1. SpringCloud
2. SpringCloud微服务与Dubbo的区别

dubbo的服务是通过RPC实现的，接口强依赖（RPC远程过程调用）通过TCP等一大堆复杂的过程

SpringCloud是使用REST API ,REST 相比RPC更为轻量化，服务方和调用方的依赖只是靠一纸契约，不存在代码级别的强依赖。SpringCloud组合了大量的组件

通过上面再几个环节上的分析，相信大家对Dubbo和Spring Cloud有了一个初步的了解。就我个人对这两个框架的使用经验和理解，打个不恰当的比喻：使用Dubbo 构建的微服务架构就像组装电脑，各环节我们的选择自由度很高，但是最终结果很有可能因为一条内存质量不行就点不亮了，总是让人不怎么放心，但是如果你是一 名高手，那这些都不是问题；而Spring Cloud就像品牌机，在Spring Source的整合下，做了大量的兼容性测试，保证了机器拥有更高的稳定性，但是如果要在使用非 原装组件外的东西，就需要对其基础有足够的了解。

1. SpringCloud与SpringBoot 区别

SpringBoot 是Spring快速配置脚手架，可以基于SpringBoot快速开发单个微服务，SpringBoot基于Spring，是Spring的学习和使用便可快速，不仅适合原有工程结构，更适合微服务开发

SpringCloud基于SpringBoot 为微服务体系开发中的架构问题，提供了一整套的解决方案，微服务注册与发现，服务消费，服务保护与熔断，网关，分布式调用追踪，分布式配置管理等。SpringCloud依赖于SpringBoot

1. Eureka
2. 服务器雪崩

在微服务架构中，根据业务来拆分成一个个的服务，服务与服务之间可以相互调用（RPC）,在SpringCloud可以用RestTemplate+Ribbon和Feign来调用。为了保证高可用，单个服务通常会集群部署。由于网络原因或者服务自身的原因，服务并不能保证100%可用，如果单个服务出现问题，调用这个服务就会出现线程阻塞，此时若有大量的请求涌入，Servlet容器的线程资源会被消耗完毕，从而导致服务器瘫痪。服务与服务之间的依赖性，故障会传播，会对整个微服务系统造成灾难性的严重后果，这就是服务故障雪崩效应

1. 如何防止雪崩

解决方案：对依赖做隔离，hystrix就是处理依赖隔离的框架，同时也是可以帮助我们做依赖服务的治理和监控。hystrix能够保证在一个依赖出问题的情况下，不会导致整体服务失败，避免级联故障，以提高分布式系统的弹性。“断路器”本身是一种开关装置，当某个服务单元发生故障之后，通过断路器的故障监控（类似熔断保险丝），向调用方返回一个符合预期的、可处理的备选响应（fallback），而不是长时间的等待或者抛出调用方无法处理的异常，这样就保证了服务调用方的线程不会被长时间或不必要的占用，从而避免了故障在分布式系统中的蔓延乃至雪崩。

如何解决依赖隔离：

1. Hystrix使用命令模式HystrixCommand（command）包装依赖调用逻辑，每个命令在单独线程中/信号授权下执行
2. 可配置依赖调用超时时间，超时时间一般设置为比99.5%平均时间略高即可，当调用超时时，直接返回或执行fallback逻辑
3. 为每个依赖提供一个小的线程池（或信号），如果线程池已满调用将被立即拒绝，默认不采用排队，加速判断时间。
4. 依赖调用结果分：成功，失败（抛出异常），超时，线程拒绝，短路。请求失败（异常，拒绝，超时，短路）时执行fallback(降级逻辑)
5. 提供熔断器组件，可以自动运行或手动调用，停止当前依赖一段时间（10秒），熔断器默认错误率阈值为50%，查过将自动运行。
6. 提供实时依赖的统计和监控
7. 服务降级

所谓降级，一般是从整体负荷考虑。就是当某个服务熔断之后，服务器将不再被调用，此使客户端可以自己准备一个本地的fallback回调，返回一个缺损值，这样做虽然服务水平下降了，但是其它服务能用，总比全部挂掉要好。

1. 服务监控（hystrixDashboard）

除了隔离以来服务的调用以外，Hystrix还提供了准时的调用监控（HystrixDashboard），Hystrix会持续地记录所有通过Hystrix发起的请求执行信息，并以统计报表和图形的形式展示给用户，包括每秒执行多少请求多少成功，多少失败等。Netflix通过hystrix-metrics-event-stream项目实现了对以上指标的监控。SpringCloud也提供了HystrixDashboard的整合对监控内容转化成可视化界面。

1. SpringCloud网关
2. bootstrap配置文件

bootstrap.yml是系统级的，优先级更高

SpringCloud会创建一个 ‘BootstrapContext’，作为Spring应用的，Application Context的父上下文。初始化的时候，‘BootstrapContext’负责从外部源加载配置属性，并解析配置。这两个上下文共享一个从外部获取的‘Environment’.‘Bootstrap’属性有高优先级，默认情况下，他们不会被本地配置覆盖。‘Bootstrap context’和‘Application context’有着不通的约定。所以新增了一个‘bootstrap.yml’,保证‘Bootstrap Context’和‘Application Context’配置的分离

1. 1

1.微服务架构添加从github上读取eureka，客户端，服务端 的配置信息

2.参加上海仪电金槐显示技术有限公司的产品培训

（1）无限投屏技术

通过他们的无限路由器进行连接，他们的路由器支持网线连接与wifi连接，然后手机连接wifi就可以进行屏幕共享

苹果手机，或者安卓系统手机可以进行投屏，苹果手机可以通过自带的镜像投屏将手机屏幕进行投屏，安卓手机需要安装App才能进行投

屏，同时安卓手机拥有更多更全的功能。在连接的过程中通过播放视频等操作进行测试流畅度，相对来说延迟较低，还是比较流程的。

（2）智能物联网讲桌

搭配中控主机，能够实现一键联动开启关闭电脑，能操作投影仪，电动屏幕，以及可定制化的智能化控制。

智能物联网讲桌自带无线麦克风，无线麦克风集合了PPT翻页，音量调节，激光灯 上课的过程中比较实用的功能。

3.信锐科技的 智慧学校物联网解决方案

他们的产品主要是针对教室智能化设备的集中管控，使用了现在比较稳定的LORA作为传输，相对来说信号比较稳定。对所有管控的设备在大

屏幕中进行展示。同时能对用电安全进行监控，也可以远程控制设备的用电与断电