

第11期-分布式微服务的挑战和解决方案

2020年1月2日 14:58

挑战 #1：如何定义每个微服务的边界

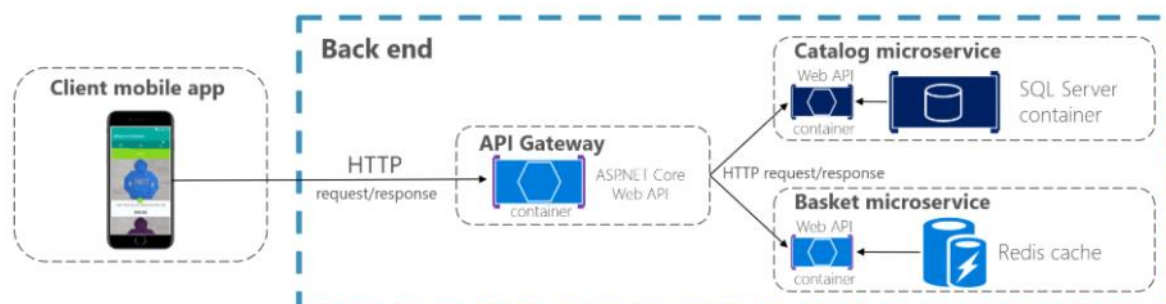
首先，需要着重关注应用程序的逻辑域模型和相关数据是否独立运行，形成分离岛，每个上下文可能具有不同的业务语言（不同的业务术语），例如：用户在身份认证上下文中称为用户，在 CRM 上下文中称为客户，在订单上下文中称为购买者等。

挑战 #2：如何创建从多个微服务中检索数据的查询

一个屏幕可能要展示来自多个微服务数据库的数据结合，以便提高通信效率。解决方案：

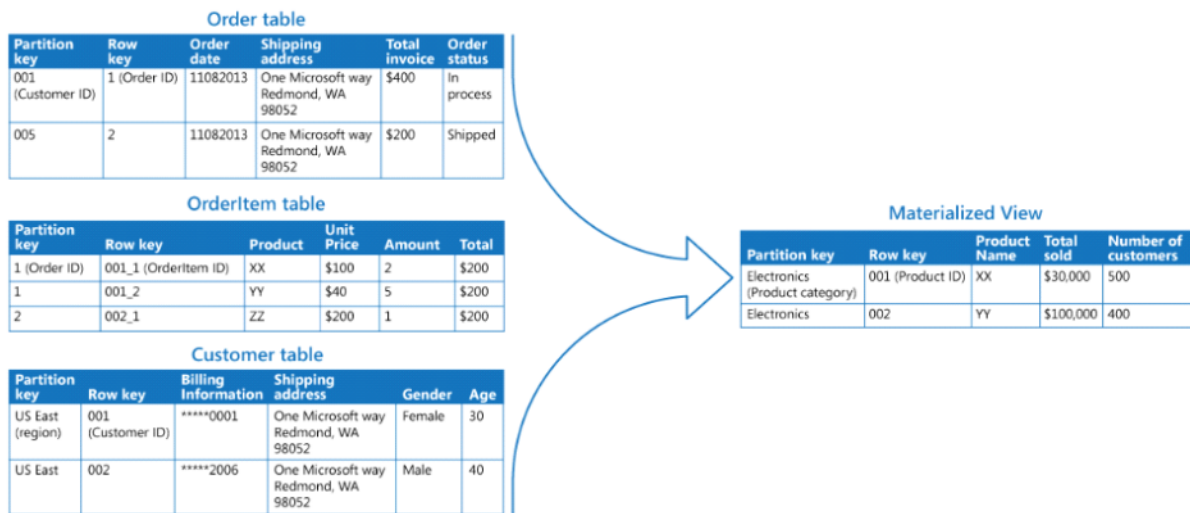
API 网关

聚合微服务，但可能成为系统中的瓶颈点，并可能违反微服务自治原则，若要降低这种可能性，尽量采用多个细粒度 API 网关。



CQRS命令查询职责分离

用于聚合来自多个微服务的数据的另一个解决方案是具体化视图模式。



如果有多个数据库，并且每个数据库都属于不同微服务，则无法查询这些数据库和创建 SQL 联接。此时，复杂查询则变成了一种挑战。可以使用 CQRS 方法来满足需求，即在仅用于查询的不同数据库中创建非规范化表。

要完成额外的开发工作，并且需要达成最终一致性。

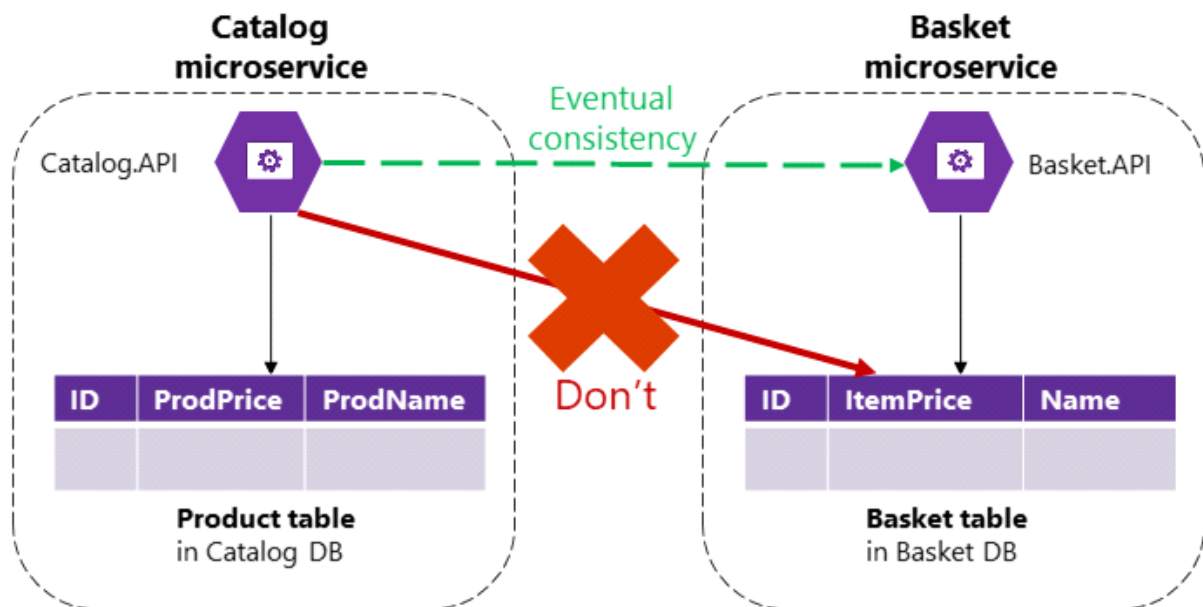
挑战 #3：如何实现微服务之间的一致性

每个微服务拥有的数据是该微服务专有的，并且只能通过其本身的微服务 API 访问。因此，面临的挑战是如何在保持多个微服务数据一致性。

比如：产品加入购物车，产品价格变动，购物车价格也要更新。

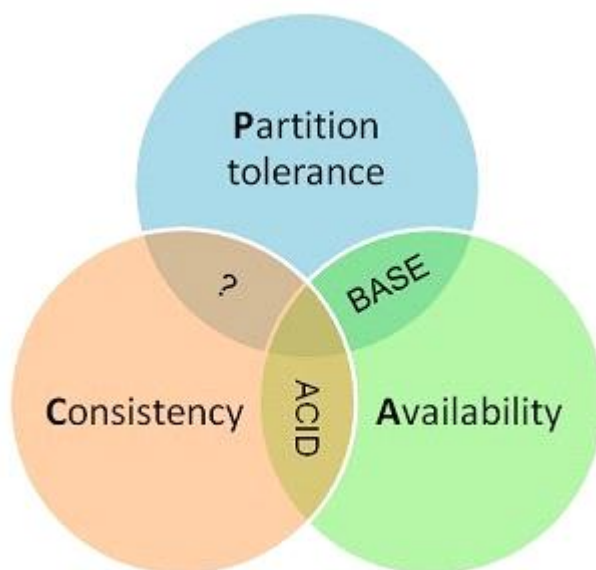
单体应用中，只需使用 ACID 事务将产品更改价格和更新购物篮当前价格拉入一个事务。

微服务中，产品信息和购物篮属于其各自的微服务，都拥有自己的数据库，两个微服务独立自治，无法关联事务。产品微服务不应直接更新购物篮表，因为购物篮表属于购物篮微服务。



Databases are private per microservice

方案：使用微服务间API调用同步通信，使用事件队列异步通信保持最终一致性。在通过事件驱动通信和发布订阅系统形成的微服务之间使用最终一致性。



CAP原则和BASE原则

谈谈分布式系统的CAP理论

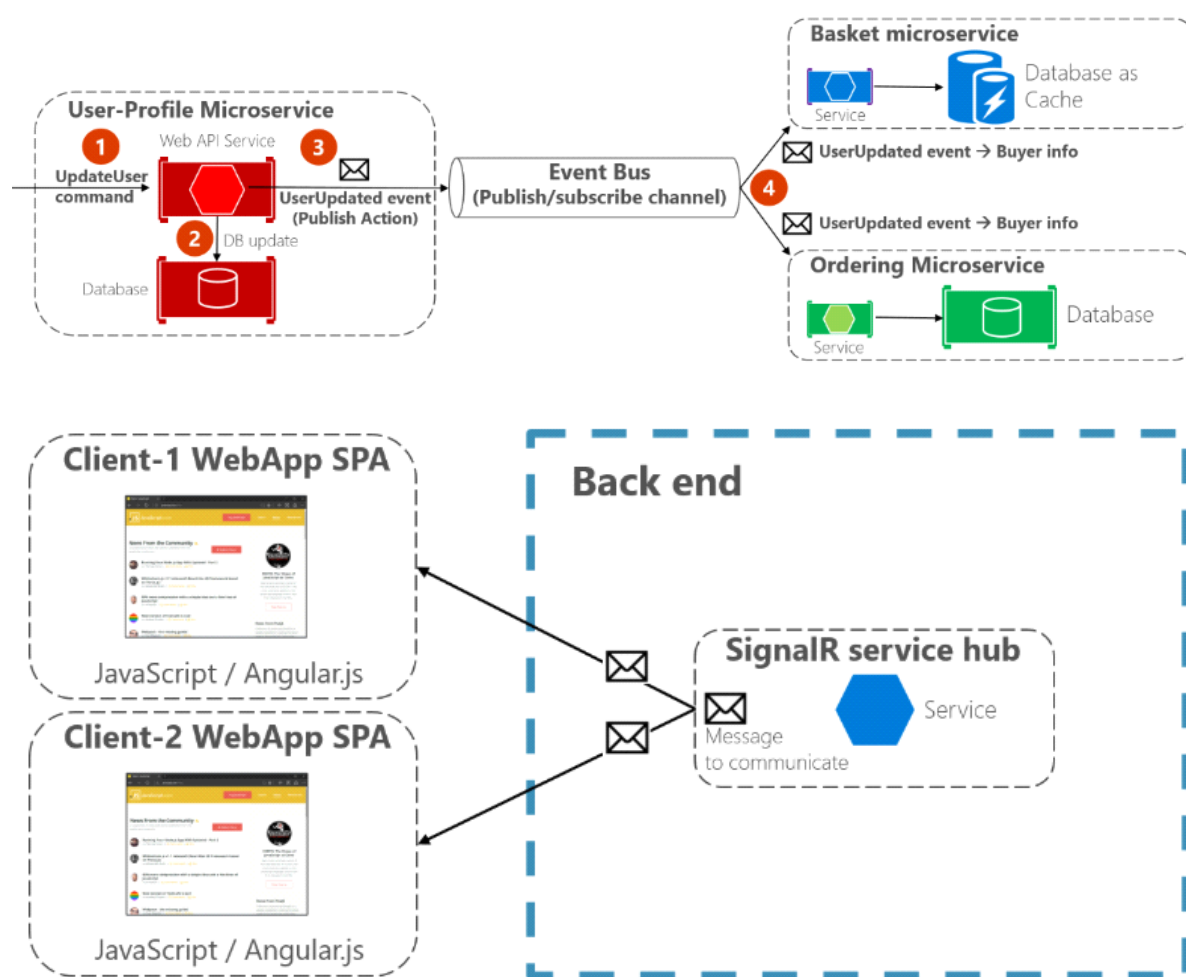
CAP原则(CAP定理)、BASE理论

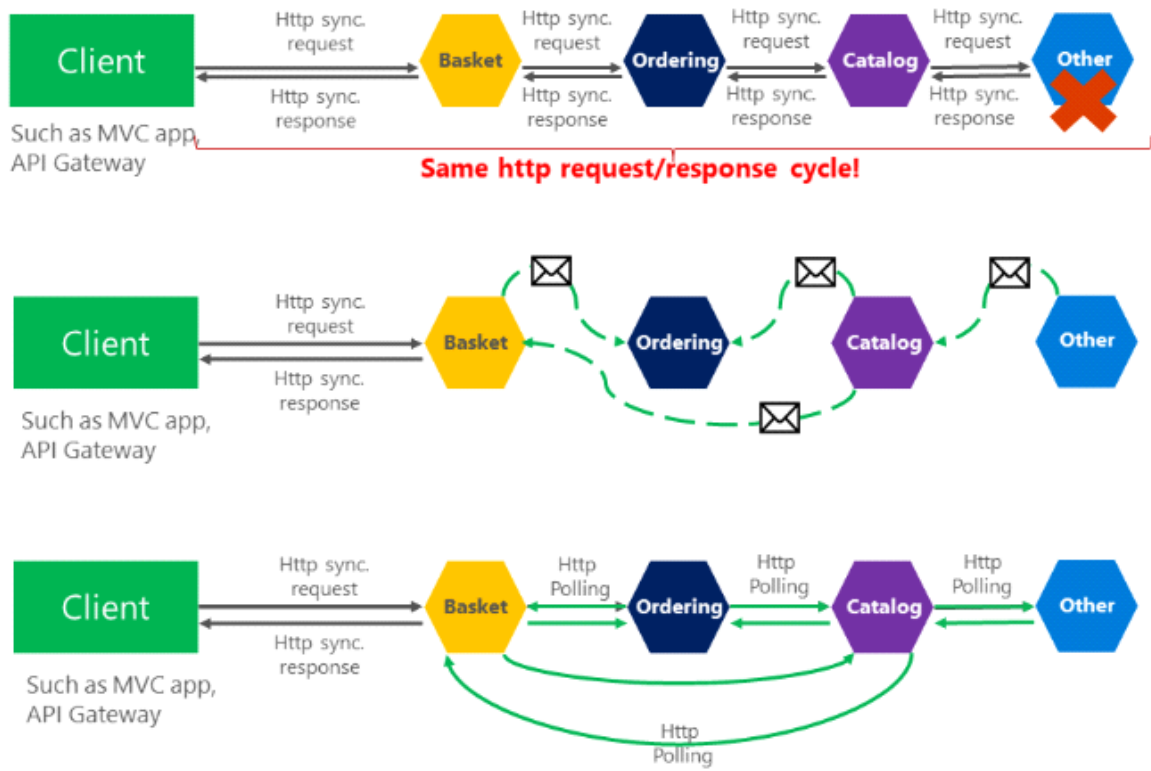
挑战 #4：如何设计跨微服务边界的通信

常用方法是实现基于 HTTP (REST) 的微服务，因为该服务具有简便性。基于 HTTP 的方法是完全可以接受的；这里存在的问题与其使用方法有关。如果使用 HTTP 请求和响应只是为了与来自客户端应用程序或 API 网关的微服务进行交互，这是可行的。但如果创建跨微服务的长链同步 HTTP 调用，就好像微服务是单片应用程序中的对象一样进行跨边界通信，则应用程序最终会遇到问题。

阻止和低降低。由于 HTTP 的同步性质，直到所有内部 HTTP 调用完成后，原始请求才会收到响应。HTTP 调用链越复杂，实现基于 HTTP 的故障策略也会越复杂。

应尽可能少地使用跨微服务的请求/响应通信链。建议只使用异步交互进行微服务之间的通信，方法是使用基于消息和基于事件的异步通信，或（异步）使用独立于原始 HTTP 请求/响应周期的 HTTP 轮询。





微软微服务体系架构指南

分布式数据管理的挑战和解决方案

API 网关模式与客户端到微服务直接通信

微服务体系结构中的通信

基于消息的异步通信

在微服务（集成事件）之间实现基于事件的通信

使用 RabbitMQ 实现用于开发或测试环境的事件总线

订阅事件