1. 系统架构的演变
2. 集中式架构
3. 概述

当网站的流量很小时，只需要一个应用，将所有的功能都部署在一起，以减少部署节点和成本。

1. 优点

开发速度快，维护成本低。

1. 缺点

代码耦合度高，维护困难，无法水平扩展，容错率低，并发能力差。

1. 垂直拆分
2. 概述

由于访问量逐渐增大，单一的应用无法满足需求，需要根据业务功能对系统进行拆分。

1. 优点

系统拆分实现了流量分担，解决了并发问题，可以针对不同的模块进行优化，方便水平扩展，负载均衡，提高容错率。

1. 缺点

系统间相互独立，有很多重复开发工作。

1. 分布式服务
2. 概述

当垂直应用越来越多时，应用之间的交互也增多了，我们可以将核心的业务抽取出来作为独立的服务，形成基础服务中心，这样其他应用调用这些常用的服务就很方便。

1. 优点

抽取了基础服务，提高代码的复用和开发效率。

1. 缺点

系统间耦合度增高，调用关系复杂，难以维护。

1. 面向服务结构（SOA）
2. 概述

包含多个服务，服务之间通过相互依赖提供一系列的功能。一个服务通常以独立的形式存在于操作系统进程中，各个服务之间通过网络调用。各个服务之间通过ESB（企业服务总线）连接，为了集成不同的服务，不同协议的服务，ESB做了消息的转化解释和路由工作，让不同的服务互联互通。

1. 缺点

服务粒度较大，ESB集成整合所有服务和协议，数据使得部署，维护困难，且所有的服务都通过一个通道通信，降低了通信的速度。

1. 微服务架构
2. 概述

微服务架构是使用一套小服务来开发单个应用的方式，每个服务基于单一业务能力构建，运行在自己的进程中，并使用轻量级机制通信，通常是HTTP API，并能够通过自动化部署来独立部署。这些服务可以使用不同语言，存储技术，并保持最低限度的集中式管理。

1. 流程

终端通过一个Gateway网关来请求服务，服务通过服务注册中心进行服务注册和管理。网关是一个服务器，是系统的唯一入口，它为每个客户端提供一个定制的API，所有的客户端和消费端都是通过统一的网关接入微服务的，在网关层处理所有的非业务功能。如身份验证，监控，负载均衡，缓存，等等。网关提供RESTful/HTTP的方式访问服务。

1. 特点
2. 每个服务对应单一的业务。
3. 服务拆分粒度很小。
4. 每个服务只需要提供标准接口API，不论该服务如何实现。
5. 服务间相互独立，互不干扰。
6. 前后端分离，提供统一的Rest接口，后端不需要再为PC，移动端开发不同的接口。
7. 数据库分离，每个服务使用自己的数据源。
8. 微服务与SOA的区别

微服务架构和SOA都是对系统进行拆分，SOA中有ESB作为中心总线，而微服务做了去中心化处理，将ESB去掉了，更加灵活，着重分散管理，而不是中央管理。

1. 服务调用方式
2. RPC和HTTP

服务之间的远程调用方式有两种：RPC和HTTP。

1. RPC

RPC全称为 Remote Produce Call远程过程调用，RPC是基于Socket，工作在会话层，自定义数据格式，速度快，效率高。代表产品为webservice,dubbo。

1. HTTP

http是一种网络传输协议，基于TCP,工作在应用层，规定了数据传输的格式。客户端和服务器的通信基本都是HTTP协议，缺点是消息封装臃肿，优点是对于服务的提供和调用没有技术限定，不会限制使用什么语言，技术，HTTP只负责消息的传递。Rest风格，Spring Cloud套件就是通过HTTP协议实现的。

1. RPC与HTTP的区别

RPC是根据语言的API来定义的，而不是根据网络的应用来定义的。它会限制使用语言，技术的灵活性。如果使用的JAVA语言，那么使用Dubbo比较合适。

如果不只是使用一种语言，技术，那么使用HTTP方式来实现服务间的调用更好。

1. Spring RestTemplate的使用

一般有三种http客户端工具类包可以方便地进行http服务调用：httpClient，okHttp，JDK原生URLConnection。Spring提供了RestTemplate的工具类对上述的三种http客户端工具进行了封装，可以直接在Spring项目中使用RestTempplate进行服务调用。

1. Spring Cloud
2. 概述

Spring Cloud是Spring旗下的项目，完全支持Spring Boot的开发，用很少的配置就能够完成微服务框架的搭建。Spring Cloud能够将其他非常流行的技术整合到一起，如配置管理，服务发现，智能路由，负载均衡，熔断器，控制总线，集群状态等功能；协调分布式环境中各个系统，为各类服务提供模板性配置，

1. 组件构成

Spring Cloud不是一个组件，而是许多组件的集合。主要的组件为：注册中心（Eureka），服务网关（Zuul, Gateway），负载均衡（Ribbon），服务调用（Feign），熔断器（Hystrix或Resilience4j）。当我们需要使用某些组件时，直接在项目中添加相关的启动器依赖即可。

1. 流程

当请求通过网关进入后，负载均衡从服务注册中心中获取请求的地址列表，并基于负载均衡算法来选择一个地址来访问；如果服务之间有调用的需求，那么通过服务调用Feign进行调用；如果服务需要修改配置，那么从配置服务器中实时地更新配置。

1. 版本

Spring Cloud 的版本不是普通的数字，而是使用伦敦地铁站的名字来命名的（脑残操作）。

”