导读：微服务架构下的分布式概览

微服务架构的演变

微服务是一种服务间松耦合的、每个服务之间高度自治并且使用轻量级协议进行通信的可持续集成部署的分布式架构体系。这一句包含了微服务的特点，微服务架构和其他架构有什么区别？以下对比一些常见的架构。

单体架构

单体架构是最简单的软件架构，常用于传统的应用软件开发以及传统 Web 应用。传统 Web 应用，一般是将所有功能模块都打包（jar、war）在一个 Web 容器（JBoss、Tomcate）中部署、运行。随着业务复杂度增加、技术团队规模扩大，在一个单体应用中维护代码，会降低开发效率，即使是处理一个小需求，也需要将所有机器上的应用全部部署一遍，增加了运维的复杂度。

SOA 架构

当某一天使用单体架构发现很难推进需求的开发、以及日积月累的技术债时，很多企业会开始做单体服务的拆分，拆分的方式一般有水平拆分和垂直拆分。垂直拆分是把一个应用拆成松耦合的多个独立的应用，让应用可以独立部署，有独立的团队进行维护；水平拆分是把一些通用的，会被很多上层服务调用的模块独立拆分出去，形成一个共享的基础服务，这样拆分可以对一些性能瓶颈的应用进行单独的优化和运维管理，也在一定程度上防止了垂直拆分的重复造轮子。

SOA 也叫面向服务的架构，从单体服务到 SOA 的演进，需要结合水平拆分及垂直拆分。SOA 强调用统一的协议进行服务间的通信，服务间运行在彼此独立的硬件平台但是需通过统一的协议接口相互协作，也即将应用系统服务化。举个易懂的例子，单体服务如果相当于一个快餐店，所有的服务员职责都是一样的，又要负责收银结算，又要负责做汉堡，又要负责端盘子，又要负责打扫，服务员之间不需要有交流，用户来了后，服务员从前到后负责到底。SOA 相当于让服务员有职责分工，收银员负责收银，厨师负责做汉堡，保洁阿姨负责打扫等，所有服务员需要用同一种语言交流，方便工作协调。

微服务和 SOA

微服务也是一种服务化，不过其和 SOA 架构的服务化概念也是有区别的，可以从以下几个关键字来理解：

- 松耦合：每个微服务内部都可以使用 DDD（领域驱动设计）的思想进行设计领域模型，服务间尽量减少同步的调用，多使用消息的方式让服务间的领域事件来进行解耦。
- 轻量级协议：Dubbo 是 SOA 的开源的标准实现之一，类似的还有像 gRPC、Thrift 等。微服务更倾向于使用 Restful 风格的 API，轻量级的协议可以很好得支持跨语言开发的服务，可能有的微服务用 Java 语言实现，有的用 Go 语言，有的用 C++，但所有的语言都可以支持 Http 协议通信，所有的开发人员都能理解 Restful 风格 API 的含义。
- 高度自治和持续集成：从底层的角度来说，SOA 更加倾向于基于虚拟机或者服务器的部署，每个应用都

部署在不同的机器上，一般持续集成工具更多是由运维团队写一些 Shell 脚本以及提供基于共同协议（比如 Dubbo 管理页面）的开发部署页面。微服务可以很好得和容器技术结合，容器技术比微服务出现得晚，但是容器技术的出现让微服务的实施更加简便，目前 Docker 已经成为很多微服务实践的基础容器。因为容器的特色，所以一台机器上可以部署几十个、几百个不同的微服务。如果某个微服务流量压力比其他微服务大，可以在不增加机器的情况下，在一台机器上多分配一些该微服务的容器实例。同时，因为 Docker 的容器编排社区日渐成熟，类似 Mesos、Kubernetes 及 Docker 官方提供的 Swarm 都可以作为持续集成部署的技术选择。

其实从架构的演进的角度来看，整体的演进都是朝着越来越轻量级、越来越灵活的应用方向发展，甚至到近两年日渐成熟起来的 Serverless（无服务）架构。从单体服务到分层的服务，再到面向服务、再到微服务甚至无服务，对于架构的挑战是越来越大。

微服务架构和分布式

微服务架构属于分布式系统吗？答案是肯定的。微服务和 SOA 都是典型的分布式架构，只不过微服务的部署粒度更细，服务扩展更灵活。

理解微服务中的分布式

怎样理解微服务中的分布式？举一个招聘时一个同学来面试的例子。A 同学说，目前所在公司在做从单应用到微服务架构迁移，已经差不多完成了。提到微服务感觉就有话题聊了，于是便问“是否可以简单描述下服务拆分后的部署结构、底层存储的拆分、迁移方案？”。于是 A 同学说，只是做了代码工程结构的拆分，还是原来的部署方式，数据库还是那个库，所有的微服务都用一个库，分布式事务处理方式是“避免”，尽量都同步调用……于是我就跟这位同学友好地微笑说再见了。

微服务的分布式不仅仅是容器应用层面的分布式，其为了高度自治，底层的存储体系也应该互相独立，并且也不是所有的微服务都需要持久化的存储服务。一个“手机验证码”微服务可能底层存储只用一个 Redis；一个“营销活动搭建页面”微服务可能底层存储只需要一个 MongoDB。

微服务中的分布式场景除了服务本身需要有服务发现、负载均衡，微服务依赖的底层存储也会有分布式的场景：为了高可用性和性能需要处理数据库的复制、分区，并且在存储的分库情况下，微服务需要能保证分布式事务的一致性。

课程背景

微服务架构的技术体系、社区目前已经越来越成熟，所以在初期选择使用或者企业技术体系转型微服务的时候，需要了解微服务架构中的分布式的问题：在所有服务都是更小单元的部署结构时，一个请求需要调动更多的服务资源，怎样获得更好的性能？当业务规模增大，需要有地理分布不同的微服务集群时，其底层的数据存储集群是多数据中心还是单数据集群？数据存储如何进行数据复制？业务数据达到大数据量时怎样进行数据的分区？分布式事务怎样保证一致性？不同程度的一致性有什么差别？基于容器技术的服务发现怎么处理？应该用哪些 RPC 技术，用哪些分布式消息队列来完成服务通信和解耦？那么多的分布式技术框架、算法、服务应该选哪个才适合企业的业务场景？

本课程从微服务不得不面对和解决的分布式问题出发，包含分布式技术的一系列理论以及架构模型、算法的

介绍，同时结合技术选型和实践应用，提供一系列解决方案的梳理。相信阅读完整个课程，你会对微服务的分布式问题有个系统地理解。本课程会对微服务的分布式场景问题一一击破，为你提供解决思路。

课程内容

本课程示例代码地址如下：

- Microservice
- API-gateway

微服务架构下的分布式场景及方案——分布式系统的问题

- 引出分布式系统的可能问题：节点故障、网络延迟，结合错误检测的可行方案进行介绍；
- 分布式中的时间和顺序的问题，以及标量时钟和向量时钟的实现。

微服务架构下的分布式场景及方案——分布式数据存储

- 分布式数据存储的技术选型、关系型数据库以及一些流行的 NoSQL 技术介绍（MongoDB、Redis、Neo4j 及 Cassandra 等）；
- 分布式存储技术使用的数据结构，了解底层数据存储原理（HashTable、SSTable、LSMTree、BTree 等）；
- 各个存储方案使用的场景以及对比。

微服务架构下的分布式场景及方案——数据复制

- 对于大规模存储集群，需要进行数据库的复制、排除单点故障；
- 数据复制的模型和实现以及几种复制日志的实现方式；
- 主备同步、主主复制、多数据中心的数据复制方案；
- 数据复制中的读写一致性问题以及写冲突问题的解决；
- 介绍以 MySQL 为例延伸集群数据复制方案。

微服务架构下的分布式场景及方案——数据分区

- 当单个领域模型维度的数据已经到一定规模时，需要进行数据分区，减轻单库压力。数据分区和分表又有哪些不同？数据分区可以如何实现？
- 以 MySQL 的分区策略为例介绍不同分区策略的实现。
- 数据分区后，请求的路由有哪些解决方案？会展开介绍不同的方案又有什么差别。

微服务架构下的分布式场景及方案——服务发现和服务通信

- 基于容器技术的微服务体系，怎样选择服务发现、负载均衡的技术实现？不同的服务发现的技术有什么区别，如何选型？
- 为了达到松耦合的目的以及基于 DDD 思想，微服务之间减少服务调用可以通过哪些技术实现？API

Gateway 可以用哪些技术框架实现？远程调用可以有哪些技术框架？怎样提高同步通信的性能？
- 分布式的消息队列都有哪些开源、商业实现？应该怎样选择适合的消息队列？
- 使用 DDD 思想应该如何应对服务通信，如何在实践中应用 DDD？

微服务架构下的分布式场景及方案——分布式存储集群的事务

- 理解分布式中的事务以及本地事务的基础概念；
- 分布式存储的隔离级别以及各个 DB 支持的隔离方案的实现原理；
- 以 MySQL InnoDB 中的 MVCC 为例看并发控制的在 MySQL 的实现，学习存储系统对于分布式事务的实现思想。

微服务架构下的分布式场景及方案——分布式一致性

- 了解分布式系统的一致性有哪些问题以及一致性的几种实现程度的模型：线性一致性（强一致性）、顺序一致性及因果一致性、最终一致性；
- 分布式一致性相关的理论 CAP（CA、CP、AP 的相关算法）的介绍以及适合用于哪些实践；
- 介绍 FLP 不可能结果，以及 BASE 理论。

微服务架构下的分布式场景及方案——分布式事务实践

- 了解微服务中分布式事务的问题；
- 介绍强一致性的实践：二阶段、三阶段。2PC、3PC 的优缺点和限制，XA 协议的介绍和实践方案，以及最终一致性实践：TCC 模型和实践方案；
- 分布式锁的实现模型和实践方案；
- 基于微服务下的分布式事务实践案例分析。

微服务架构下的分布式场景及方案——共识问题

- 了解为什么分布式场景下有共识问题；
- 介绍共识算法和全局消息广播的实现，公式算法的基础：leader 选举和 quorum 算法，以及一些已实现的算法的介绍和对比：VSR、Raft、Paxos、ZAB；
- 共识算法在微服务体系的应用场景介绍：服务发现、一致性 kv 存储（Etcd、Zk）以及技术的选型如何权衡一致性的追求和性能。

微服务架构下的分布式场景及方案——架构设计

- 了解了很多分布式的问题和解决方案之后，回归微服务架构模型、技术选型、再回顾下微服务的弊端和特点；
- 微服务体系的架构要素分析：安全、伸缩性、性能、可用性、扩展性；
- 结合团队、业务场景的 DDD 实践和总结。

课程寄语

- 如果你是一位开发工程师，相信阅读完本系列课程，将会了解很多分布式系统的理论知识，同时也会理解一些分布式存储、中间件技术的原理，对工作中的分布式架构会有体系化的清晰认知。
- 如果你是一位架构师，本系列课程提供了对于分布式系统问题的全面梳理，以及一些技术背后的理论，结合实践和目前业界先进的方案，对于技术选型和架构搭建提供了参考。