ВЗАИМОДЕЙСТВИЕ В АРХИТЕКТУРЕ МИКРОСЛУЖБ С ПОМОЩЬЮ АСИНХРОННОГО ПРОТОКОЛА

Панина В.В.[1], Мирошниченко Е.А.[2]
[1]Национальный исследовательски Томский политехнический университет, Инженерная школа информационных технологий и робототехники, 8ИМ21, e-mail: vvp39@tpu.ru
[2]Национальный исследовательски Томский политехнический университет, Инженерная школа информационных технологий и робототехники, доцент

Введение

С каждым годом растёт популярность использования микросервисной архитектуры в противовес монолиту. Основными преимуществами такой архитектуры являются высокая отказоустойчивость и автономность, за счёт этого уменьшаются строгие зависимости между элементами системы. Еще одним преимуществом является масштабируемость – то есть способность системы увеличивать производительность за счет увеличения количества выделенных ей ресурсов. Микросервисная архитектура позволяет масштабировать каждый сервис отдельно, не затрагивая остальную систему.

Вместе с тем появляются новые сложности, а именно – взаимодействие между сервисами, ведь каждый сервис физически изолирован от других. Для решения этой проблемы существуют подходы с применением протоколов взаимодействия.

Цель настоящей работы — рассмотреть в архитектуре микросервисов взаимодействие с помощью синхронного и асинхронного протоколов, провести их сравнение, а также подробнее описать асинхронный протокол на примере брокера сообщений RabbitMQ.

Протоколы взаимодействия в микросервисной архитектуре

Приложение на основе микрослужб представляет собой распределенную систему, работающую на нескольких процессах или службах, иногда даже не нескольких серверах или узлах. Обычно каждый экземпляр службы — это процесс. Таким образом, службы должны взаимодействовать по протоколу внутрипроцессного взаимодействия, например HTTP (Hyper Text Transfer Protocol), AMQP (Advanced Message Queuing Protocol) или двоичному протоколу, такому как TCP, в зависимости от характера каждой службы [1].

Обычно используются либо синхронные HTTP-запросы и -ответы, либо асинхронные протоколы AMQP– протокол обмена сообщениями через AMQP-брокер. Существует множество брокеров: Kafka, RabbitMQ, ActiveMQ, ZeroMQ и другие [2].

1. Синхронный протокол.

HTTP – это синхронный протокол. Образуется цепочка запросов, важно, что код клиента сможет продолжить выполнение задачи только после получения ответа от HTTP-сервера (рисунок 1). Такая схема взаимодействия может считаться анти-паттерном при разработке, так как нарушается принцип автономности микросервисной архитектуры. Такая архитектура будет неустойчива к сбоям – при недоступности одного сервиса будет недоступно все приложение.

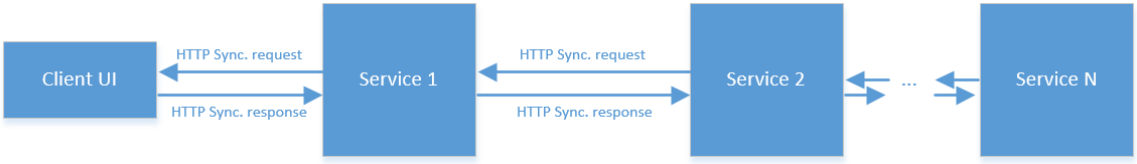

Рис. 1. Схема взаимодействия микросервисов с помощью синхронного протокола

2. Асинхронный протокол.

Протоколы AMQP используют асинхронные сообщения. Код клиента или отправитель сообщения не ожидает ответа, а отправляет сообщение, как при отправке сообщения в очередь брокера сообщений (например, RabbitMQ). Схема взаимодействия представлена на рисунке 2.

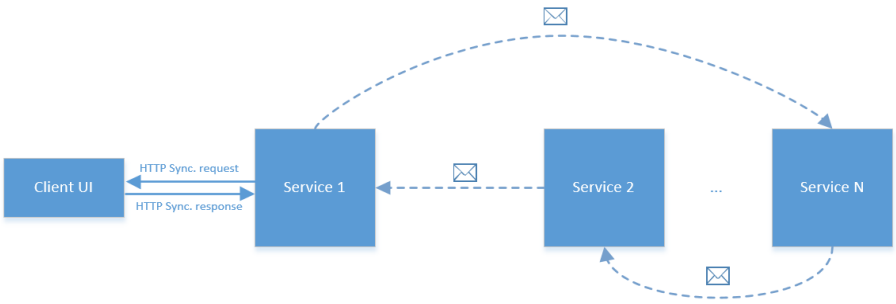

Рис. 2. Схема взаимодействия микросервисов с помощью асинхронного протокола

Здесь HTTP-ответ от сервиса приходит моментально. Например, при совершении оплаты товара на каком-либо сайте (Client UI), происходит взаимодействие с сервисом (Service 1) по обработке платежей. Этот сервис обращается к другому (Service N) с запросом на получение push-уведомления на мобильное устройство заказчика. Service 1, не дожидаясь ответа от Service N, перенаправляет Client UI на новую страницу с формой для заполнения пароля из push-уведомления. Таким образом реализуется асинхронный протокол. Далее в докладе подробнее описано взаимодействие между сервисами с помощью брокера RabbitMQ (рис. 3).

Характеристики протоколов вынесены в сводную таблицу, где они описаны в соответствии с критериями микросервисной архитектуры.

Таблица 1

Анализ синхронного и асинхронного протоколов

Критерий	AMQP	HTTP	Описание
Асинхронность	+	-	
Передача данных между микросервисами	+	+	Основной критерий для протоколов передачи данных. Оба протокола справляются.
Прозрачность	+	+	Возможность отследить запросы и ответы
Документация	+	+	У обоих протоколов есть сопровождающая документация
Масштабируемость	+	-	
Гарантия доставки сообщений	+	-	HTTP протокол «теряет» сообщение в случае отказа одного из сервисов, AMQP хранит сообщение в очереди.
Управление проблемами сервера	+	-	Протокол HTTP не способен реагировать на проблему сбоя сервера, AMQP во время сбоя серверов хранит непереданные сообщения в очереди, может сам по себе привести к сбою сервера

У AMQP надежная доставка сообщений. Это асинхронный протокол, так что о доставке можно вообще не волноваться. HTTP, в свою очередь, — самый поддерживаемый протокол в интернете. Разработчики знают HTTP: не нужно дополнительно обучать каждого новичка на проекте [3]. В общем случае без каких-либо конкретных требований оба протокола справляются со своей задачей.

Брокер RabbitMQ

RabbitMQ первоначально был выпущен в 2007 компаниями L Shift и CohesiveFT и был одним из самых первых брокеров обмена сообщениями для реализации спецификации AMPQ. Помимо RabbitMQ, существуют реализации протокола AMQP, разработанные компанией Apache Software Foundation — Apache ActiveMQ и Apache Qpid [4].

AMQP имеет несколько базовых понятий [5]:

– Message – передаваемая информация;

– Exchange – обменник/биржа, в неё отправляются сообщения, откуда они распределяются по очередям. Сообщение не хранится на обменнике/бирже;

– Queue – очередь, хранилище сообщений. Сообщений хранятся в очереди до тех пор, пока клиенты не заберут их; Очередь хранит ссылку на полученное сообщение, которое физически может хранится в оперативной памяти или на диске, в зависимости от свойств самой очереди [6].

– Routing key – маршрут сообщений. Определяет в какую очередь будет добавлено сообщение;

– Producer /publisher– отправитель сообщения;
– Consumer – получатель сообщения.

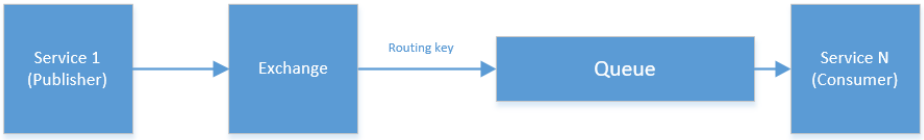

Рис.3. Схема взаимодействия AMQP

Когда Service N готов получить сообщение из очереди, брокер копирует его по ссылке из очереди и отправляет копию сообщения потребителю, далее получив сообщение, Service N отправляет брокеру подтверждение об успешном получении данных, и брокер удаляет копию сообщения из очереди и исходное сообщение из очереди, очищая место в оперативной памяти или на жестком диске.

Таким образом, RabbitMQ может использоваться для фоновой обработки долгосрочных задач на веб-сайтах, гарантированной доставки сообщений до получателя, упорядоченности транзакций с помощью очереди и высокого уровня отказоустойчивости.

В статье «Как оживает Rabbit» [7] описывается показательный примера, когда нужно использовать RabbitMQ.

RabbitMQ полезен для приложения какой-либо социальной сети. При новом добавлении поста от какого-либо пользователя все подписчики должны быть оповещены об этом. Это может быть сильно потребляющей время операцией. Решением было бы помещать задачу оповещения в надлежащую очередь для всякого поста по мере его появления. Когда соответствующий обработчик получит этот запрос, он извлечёт перечень последователей для данного отправителя и обновит их все.

Заключение

Протокол взаимодействия AMQP и брокер RabbitMQ служит для информационных систем обменником данных с гарантированной доставкой сообщений, возможностью к масштабированию системы. AMQP можно использовать в любой ситуации, когда существует потребность в высококачественной и безопасной доставке сообщений между сервисами в архитектуре микрослужб.

Список использованных источников

1. Взаимодействие в архитектуре микрослужб. [Электронный ресурс]. – URL: https://learn.microsoft.com/ru-ru/dotnet/architecture/microservices/architect-microservice-container-applications/communication-in-microservice-architecture (дата обращения 24.02.2023).
2. Радишевский В. Л. Распределенный брокер сообщений Kafka для высокоскоростной передачи и агрегации данных / В. Л. Радишевский, А. Д. Кульневич. – XV Международная научно-практическая конференция студентов аспирантов и молодых учёных «Молодёжь и современные информационные технологии», 2017. – 284с.
3. AMQP vs HTTP: кто кого. [Электронный ресурс]. – URL: https://uncaughtexception.ru/posts/2017/08/28/amqp-vs-http-kto-kogo/ (дата обращения 24.02.2023).
4. Р.Н. Акрамовский. Автоматизация документооборота на предприятии с помощью брокера сообщений RabbitMQ. [Электронный ресурс]. – URL: https://www.elibrary.ru/download/elibrary_41175778_58229653.pdf (дата обращения 25.02.2023).
5. RabbitMQ: Простая и эффективная очередь сообщений. [Электронный ресурс]. – URL: https://makeomatic.ru/blog/2013/10/16/RabbitMQ/ (дата обращения 25.02.2023).
6. RabbitMQ для аналитика: практический ликбез. [Электронный ресурс]. – URL: https://babok-school.ru/blogs/rabbitmq-for-analyst/ (дата обращения 25.02.2023).
7. Как оживает Rabbit. [Электронный ресурс]. – URL: http://onreader.mdl.ru/RabbitMQEssentialsDistributedScalableApplications.2nd/content/Ch01.html (дата обращения 25.02.2023).