目录

[**第一章** **微服务简介** 2](#_Toc47441786)

[**1.1** **单体架构** 2](#_Toc47441787)

[**1.1.1** **单体架构得优缺点** 2](#_Toc47441788)

[**1.2** **微服务是什么** 2](#_Toc47441789)

[**1.3** **微服务具备得特性** 2](#_Toc47441790)

[**1.4** **微服务优缺点** 2](#_Toc47441791)

[**1.5** **微服务设计原则** 2](#_Toc47441792)

[**第二章** **微服务技术知识** 2](#_Toc47441793)

[**2.1 Spring Cloud五大组件** 2](#_Toc47441794)

[**2.2** **Netflix Eureka** 2](#_Toc47441795)

[**2.2.1** **Eureka基本知识** 2](#_Toc47441796)

1. **微服务简介**
   1. **单体架构**
      1. **单体架构得优缺点**

优点：1.便于开发2.易于测试3.易于部署

缺点：1.复杂性高2.开发速度慢3.扩展性差4.代码部署周期长5.如果系统出现内部错误，有可能直接可能造成系统宕机。6.阻碍技术创新

* 1. **微服务是什么**

把一个庞大的系统拆分为若干个小模块，每个模块都可以独立应用服务，各个服务之间采用HTTP等轻量级的机制进行通讯。这些服务围绕业务功进行构建，通过全自动部署进行独立部署。各个服务可以采用不同语言进行编写，使用不同的数据存储技术。

* 1. **微服务具备得特性**

1. 每个服务可独立运行在自己的进程里；
2. 一系列独立运行的微服务共同构建起了整个系统；
3. 每个服务为独立的业务开发，一个微服务一般完成某个特定的功能，比如：订单管理、用户管理等；
4. 微服务之间通过一些轻量的通信机制进行通信，例如通过REST API或者RPC的方式进行调用。
   1. **微服务优缺点**

优点：

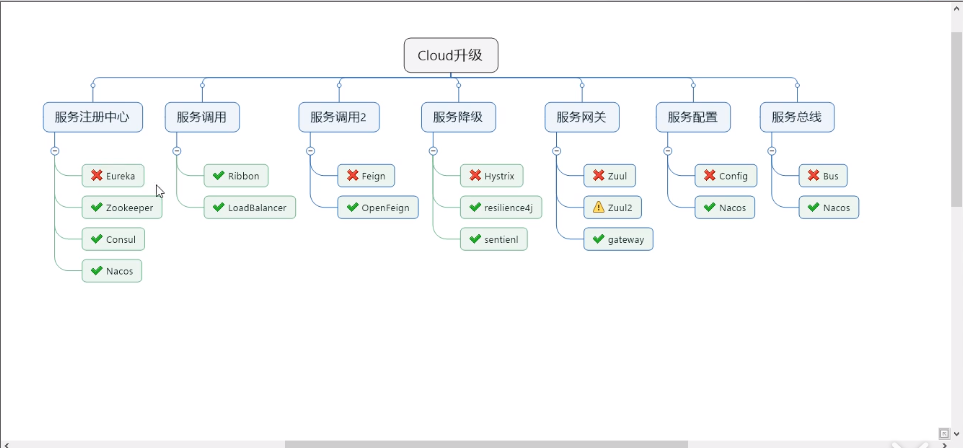
1. 易于开发和维护
2. 启动较快
3. 局部修改容易部署
4. 技术栈不受限
5. 按需伸缩

缺点：

1. 微服务过多，服务治理成本高，不利于系统维护
2. 分布式系统开发的技术成本高（容错、分布式事务等），对团队挑战大。
   1. **微服务设计原则**
3. 单一职责原则
4. 服务自治原则
5. 轻量级通信原则
6. 接口明确原则
7. **微服务技术知识**

**2.1 Spring Cloud五大组件**

1. 服务注册中心——Netflix Eureka——Zookeeper——Consul——AlibabaNacos
2. 客户端负载均衡——Netflix Ribbon——LoadBalance
3. 断路器——Netflix Hystrix——resilience4j——Sentinel
4. 服务网关——Netflix Zuul——Gateway
5. 分布式配置——Spring Cloud Config——AlibabaNacos

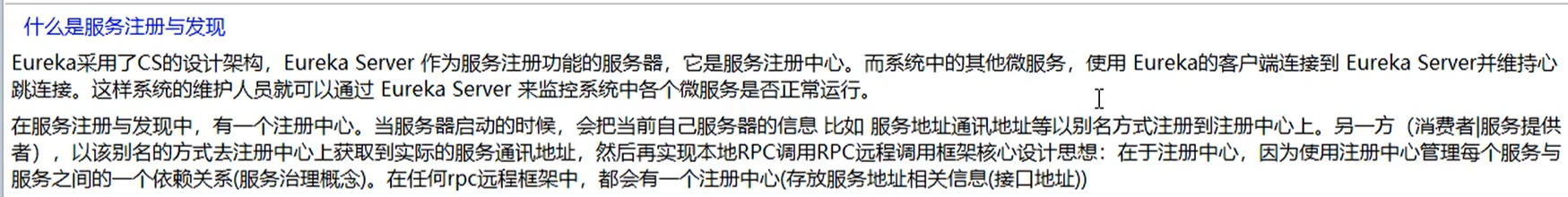


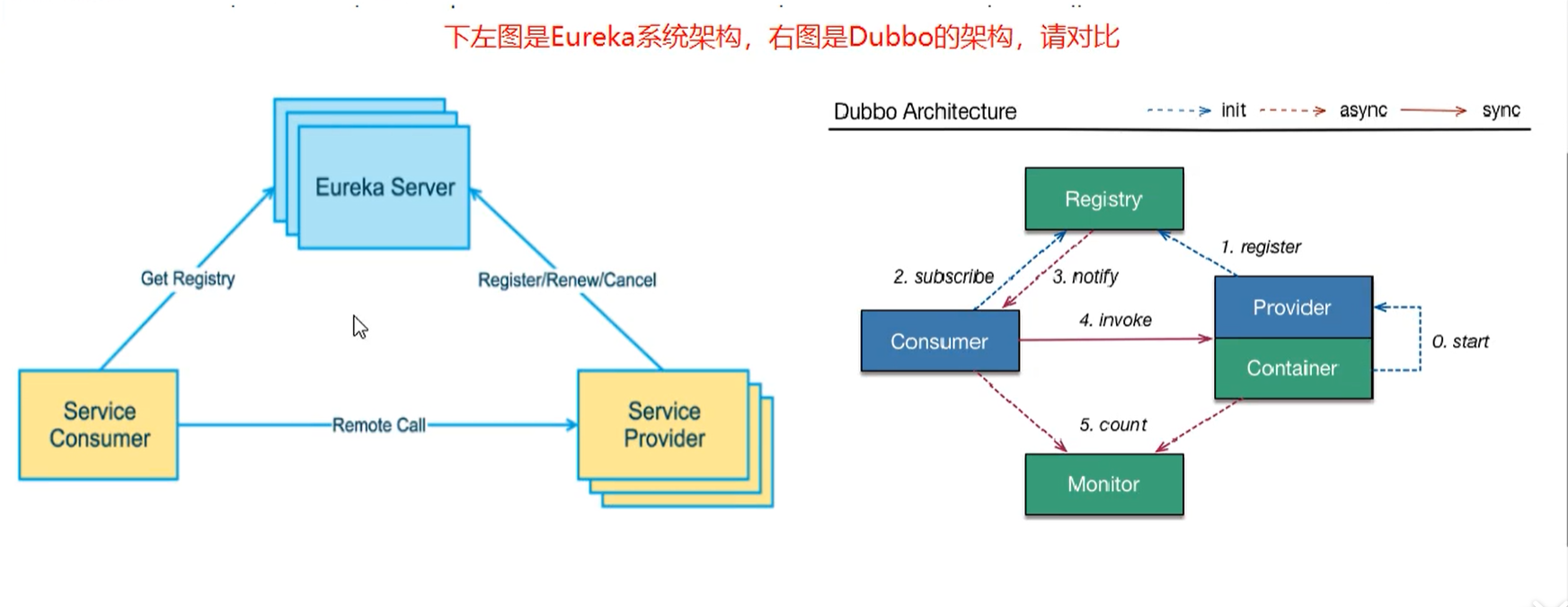
* 1. **Netflix Eureka**
     1. **Eureka基本知识**

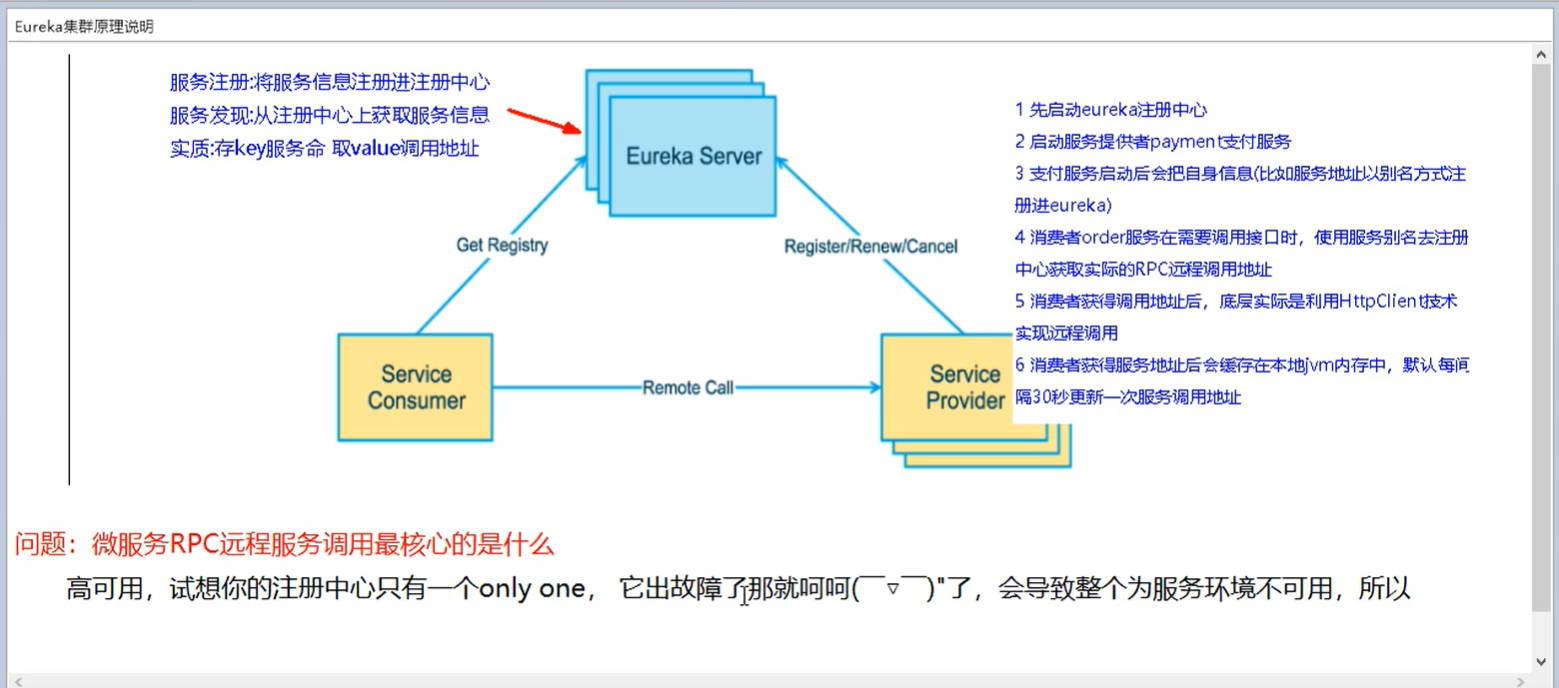
什么是服务治理？

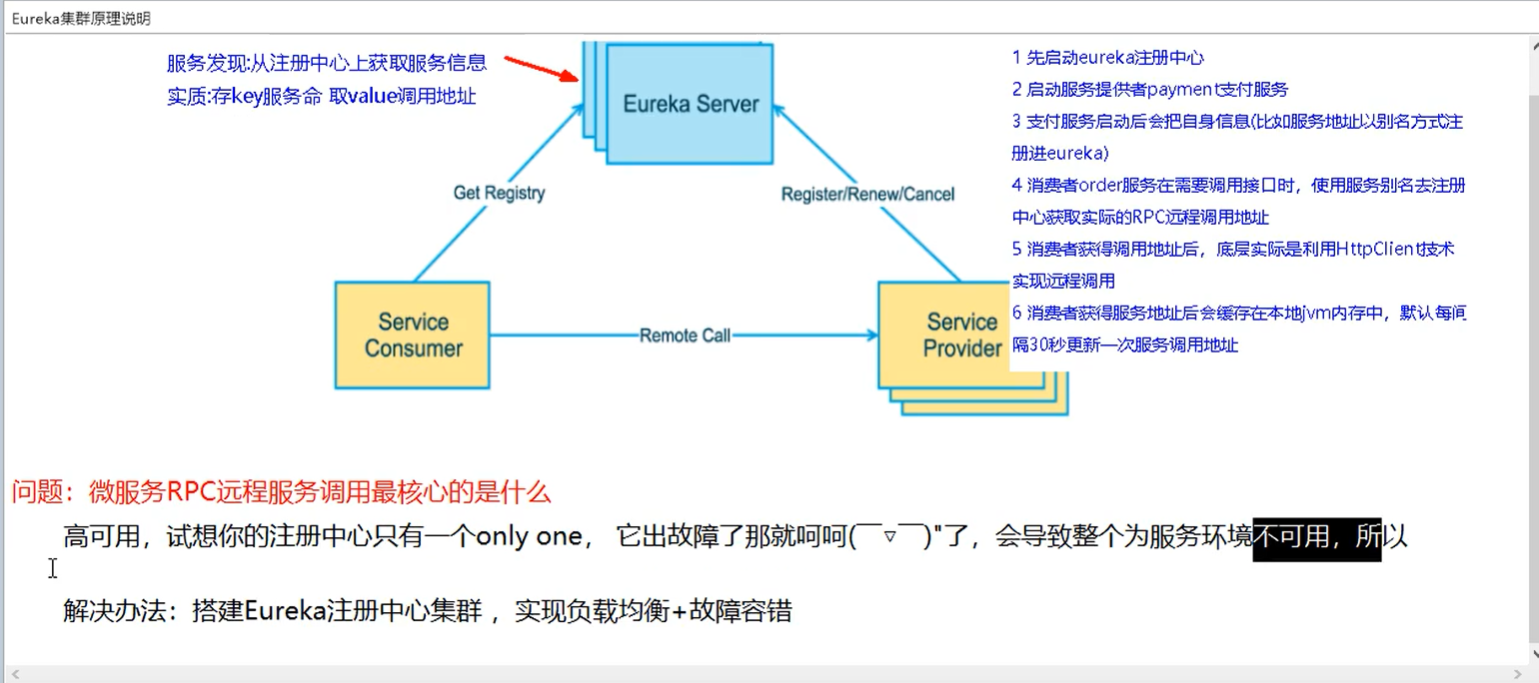
Spring Cloud封装了Netfix公司开发的Eureka模块来实现服务治理

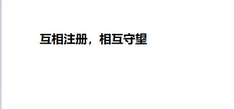
在传统得rpc远程调用框架中，管理每个服务之间依赖关系比较复杂，管理比较复杂，所以需要使用服务治理，服务之间以来关系，可以实现服务调用、负载均衡、容错等，实现服务发现与注册。

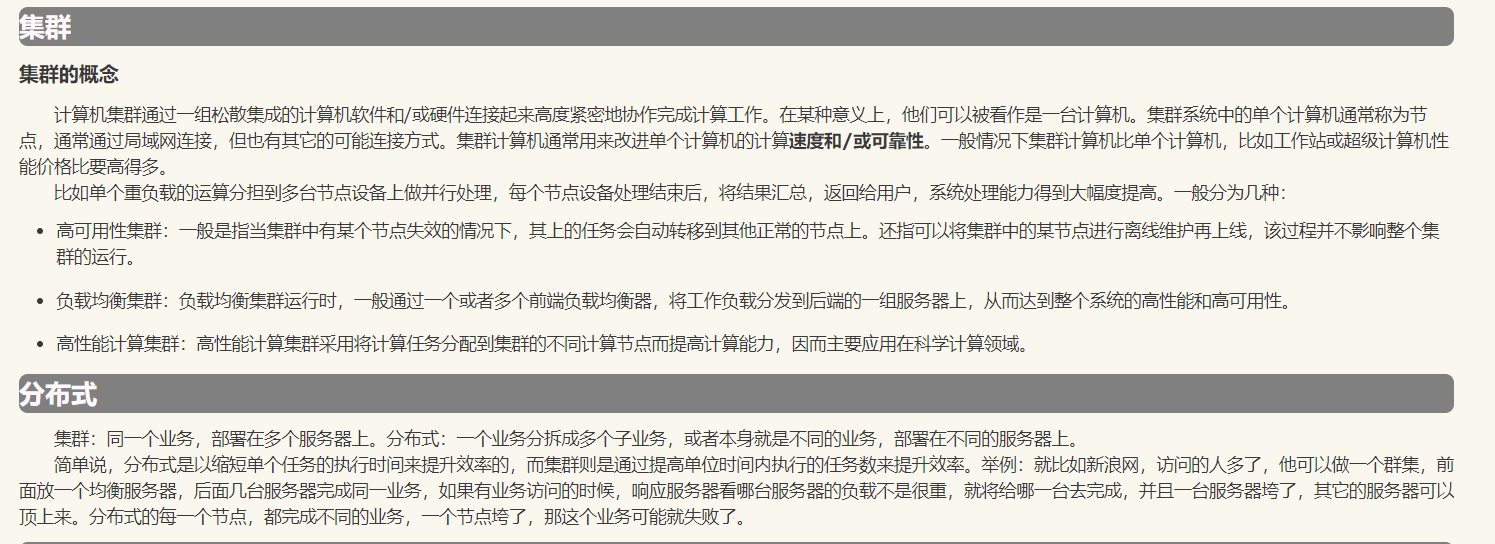




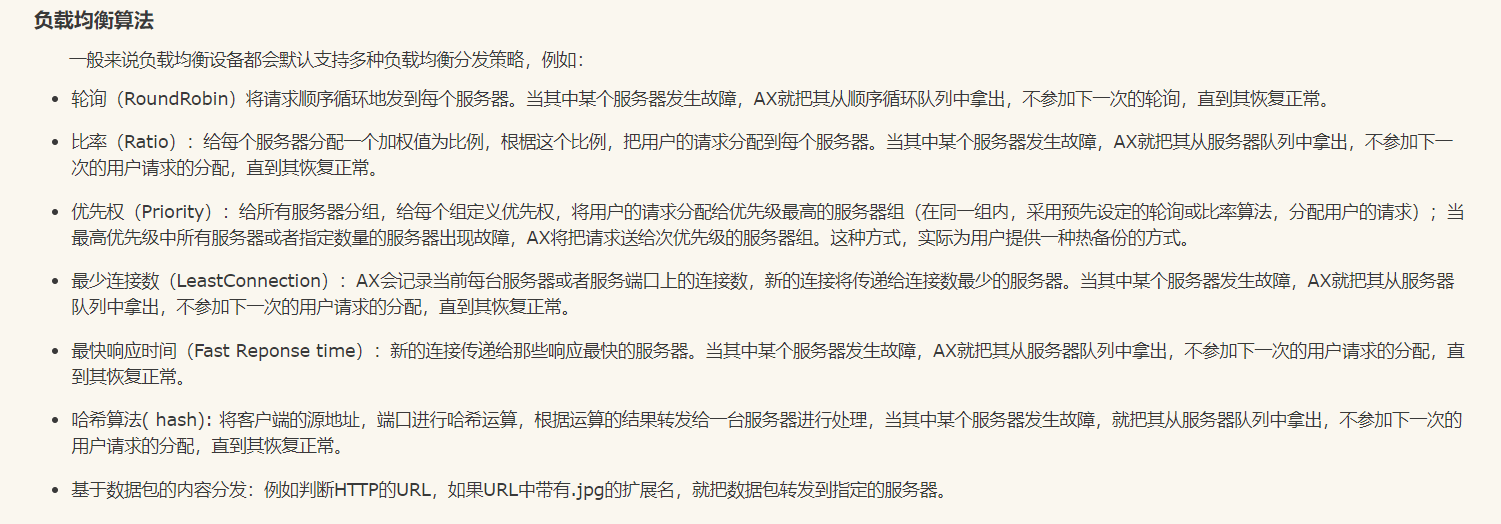


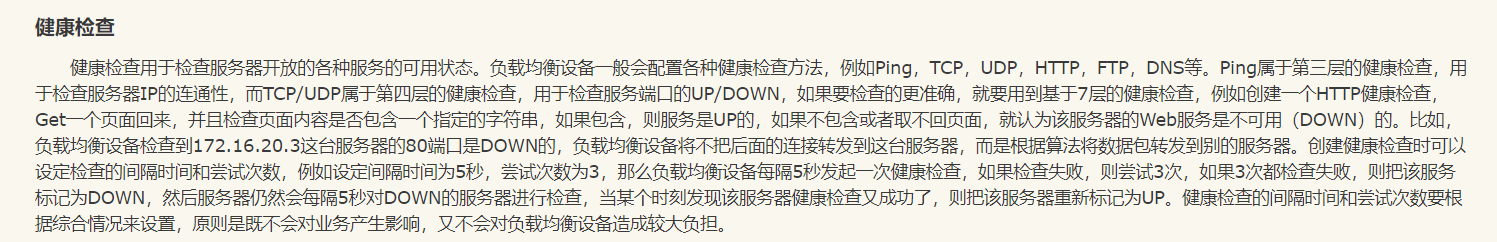












Ribbon默认得负载均衡算法：轮询算法

XI

**消息队列**

我们可以把消息队列比作是一个存放消息的容器，当我们需要使用消息的时候可以取出消息提供自己使用。消息队列是分布式系统中重要的组件，使用消息队列主要是为了通过异步处理提高系统性能和削峰、降低系统耦合性。

举个例子：

比如说客户下个订单操作，耗时很短。然后这理产品经理要加个优惠券、积分，最后是不是还得扣库存。如果在同一个业务实现的话那么耗时时间就要加长，客户体验就很差。此时我们需要用到消息队列，首先得把这些业务分别拆分出来，先降低耦合度。客户下个订单，立刻成功。把订单得信息push到消息队列中，其他业务可以异步得从队列中pull到订单信息。在同时进行业务处理。这样就达到了高性能，高可用。

流量削锋也是消息队列中的常用场景，一般在秒杀或团抢活动中使用广泛。  
应用场景：秒杀活动，一般会因为流量过大，导致流量暴增，应用挂掉。为解决这个问题，一般需要在应用前端加入消息队列。  
a、可以控制活动的人数  
b、可以缓解短时间内高流量压垮应用

用户的请求，服务器接收后，首先写入消息队列。假如消息队列长度超过最大数量，则直接抛弃用户请求或跳转到错误页面。  
秒杀业务根据消息队列中的请求信息，再做后续处理

消息队列好处：通过异步处理提高系统性能和流量削峰、降低系统耦合性。

缺点：重复性，顺序性。

重复性可以建立一个应用状态表去管理，每次操作的时候去对比一下，如果重复就不执行。