, какой запрос вызвал сбой, например, благодаря иерархии вызовов в IDE. В очередях потребуется позаботиться о системе трассировки, чтобы быстро связать несколько этапов обработки одного запроса для обнаружения причины ошибки.
- При использовании очередей вы неизбежно столкнетесь с выбором стратегии доставки сообщений. В идеале сообщения должны обрабатываться каждым потребителем однократно. Но на практике это сложно реализовать из-за несовершенства сетей и прочей инфраструктуры. Большинство брокеров поддерживают две стратегии: доставка хотя бы раз (At-least-once) или максимум раз (At-most-once). Первая может привести к дубликатам, вторая — к потере сообщений. Обе требуют тщательного мониторинга. Некоторые брокеры также гарантируют строго однократную доставку (Exactly-once) с использованием порядковых номеров пакетов данных, но даже в этом случае требуется дополнительная проверка на стороне получателя.

В каких случаях очереди неэффективны

Конечно, очереди не являются универсальным средством для любых приложений. Рассмотрим варианты, когда очереди не будут самым эффективным решением:

- У вашего приложения простая архитектура и функции, и вы не ожидаете его роста. Важно понимать, что очереди сообщений — это дополнительная сложность. Эту систему также необходимо настраивать, поддерживать, осуществлять мониторинг ее работы и так далее. Да, можно использовать Managed-решение, но вряд ли это будет оправдано для небольших приложений. Добавление очередей должно упрощать архитектуру, а не усложнять ее.
- Вы используете монолитное программное обеспечение, в котором развязка (Decoupling) невозможна или не приоритетна. Если вы не планируете разбивать монолит на микросервисы, но вам требуется асинхронность — для ее реализации обычно достаточно стандартной многопоточной модели. Очереди могут оказаться избыточным решением до тех пор, пока не возникнет явная необходимость в разделении приложения на автономные компоненты, способные независимо выполнять задачи.