微服务架构如何演变的？

微服务架构与 SOA 架构的区别

1.微服务架构基于 SOA 架构 演变过来,继承 SOA 架构的优点,在微服务架构中去除 SOA 架构中的 ESB 企业服务总线，采用 http+json（restful）进行传输。
2.微服务架构比 SOA 架构粒度会更加精细，让专业的人去做专业的事情（专注），目的提高效率，每个服务于服务之间互不影响，微服务架构中，每个服务必须独立部署，微服务架构更加轻巧，轻量级。
3.SOA 架构中可能数据库存储会发生共享，微服务强调独每个服务都是单独数据库，保证每个服务于服务之间互不影响。
4.项目体现特征微服务架构比 SOA 架构更加适合与互联网公司敏捷开发、快速迭代版本，因为粒度非常精细。

在很多大型互联网公司都会根据当前的热点新闻,开发项目,这个项目开发模式为突击团队,肯定会采用到微服务架构模式。

微服务架构架构模式的优缺点

优点：
1. 让专业的人做专业的事情
2. 项目都会独立部署，互不影响
3. 适合于互联网公司敏捷开发模式
4. 降低耦合度

缺点：
1. 适合中大型互联网公司实现
2. 服务比较维护运营的成本非常高
3. 很多分布式解决方案的问题

SpringCloud 第一代与二代的区别

SpringCloud 第一代属于 Netflix 和组件

SpringCloud 第二代属于阿里巴巴 Cloud+优秀国内开源组件组合+自己研发

名称	SpringCloud第一代	SpringCloud第二代
网关	Spring Cloud Zuul	Spring Cloud Gateway
注册中心	Eureka(不再更新), Consul,ZK	阿里Nacos, 拍拍贷radar等可选
配置中心	SpringCloud Config	阿里Nacos, 携程Apollo, 随行付Config
客户端软负载均衡	Ribbon	Spring-cloud-loadbalancer
熔断器	Hystrix	spring-cloud-r4j(Resilience4J), 阿里 Sentinel

SpringCloud 第一代:
SpringCloud Config 分布式配置中心
SpringCloud Netflix 核心组件
Eureka:服务治理
Hystrix:服务保护框架
Ribbon:客户端负载均衡器
Feign: 基于 ribbon 和 hystrix 的声明式服务调用组件
Zuul: 网关组件,提供智能路由、访问过滤等功能。

SpringCloud 第二代（自己研发）和优秀的组件组合:

Spring Cloud Gateway 网关
Spring Cloud Loadbalancer 客户端负载均衡器
Spring Cloud r4j(Resilience4J) 服务保护

Spring Cloud Alibaba Nacos 服务注册
Spring Cloud Alibaba Nacos 分布式配置中心
Spring Cloud Alibaba Sentinel 服务保护
SpringCloud Alibaba Seata 分布式事务解决框架
Alibaba Cloud OSS 阿里云存储
Alibaba Cloud SchedulerX 分布式任务调度平台
Alibaba Cloud SMS 分布式短信系统

微服务架构中服务治理核心概念

注册中心在整个的微服务架构中，最重要!

传统方式接口调用地址都是写死的，如果发生变化的情况下，生产者（调用者）也需要同步

的修改该为最新的接口地址

在微服务架构服务与服务之间 url 管理是非常复杂的，所以采用注册中心统一管理我们的接口调用。

远程调用与本地调用区别

远程调用与本地调用的区别：

远程调用就是采用网络通讯的模式访问接口。
本地调用：从我们注册中心上远程获取接口地址，在本地实现 rpc 远程调用。
写法为让人感觉类似于调用 jar 包的形式，实际上采用动态代理技术帮助获取返回接口。

用过 dubbo 有点类似于调用 jar 包

分布式注册中心设计原理

1. 生产者将自己的接口地址注册到我们注册中心存放。
2. 消费者从我们注册中心获取远程调用地址
3. 消费者获取到地址之后在本地实现 rpc 远程调用
4. 采用监视器监控每次调用接口成功率

Eureka 与 Zookeeper 区别

相同点、不同点、中心化思想

三者都可以实现分布式注册中心框架

不同点：
Zookeeper 采用 CP 保证数据的一致性的问题，原理采用(ZAP 原子广播协议)，当我们 ZK 领导者因为某种情况下部分节点出现了故障，会自动重新实现选举新的领导角色，整个选举的过程中为了保证数据一致性的问题，客户端暂时无法使用我们的 Zookeeper，那么这以为着整个微服务无法实现通讯。

Eureka 采用 AP 设计思想实现分布式注册中心，完全去中心化、每个节点都是相等，采用你

中有我、我中有你相互注册设计思想， 只要最后有一台 Eureka 节点存在整个微服务就可以实现通讯。

Nacos 与 Eureka、Zookeeper 的区别

Nacos 从 1.0 版本选择 Ap 和 CP 混合形式实现注册中心,默认情况下采用 Ap,CP 则采用 Raft 协议实现保持数据的一致性。
如果选择为 Ap 模式，注册服务的实例仅支持临时模式，在网络分区的的情况允许注册服务实例，选择 CP 模式可以支持注册服务的实例为持久模式，在网络分区的产生了抖动情况下不允许注册服务实例。

discoveryClient 的作用有那些

客户端接口，使用 discoveryclient 直接从我们注册中心上根据服务名称获取接口地址。

我自己有一个 cas 单点登陆如何整合 nacos(⊙ o ⊙)啊！？

本地负载均衡的实现原理

本地负载均衡器实现的原理：就是根据服务名称从注册中心获取接口地址，让后在本地采用负载算法 获取一个地址实现 rpc 远程调用。

Member value: 127.0.1.1:8080/127.0.1.1:8081

8080
8081
8080
8081

本地负载均衡的算法

轮训、权重、取模、随机、hash 等。
Nginx 是客户端所有的请求统一都交给我们的 Nginx 处理，让后在由 Nginx 实现负载均衡转发，属于服务器端负载均衡器。

本地负载均衡器是从注册中心获取到集群地址列表，本地实现负载均衡算法，既本地负载均衡器。
应用场景的：
Nginx 属于服务器负载均衡，应用于 Tomcat/Jetty 服务器等，而我们的本地负载均衡器，应用于在微服务架构中 rpc 框架中，rest、openfeign、dubbo。

本地负载均衡与 Nginx 的区别

Nginx 是客户端所有的请求统一都交给我们的 Nginx 处理，让后在由 Nginx 实现负载均衡转发，属于服务器端负载均衡器。
本地负载均衡器是从注册中心获取到集群地址列表，本地实现负载均衡算法，既本地负载均衡器。
应用场景的：
Nginx 属于服务器负载均衡，应用于 Tomcat/Jetty 服务器等，而我们的本地负载均衡器，应用于在微服务架构中 rpc 框架中，rest、openfeign、dubbo。

LoadBalancedClient 的作用

SpringCloud 内置的负载均衡器

openfeign 客户端与 feign 客户端区别

opnfen 属于 SpringCloud 第二代自己研发
feign 属于 SpringCloud 第一代。
两者间的用法都是一样的