# 1,服务治理技术概述

1,服务注册与发现原理： 在任何rpc远程框架中，都会有一个注册中心

注册中心的概念： 存放服务地址相关信息；

Springcloud 中支持以下三种注册中心：

Eureka，Consul,(go语言编写)，Zookeeper

Dubbo 支持常用俩种 Redis和Zookeeper;

1. rpc远程调用框架核心社会思想：在于注册中心，因为注册中心管理每个服务与服务之间的一个依赖关系（服务治理概念）
2. 什么是服务治理

4,Ribbon 与 nginx 区别？

服务器端负载均衡Nginx

nginx是客户端所有请求统一交给nginx ，由nginx 进行实现负载均衡请求转发，属于服务器端负载均衡，，即请求有nginx 服务器端进行转发；

客户端负载均衡ribbon

Ribbon 是从eureka 注册中心服务器端上获取服务注册信息列表，缓存到本地，让后台在本地实现轮训负载均衡策略，即在客户端实现负载均衡。

应用场景的区别：

Nginx 适合于服务器端负载均衡，比如tomcat,Ribbon适合在微服务中rpc

远程调用中实现本地负载均衡，比如 dubbo, springcloud 中都是采用本地负载均衡。

# 2 Hystrix

1,Feign 客户端工具超时时间设置

1. Feign 客户端整合Ribbon 本地负载均衡
2. SpringCloud 微服务高可用
3. 服务保护利器Hystrix
4. 服务降级，服务隔离，服务熔断，服务雪崩效应概念
5. SpringCloud 整合Hystrix实现服务降级

1,服务雪崩效应：

解释： 默认情况下Tomcat只有一个线程池去处理的客户端发送的所有请求，这样的话在高并发的情况下，如果客户端所有的请求堆积到同一个服务器接口上，就会产生tomact的所有线程去处理该服务接口，可能会导致其他服务接口服务的访问；

1. Hystrix 服务保护框架，在微服务中Hystrix 能够为我们能够解决那些事情？
2. 断路器
3. 服务器降级
4. 服务熔断
5. 服务隔离机制
6. 服务雪崩效应：连环雪饼效应如果比较严重的话，可能会导致整个微服务接口

无法访问，所有服务都会瘫痪；

采用服务隔离机制 线程池隔离： 每个服务接口都有自己独立的线程池，每个线程池互不影响，缺点： cpu占用率高，不是所有的接口求取采用线程池隔离，核心关键接口；

课程内容：

1. 分布式配置中心的内容
2. 常见分布式配置中心
3. 搭建git环境存储配置中心
4. 搭建ConfigServer 配置文件中心
5. COnfigClient读取ConfigServer配置文件信息
6. SpringCloudConfig 实时刷新配置文件
7. SprigCloudConfig设计原理分析

# 接口网关

1,网关API接口，gateway(网关) 接口网关注意： 接口没有界面

2，接口什么背景下产生？ 在面向服务架构和微服务背景下产生，目的为了解耦，rpc远程调用中产生的。

3，接口如何分类：

开放接口：

其他机构合作伙伴进行调用（必须在外网访问）微信公众号开发等

需要通过 appid+appsocet生成accessToken进行通讯。对接支付开发，微信开发；

目的可以授权一些接口权限OAuth2.0 协议方式，第三方联合登录。

内部接口：

一般只能在局域网中进行访问，服务与服务调用之间关系都在同一个微服务系统中。

目的为保证安全问题：

现在一套公司的项目接口，你会如何设计

考虑：

接口权限（开发接口|内怒接口），考虑幂等特性，安全性（https）,防止篡改数据（验证签名），使用网关拦截， 接口实现黑白名单，接口使用http协议+json格式restful 目的为了跨平台。考虑高并发，对接口服务实现保护，服务降级，熔断，隔离之类的。最后使用swagger统一管理接口；

Nginx 与Zuul的区别；

相同点：

Zuul和nginx 都可以实现负载均衡。反向代理，过滤请求，实现网关效果；

不同点：

Nginx 采用C 语言编写；

Zuul采用java 语言编写：

ZUul 负载均衡的实现： 采用ribbon+eureka实现本地负载均衡；

Nginx 负载均衡实现： 采用服务器端实现负载均衡。

Nginx比zuul 功能会强大点因为nginx可以整合一些脚本语言（nginx+lua）

nginx适合于服务器端负载均衡，也可以实现网关

ZUUL适合微服务中实现网关，而且使用技术是java 语言

最好建议nginx+zuul实现网关

nginx作用实现反向代理

Zuul 对微服务实现网关拦截

网关做到事情：

1. 对所有的服务进行拦截
2. 网关安全控制，统一异常处理，xxs,sql注入
3. 权限控制，黑白名单，性能监控，日志打印；

# Doubbo优缺点：

优点：

1. 透明化的远程调用，像调用本地方法一样调用远程方法：只需要简单配置，没有API侵入。
2. 软负载均衡以及容错机制；可以在内网替代 nginx lvs 等硬件负载均衡器
3. 服务注册中心自动 注册&配置管理

不需要写死服务提供者的地址，注册中心基于接口名自动查询提供者ip

使用类似zookeeper等分布式协调服务作为服务注册中心，可以将绝大多数配置移入zookeeper集群  
5，服务接口监控和治理：

-Dubbo-admin 与dubbo-moitor提供了完善的服务接口管理和监控功能，针对不同应用的不同接口，可以多协议，多注册中心；

缺点： 只支持java

1. 在微服务rpc远程调用框架中，服务端负载均衡如何设计？

微服务的负载均衡不建议使用nginx或者lvs做负载均衡

1. 在微服务rpc远程调用框架中，服务的负载均衡都是采用本地的

负载均衡器，springcloud里面的负载均衡使用ribbon（本地负载均衡客户端）

3为什么rpc远程调用框架都是使用本地的负载均衡，而不是nginx

nginx如果采用neginx实现负载均衡的话，那么所以的请求都需要把服务先交给nginx，再有nginx统一实现负载均衡

本地负载均衡： 本地服务从注册中心获取拂去信息列表（缓存在jvm），以后在本地使用rpc远程调用的时候统一实现rpc进行负载均衡；

消息队列Rabbitmq的五种形式队列

1,点对点（简单）的队列

1. 工作队列模式
2. 发布订阅模式
3. 路由模式Routing
4. 通配符模式Topics

1,点对点模式： 一对一模式，一个生产者投递消息给队列，只能被允许有一个消费者进行消费， 注意： 如果消费集群的话，会进行均摊消费

kafka优点

高吞吐量，低延迟：kafka 每秒可以处理几十万条消息，他的延迟最低只有几毫秒

每个topic可以分多个partition ,consumer group 对partLiLion进行consume操作。

可扩展性： kafka 集群支持热扩展

持久性： 可靠性；消息被持久化到本地磁盘，并且支持数据备份防止数据丢失

容错性： 允许集群节点失败（若副本数量为n,则允许n-1个节点失败）

高并发： 支持数千个客户端同时读写；