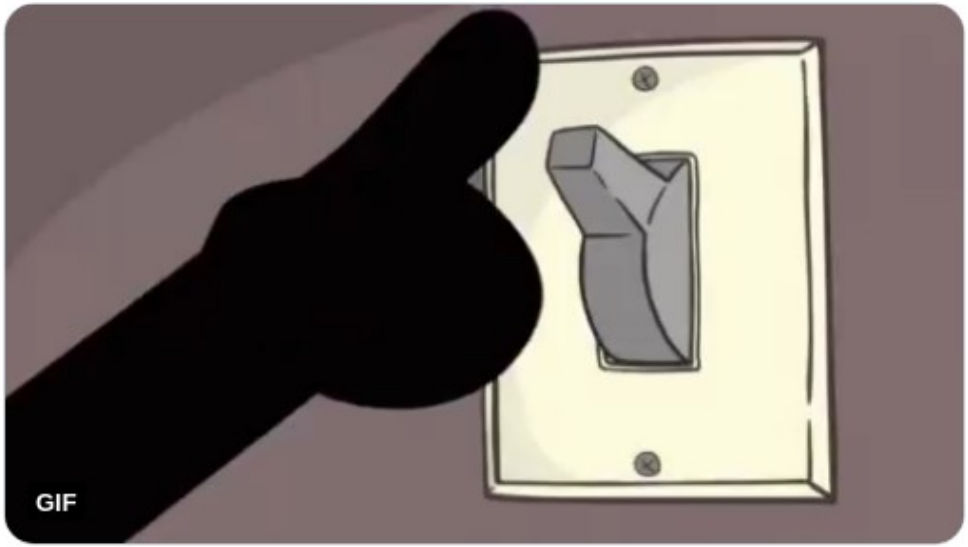
Microservices and sagas

Pinned Tweet

David Russell @gitpitch · Feb 22

This is a friendly and final reminder that GitPitch will shut down permanently on March 1st, 2021. On that date I will shutdown the online service plus end support for all GitPitch software: desktop, enterprise, and CICD. Thank you all for sharing in this adventure 💙 David.



2 5 12

План

- Часть 1. Определения, контекст и общие положения
- Часть 2. Запись данных
- Часть 3. Чтение данных

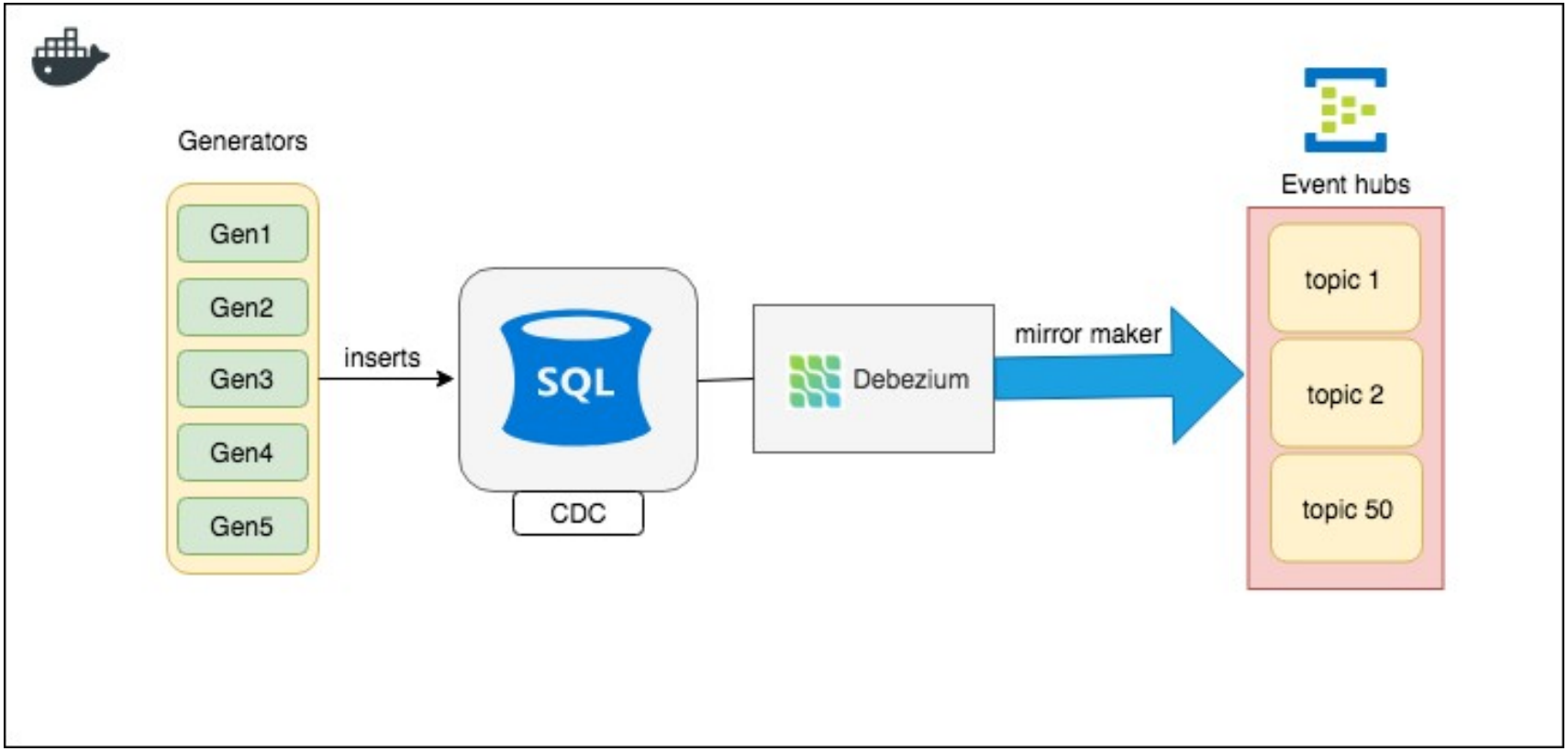
Часть 1. Определение микросервиса

- Микросервис - обособленный модуль со своим хранилищем(sql, nosql), над разработкой которого работает 1 команда(8-10 человек).

Часть 1. Общий принцип общения между сервисами

- 1. приложение сохраняет бизнес объекты и ивент который нужно опубликовать в локальное sql хранилище через eventuate
- 2. CDC(change data capture) подхватывает из например postgres write-ahead-log'a ивенты и пушит их в кафку
- 3. консумер уже из другого сервиса обрабатывает полученный ивент

Часть 1. Общий принцип общения между сервисами



Часть 2. Запись данных. Saga

- Saga - модель взаимодействия нескольких сервисов для того чтобы одна операция которая span'иться на несколько сервисов выполнялась транзакционно
- Она же - совокупность действий(локальных транзакций) которые являются одной логической "транзакцией"

Часть 2. Запись данных. Saga. Примеры локальных транзакций

Примеры действий(локальные транзакции):

- "понизить баланс клиенту"
- "послать запрос в нейтивпей и отрепортить назад ответ"
- "повысить баланс мерчанту"

Часть 2. Запись данных. Saga. Типы локальных транзакций

- Compensable - транзакция для которой существует откат(казахская транзакция). Например "заблокировать деньги на счету юзера", так как всегда можно разблокировать деньги назад
- Pivot - транзакция, для которой отката не сущесвтует и она всегда либо выполнится, либо зафейлится. Например отправить запрос в нейтивпей - это будет означать что либо транзакция прошла и клиент получит деньги на свой киви кошелек, либо что транзакция не прошла и тут ничего не сделаешь.
- Retryable - транзакция которая в конечно счете выполнится. Например "разблокировать деньги клиента", если сага зафейлилась, либо "списать заблокированные деньги клиента + добавить денег мерчанту"

Часть 2. Запись данных. Saga. Типы локальных транзакций

В общем виде saga всегда выглядит вот так(все опциональны):

- -Compensable transaction1
- ...
- -Compensable transactionN
- -pivot transaction
- -Retryable transaction1
- ...
- -Retryable transactionM

Часть 2. Запись данных. Saga. Типы локальных транзакций - пример

p2p перевод:

- блок денег юзера1
- пополнение баланса юзера2
- списание юзера1

Часть 2. Запись данных. Saga. Типы локальных транзакций - пример

p2p перевод:

- блок денег юзера1
- пополнение баланса юзера2
- списание юзера1

- 1. pivot -> retryable -> retryable
- 2. compensable -> pivot -> retryable
- 3. compensable -> retryable -> retryable

Часть 2. Запись данных. Saga. Типы локальных транзакций - пример2

Оплата по карте услуги:

- Блок денег юзера1(может зафейлиться)
- Блок денег на карте(может зафейлиться, но если успех, то можно разблокировать или списать деньги специальным вызовом)
- Оплата комм. услуги(возвращает либо “успех”, либо “неуспех” и это финальный статус)
- Пополнение баланса юзера2(не может зафейлить)
- Списание денег с карты(не может зафейлить)
- Списание эл. денег юзера1(не может зафейлить)

Часть 2. Запись данных. Saga. Типы локальных транзакций - пример

Оплата по карте услуги:

- Блок денег юзера1(может зафейлиться)
- Блок денег на карте(может зафейлиться, но если успех, то можно разблокировать или списать деньги специальным вызовом)
- Оплата комм. услуги(возвращает либо “успех”, либо “неуспех” и это финальный статус)
- Пополнение баланса юзера2(не может зафейлить)
- Списание денег с карты(не может зафейлить)
- Списание эл. денег юзера1(не может зафейлить)

- Ответ: compensable -> compensable -> pivot -> retryable -> retryable

Часть 2. Запись данных. Saga. Виды и свойства saga

ACD - без Isolation

Типы:

- хореографическая сага
- оркестраторная сага

Часть 2. Запись данных. Saga. Хореографическая.

- Нет общего координатора, распределенные апдейты, хорошо подходит event-stream подходу
- - Сервис1: блокирует деньги и публикует ивент "деньги пользователя XXX по транзакции YYY заблокированы"
- - Сервис2: получает предыдущее сообщение и запрашивает нейтивпей на оплату. Получив ответ публикует ивент "оплата нейтивпею по транзакции YYY прошла успешно"
- - Сервис1: получает предыдущее сообщение и добавляет деньги мерчанту и снимает с пользователя и публикует сообщение "транзакция YYY завершена успешно"

Часть 2. Запись данных. Saga. Хореографическая.

Плюсы:

- Проще при небольших сагах

Минусы:

- При сложных превращается в лапшу
- Сложно сказать состояние транзакции, так как оно размазано по сервисам

Часть 2. Запись данных. Saga. Оркестраторная

- Единый координатор.
- оркестратор: шлет сообщение сервису1: "заблокируй деньги"
- сервис1: "заблокировано!"
- оркестратор: шлет сообщение сервису2: "оплати нейтивпей"
- сервис2: "оплачено!"
- оркестратор: шлет сообщение сервису1: "списывай деньги у клиента и добавляй мерчанту"
- сервис1: "сделано!"
- оркестратор: завершает у себя транзакции и может еще что шлет.

Часть 2. Запись данных. Saga. Оркестраторная

- Плюсы:
- Проще при больших сагах
- Состояние транзакции хранится в координаторе

Минусы:

- Новая сущность - координатор
- Больше трафика и места на диске

Часть 2. Запись данных. Как много информации в ивентах?

- Как много информации в ивентах?
- 1. "Транзакция YYY - деньги заблокированы" или
- 2. "транзакция YYY по оплате нейтивпей на номер счета ZZZ - деньги заблокированы"
- Плюсы первого подхода:
- меньше трафика и места на диске(так как все сообщения персистятся и реплицируются кафкой)
- меньше согласований по формату(которые он называется desgin time coupling) в очереди(правда оно остается в запросах к сервису)
- Плюсы второго:
- нет лишних обращений. Никуда лезть не нужно, все данные тут, на руках.

Часть 3. Чтение данных. Подходы

- api gateway
- CQRS паттерн

Часть 3. Чтение данных. API Gateway

- API Gateway - синхронный вызов нужных сервисов
- Минусы:
- runtime coupling
- иногда невозможно эффективно реализовать

Плюсы:

- Легче в реализации и поддержке
- Не тратит лишнего места на диске

Часть 3. Чтение данных. CQRS

Command Query Responsibility Segregation - хранилище под конкретные запросы.

Плюсы:

- возможность делать джоины
- эффективность

Минусы:

- требуется реализация
- требуется место на диске

Итог

- Микросервис – сервис который поддерживает одна команда
- Для записи используется сага паттерн
- Для обмена сообщениями между сервисами используется CDC, кафка и фреймворк eventuate
- Сага бывает хореографическая и оркестраторная
- Локальные транзакции в сага: compensable, pivot, retryable
- Чтение реализовано через api gateway или через CQRS
- 2 вида coupling’a: runtime и design time