- RPC
  - 微服务是什么
  - 什么是RPC
    - 什么是本地调用？什么是远程调用
    - RPC是不是只能用在所谓的微服务里呢?
  - RPC的核心价值
  - 为什么我们需要RPC
  - 如何学习RPC

RPC

在讲RPC之前，我们要先了解一下什么是**微服务**。

微服务是什么

microservice

维基上对其定义为：
一种软件开发技术
面向服务的体系结构（SOA）架构样式的一种变体，它提倡将单一应用程序划分成一组小的服务，服务之间互相协调、互相配合，为用户提供最终价值。

每个服务运行在其独立的进程中，服务于服务间采用轻量级的通信机制互相沟通（通常是基于HTTP的RESTful API）。

每个服务都围绕着具体业务进行构建，并且能够独立地部署到生产环境、类生产环境等。
另外，应尽量避免统一的、集中式的服务管理机制，对具体的一个服务而言，应根据上下文，选择合适的语言、工具对其进行构建。

微服务（或微服务架构）是一种云原生架构方法，其中单个应用程序由许多松散耦合且可独立部署的较小组件或服务组成。这些服务通常

1. 有自己的堆栈，包括数据库和数据模型；
2. 通过REST API，事件流和消息代理的组合相互通信；
3. 它们是按业务能力组织的，分隔服务的线通常称为有界上下文。

尽管有关微服务的许多讨论都围绕体系结构定义和特征展开，但它们的价值可以通过相当简单的业务和组织收益更普遍地理解：

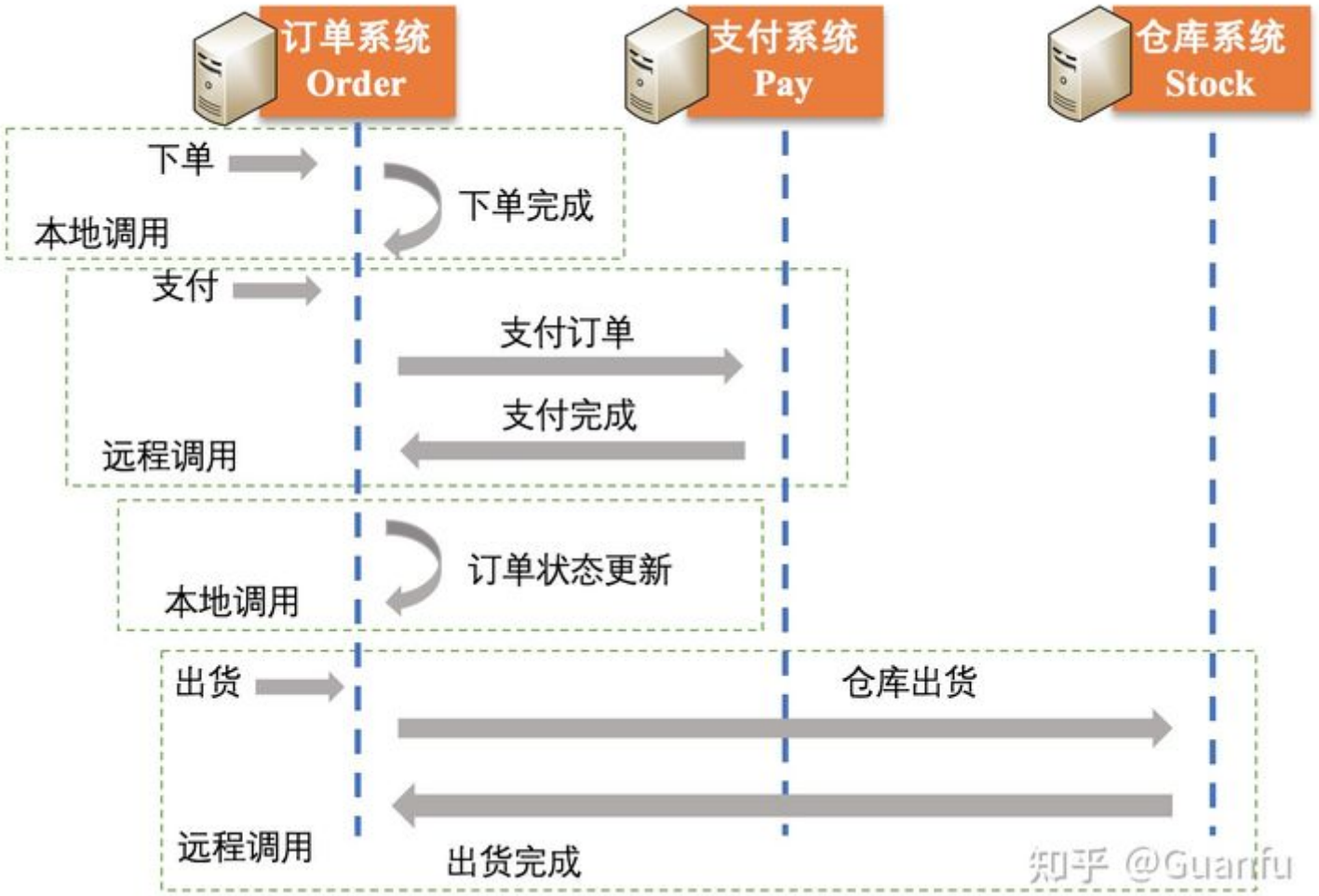
可以更轻松地更新代码。
团队可以为不同的组件使用不同的堆栈。
组件可以彼此独立地进行缩放，从而减少了因必须缩放整个应用程序而产生的浪费和成本，因为单个功能可能面临过多的负载。
微服务也可以通过它们不是什么来理解。微服务架构最经常得出的两个比较是整体架构和面向服务的架构（SOA）。
微服务和整体架构之间的区别在于，微服务由许多较小的，松散耦合的服务组成一个应用程序，与大型，紧密耦合的应用程序的整体方法相反。
微服务和SOA之间的差异可能不太清楚。虽然可以在微服务和SOA之间形成技术对比，尤其是围绕企业服务总线（ESB）的作用，但将差异视为范围之一更容易。SOA是企业范围内的一项工作，旨在标准化所有服务之间相互交流和集成的方式，而微服务体系结构则是特定于应用程序的。

什么是RPC

RPC最大的特点就是可以让我们像调用本地一样发起远程调用。
说到这里，菜鸡如我有如下疑问：

什么是本地调用？什么是远程调用

逼乎相关链接



本地调用通常指的是，进程内函数之间的相互调用；

远程调用，是进程间函数的相互调用，是一种进程间通信模式。通过远程调用，一个进程可以看到其他进程的函数、方法等，这是不是与通常所说的“千里眼”有点类似呢?

在分布式领域中，一个系统由很多服务组成，不同的服务由各自的进程单独负责。因此，远程调用在分布式通信中尤为重要。

根据进程是否部署在同一台机器上，远程调用可以分为如下两类：

本地过程调用（Local Procedure Call，LPC），是指同一台机器上运行的不同进程之间的互相通信，即在多进程操作系统中，运行的不同进程之间可以通过 LPC 进行函数调用。

远程过程调用（Remote Procedure Call，RPC），是指不同机器上运行的进程之间的相互通信，某一机器上运行的进程在不知道底层通信细节的情况下，就像访问本地服务一样，去调用远程机器上的服务。

注意
本地调用和本地过程调用不是一个东西！

RPC是微服务的基础，这是毋庸置疑的。

RPC是不是只能用在所谓的微服务里呢?

当然不是，只要涉及到网络通信，我们就有可能用到RPC。

我们看两个例子：

例 1：大型分布式应用系统可能会依赖消息队列、分布式缓存、分布式数据库以及统一配置中心等，应用程序与依赖的这些中间件之间都可以通过 RPC 进行通信。比如 etcd，它作为一个统一的配置服务，客户端就是通过 gRPC 框架与服务端进行通信的。

例 2：我们经常会谈到的容器编排引擎 Kubernetes，它本身就是分布式的，Kubernetes 的 kube-apiserver 与整个分布式集群中的每个组件间的通讯，都是通过 gRPC 框架进行的。

所以说，RPC 的应用场景还是非常广泛的。既然应用如此广泛，那它的核心价值又在哪里呢?

RPC的核心价值

RPC的核心价值是解决分布式系统的通信问题的一大利器。

分布式系统中的网络通信一般会采用四层的TCP或者七层的HTTP协议，而前者又占了大多数。
这主要得益于TCP协议的稳定性和高效性。

网络通信是一个复杂的过程，而RPC对这整个过程做了包装，使得开发更简单，同时也让通信变得更可靠。

为什么我们需要RPC

在上面的RPC的核心价值里头相信已经说的很清楚了。

如何学习RPC

可以用“逐步深入”这四个字来概括学习方式。

说起来也特别简单。当我们认识到，使用 RPC 就可以像调用本地一样发起远程调用，用它可以解决通信问题，这时候我们肯定要去学序列化、编解码以及网络传输这些内容。

把这些内容掌握后，你就会发现，原来这些只是 RPC 的基础，RPC 还有更吸引人的点，它真正强大的地方是它的治理功能，比如连接管理、健康检测、负载均衡、优雅启停机、异常重试、业务分组以及熔断限流等等。突然间，你会感觉自己走进了一个新世界，这些内容会成为你今后学习 RPC 的重点和难点。

这个逐步深入的过程，一定离不开真实的实践场景。学习知识，解决问题，遇到新问题，继续学习，不断解决问题，最后你会发现自己的学习曲线大概是这样的。

学习的时候不要依赖框架，先要了解基本原理以及关键的网络通信部分。

之后我们再学习 RPC 的重点和难点，了解 RPC 框架中的治理功能以及集群管理功能等；
这个时候你已经很厉害了，但这还不是终点，我们要对 RPC 活学活用，学会提升 RPC 的性能以及它在分布式环境下如何定位问题等等。