Spring Boot

1、什么是 Spring Boot？

Spring Boot 是 Spring 开源组织下的子项目，是 Spring 组件一站式解决方案，主要是简化了使用 Spring 的难度，简省了繁重的配置，提供了各种启动器，开发者能快速上手。

2、为什么要用 Spring Boot？

Spring Boot 优点非常多，如：

独立运行：内嵌了服务器
简化配置
自动配置 ：注解实现
无代码生成和 XML 配置 用注解去替代了
应用监控
上手容易

3、Spring Boot 的核心配置文件有哪几个？它们的区别是什么？

Spring Boot 的核心配置文件是 application 和 bootstrap 配置文件。

application 配置文件这个容易理解，主要用于 Spring Boot 项目的自动化配置。

bootstrap 配置文件有以下几个应用场景。相对来说用的比较少，具体看场景

使用 Spring Cloud Config 配置中心时，这时需要在 bootstrap 配置文件中添加连接到配置中心的配置属性来加载外部配置中心的配置信息；
一些固定的不能被覆盖的属性；
一些加密/解密的场景；

4、Spring Boot 的配置文件有哪几种格式？它们有什么区别？

.properties 和 .yml，它们的区别主要是书写格式不同。

1).properties
app.user.name = javastack

2).yml
app:
  user:
    name: javastack
另外，.yml 格式不支持 @PropertySource 注解导入配置。

5、Spring Boot 的核心注解是哪个？它主要由哪几个注解组成的？

启动类上面的注解是@SpringBootApplication，它也是 Spring Boot 的核心注解，主要组合包含了以下 3 个注解：
@SpringBootConfiguration：组合了 @Configuration 注解，实现配置文件的功能。
@EnableAutoConfiguration：打开自动配置的功能，也可以关闭某个自动配置的选项，如关闭数据源自动配置功能：@SpringBootApplication(exclude = { DataSourceAutoConfiguration.class })。
@ComponentScan：Spring 组件包名扫描。

6、开启 Spring Boot 特性有哪几种方式？

1）继承 spring-boot-starter-parent 项目

2）导入 spring-boot-dependencies 项目依赖

7、Spring Boot 需要独立的容器运行吗？

可以不需要，内置了 Tomcat/ Jetty 等容器。

8、运行 Spring Boot 有哪几种方式？

1）打包用命令或者放到容器中运行

2）用 Maven/ Gradle 插件运行

3）直接执行 main 方法运行

9、Spring Boot 自动配置原理是什么？

注解 @EnableAutoConfiguration, @Configuration, @ConditionalOnClass 就是自动配置的核心，首先它得是一个配置文件，其次根据类路径下是否有这个类去自动配置。

10、你如何理解 Spring Boot 中的 Starters？

Starters 可以理解为启动器，它包含了一系列可以集成到应用里面的依赖包，你可以一站式集成 Spring 及其他技术，而不需要到处找示例代码和依赖包。如你想使用 Spring JPA 访问数据库，只要加入 spring-boot-starter-data-jpa 启动器依赖就能使用了。

Starters 包含了许多项目中需要用到的依赖，它们能快速持续的运行，都是一系列得到支持的管理传递性依赖。

11、springboot 常用的 starter 有哪些

1.spring-boot-starter-web (嵌入 tomcat 和 web 开发需要 servlet 与 jsp 支持)
2.spring-boot-starter-data-jpa (数据库支持)
3.spring-boot-starter-data-redis (redis 数据库支持)
4.spring-boot-starter-data-solr (solr 搜索应用框架支持)
5.mybatis-spring-boot-starter (第三方的 mybatis 集成 starter)

12、如何在 Spring Boot 启动的时候运行一些特定的代码？

可以实现接口 ApplicationRunner 或者 CommandLineRunner，这两个接口实现方式一样，它们都只提供了一个 run 方法。

13、Spring Boot 有哪几种读取配置的方式？

Spring Boot 可以通过 @PropertySource,@Value,@Environment, @ConfigurationProperties 来绑定变量。

14、Spring Boot 支持哪些日志框架？推荐和默认的日志框架是哪个？

Spring Boot 支持 Java Util Logging, Log4j2, Lockback 作为日志框架，如果你使用 Starters 启动器，Spring Boot 将使用 Logback 作为默认日志框架

15、SpringBoot 实现热部署有哪几种方式？

主要有两种方式：
Spring Loaded
Spring-boot-devtools

16、你如何理解 Spring Boot 配置加载顺序？

在 Spring Boot 里面，可以使用以下几种方式来加载配置。

1）properties 文件；

2）YML 文件；

3）系统环境变量；

4）命令行参数；

17、Spring Boot 如何定义多套不同环境配置？

提供多套配置文件，如：

applcation.properties

application-dev.properties

application-test.properties

application-prod.properties
运行时指定具体的配置文件。

18、Spring Boot 可以兼容老 Spring 项目吗，如何做？

可以兼容，使用 @ImportResource 注解导入老 Spring 项目配置文件。

19、保护 Spring Boot 应用有哪些方法？

在生产中使用 HTTPS
使用 Snyk 检查你的依赖关系
升级到最新版本
启用 CSRF 保护
使用内容安全策略防止 XSS 攻击

20、Spring Boot 2.X 有什么新特性？与 1.X 有什么区别？

配置变更
JDK 版本升级
第三方类库升级
响应式 Spring 编程支持

HTTP/2 支持
配置属性绑定

Spring Cloud

SpringCloud 是基于 SpringBoot 的一套实现微服务的框架。它提供了微服务开发所需的配置管理、服务发现、断路器、智能路由、微代理、控制总线、全局锁、决策竞选、分布式会话和集群状态管理等组件。最重要的是，跟 SpringBoot 框架一起使用的话，会让你开发微服务架构的云服务非常方便。

SpringCloud 五大核心组件：

服务注册发现-Netflix Eureka
配置中心 - spring cloud config
负载均衡-Netflix Ribbon
断路器 - Netflix Hystrix
路由(网关) - Netflix Zuul

1、什么是微服务

以前的模式是所有的代码在同一个工程中 部署在同一个服务器中 同一个项目的不同模块不同功能互相抢占资源

微服务 将工程根据不同的业务规则拆分成微服务 微服务部署在不同的机器上 服务之间进行相互调用

Java 微服务的框架有 dubbo（只能用来做微服务），spring cloud（提供了服务的发现，断路器等）

2、springcloud 如何实现服务的注册和发现

服务在发布时 指定对应的服务名（服务名包括了 IP 地址和端口） 将服务注册到注册中心（eureka 或者 zookeeper）

这一过程是 springcloud 自动实现 只需要在 main 方法添加@EnableDisscoveryClient 同一个服务修改端口就可以启动多个实例

调用方法：传递服务名称通过注册中心获取所有的可用实例 通过负载均衡策略调用（ribbon 和 feign）对应的服务

3、ribbon 和 feign 区别

Ribbon 添加 maven 依赖 spring-starter-ribbon 使用@RibbonClient(value="服务名称") 使用 RestTemplate 调用远程服务对应的方法

feign 添加 maven 依赖 spring-starter-feign 服务提供方提供对外接口 调用方使用 在接口上使用@FeignClient("指定服务名")

Ribbon 和 Feign 的区别：

Ribbon 和 Feign 都是用于调用其他服务的，不过方式不同。

1.启动类使用的注解不同，Ribbon 用的是@RibbonClient，Feign 用的是@EnableFeignClients。

2.服务的指定位置不同，Ribbon 是在@RibbonClient 注解上声明，Feign 则是在定义抽象方法的接口中使用@FeignClient 声明。

3.调用方式不同，Ribbon 需要自己构建 http 请求，模拟 http 请求然后使用 RestTemplate 发送给其他服务，步骤相当繁琐。

Feign 则是在 Ribbon 的基础上进行了一次改进，采用接口的方式，将需要调用的其他服务的方法定义成抽象方法即可，

不需要自己构建 http 请求。不过要注意的是抽象方法的注解、方法签名要和提供服务的方法完全一致。

4、springcloud 断路器的作用

当一个服务调用另一个服务由于网络原因或者自身原因出现问题时 调用者就会等待被调用者的响应 当更多的服务请求到这些资源时 导致更多的请求等待 这样就会发生连锁效应（雪崩效应） 断路器就是解决这一问题

断路器有完全打开状态

一定时间内 达到一定的次数无法调用 并且多次检测没有恢复的迹象 断路器完全打开，那么下次请求就不会请求到该服务

半开 状态

短时间内 有恢复迹象 断路器会将部分请求发给该服务 当能正常调用时 断路器关闭

关闭

当服务一直处于正常状态 能正常调用 断路器关闭

Spring Cloud 底层实现讲解：https://mp.weixin.qq.com/s/r9F8qYw8PIcyjGR2yS0Jzg

5、使用 Spring Cloud 有什么优势？

使用 Spring Boot 开发分布式微服务时，我们面临以下问题

与分布式系统相关的复杂性-这种开销包括网络问题，延迟开销，带宽问题，安全问题。

服务发现-服务发现工具管理群集中的流程和服务如何查找和互相交谈。它涉及一个服务目录，在该目录中注册服务，然后能够查找并连接到该目录中的服务。

冗余-分布式系统中的冗余问题。

负载平衡 --负载平衡改善跨多个计算资源的工作负荷，诸如计算机，计算机集群，网络链路，中央处理单元，或磁盘驱动器的分布。

性能-问题 由于各种运营开销导致的性能问题。

部署复杂性-Devops 技能的要求。

6、服务注册和发现是什么意思？Spring Cloud 如何实现？

当我们开始一个项目时，我们通常在属性文件中进行所有的配置。随着越来越多的服务开发和部署，添加和修改这些属性变得更加复杂。有些服务可能会下降，而某些位置可能会发生变化。手动更改属性可能会产生问题。 Eureka 服务注册和发现可以在这种情况下提供帮助。由于所有服务都在 Eureka 服务器上注册并通过调用 Eureka 服务器完成查找，因此**无需处理服务地点的任何更改和处理。**

7、负载平衡的意义什么？

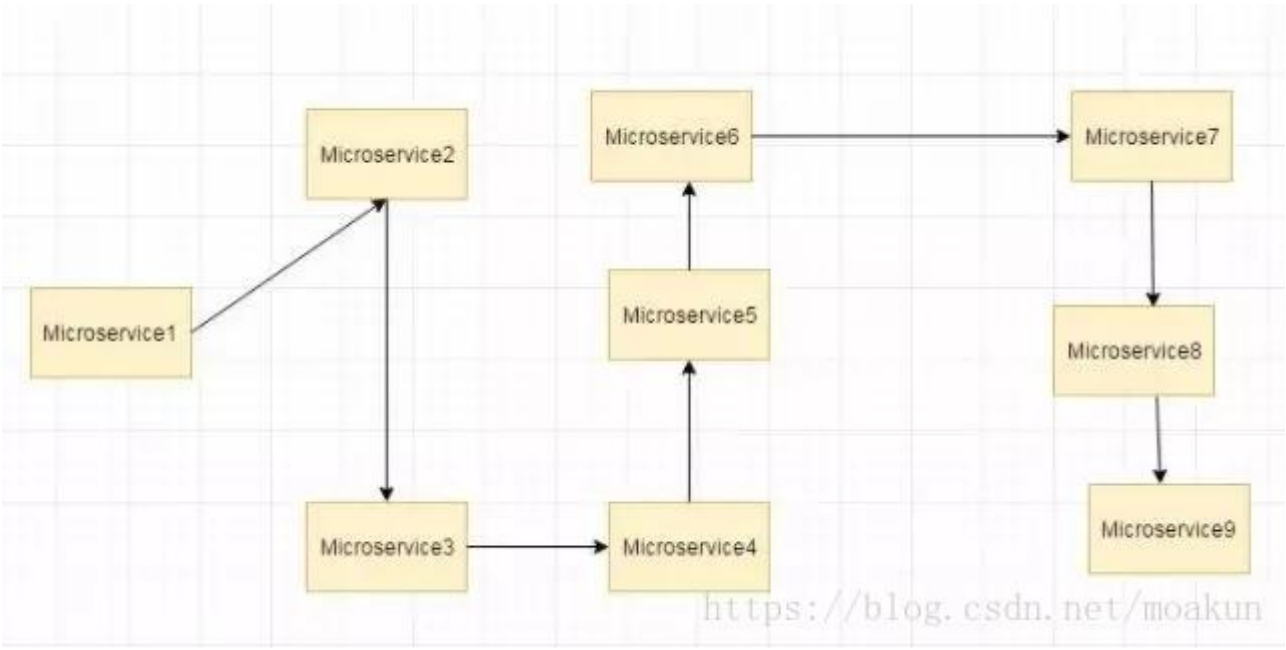
在计算中，负载平衡可以改善跨计算机，计算机集群，网络链接，中央处理单元或磁盘驱动器等多种计算资源的工作负载分布。负载平衡旨在优化资源使用，最大化吞吐量，最小化响应时间并避免任何单一资源的过载。使用多个组件进行负载平衡而不是单个组件可能会通过冗余来提高可靠性和可用性。负载平衡通常涉及专用软件或硬件，例如多层交换机或域名系统服务器进程。

8、什么是 Hystrix？它如何实现容错？

Hystrix 是一个延迟和容错库，旨在隔离远程系统，服务和第三方库的访问点，当出现故障是不可避免的故障时，停止级联故障并在复杂的分布式系统中实现弹性。

通常对于使用微服务架构开发的系统，涉及到许多微服务。这些微服务彼此协作。

思考以下微服务



假设如果上图中的微服务 9 失败了，那么使用传统方法我们将传播一个异常。但这仍然会导致整个系统崩溃。

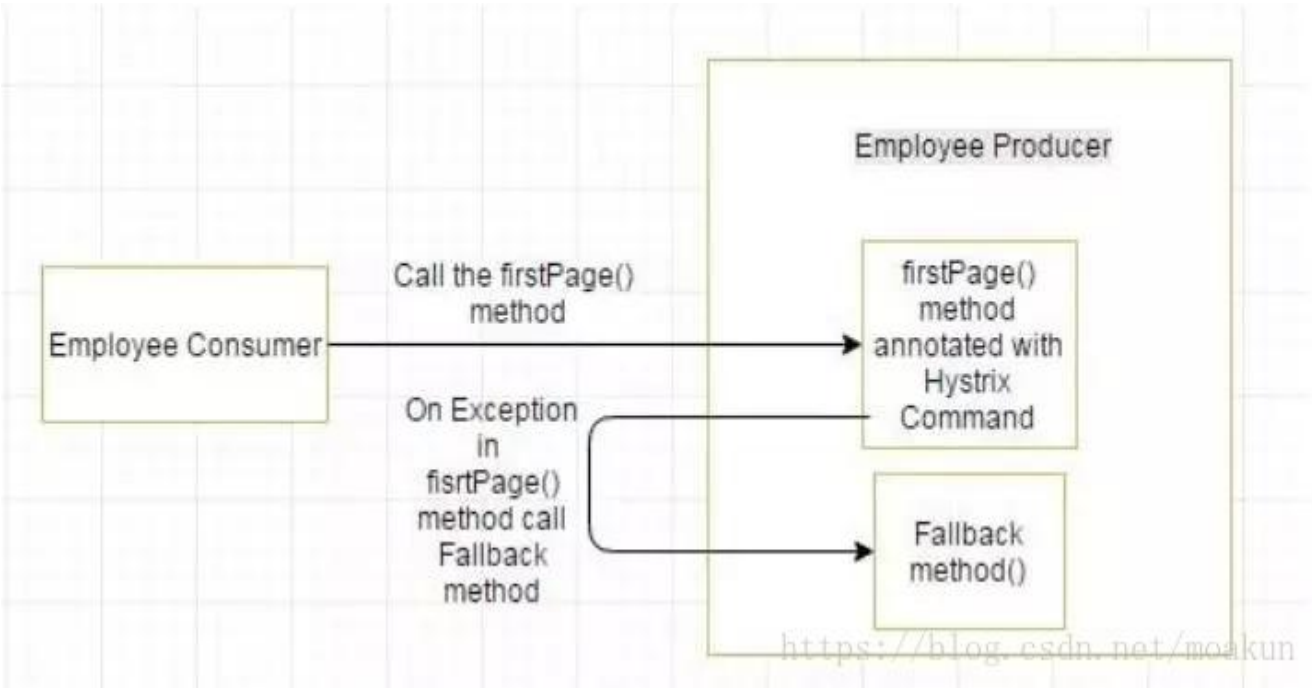
随着微服务数量的增加,这个问题变得更加复杂。微服务的数量可以高达 1000.这是 hystrix 出现的地方 我们将使用 Hystrix 在这种情况下的 Fallback 方法功能。我们有两个服务 employee-consumer 使用由 employee-consumer 公开的服务。

简化图如下所示

现在假设由于某种原因，employee-producer 公开的服务会抛出异常。我们在这种情况下使用 Hystrix 定义了一个回退方法。这种后备方法应该具有与公开服务相同的返回类型。如果暴露服务中出现异常，则回退方法将返回一些值。

9、什么是 Hystrix 断路器？我们需要它吗？

由于某些原因，employee-consumer 公开服务会引发异常。在这种情况下使用 Hystrix 我们定义了一个回退方法。如果在公开服务中发生异常，则回退方法返回一些默认值。



如果 firstPage method() 中的异常继续发生，则 Hystrix 电路将中断，并且员工使用者将一起跳过 firtsPage 方法，并直接调用回退方法。 断路器的目的是给第一页方法或第一页方法可能调用的其他方法留出时间，并导致异常恢复。可能发生的情况是，在负载较小的情况下，导致异常的问题有更好的恢复机会 。

10、什么是 Netflix Feign？它的优点是什么？

Feign 是受到 Retrofit，JAXRS-2.0 和 WebSocket 启发的 java 客户端联编程序。Feign 的第一个目标是将约束分母的复杂性统一到 http apis，而不考虑其稳定性。在 employee-consumer 的例子中，我们使用了 employee-producer 使用 REST 模板公开的 REST 服务。

但是我们必须编写大量代码才能执行以下步骤

使用功能区进行负载平衡。

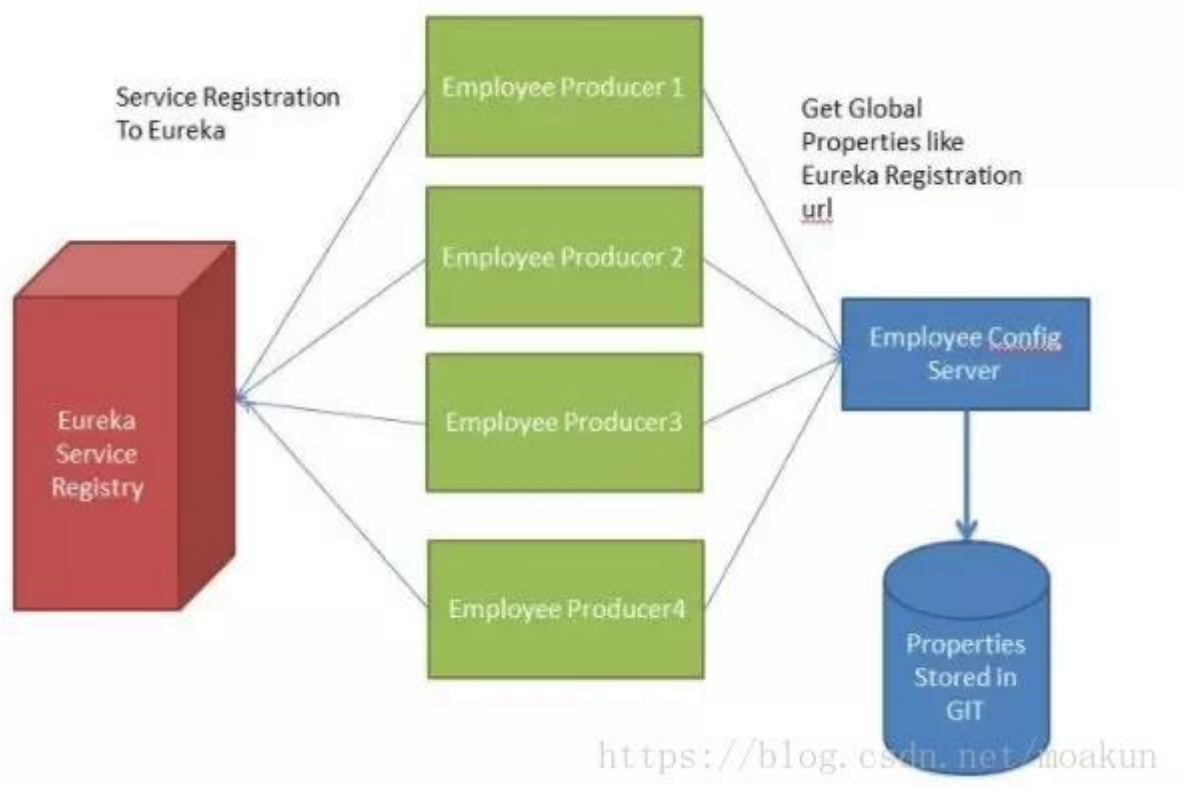
获取服务实例，然后获取基本 URL。

利用 REST 模板来使用服务。

什么是 Spring Cloud Bus？我们需要它吗？

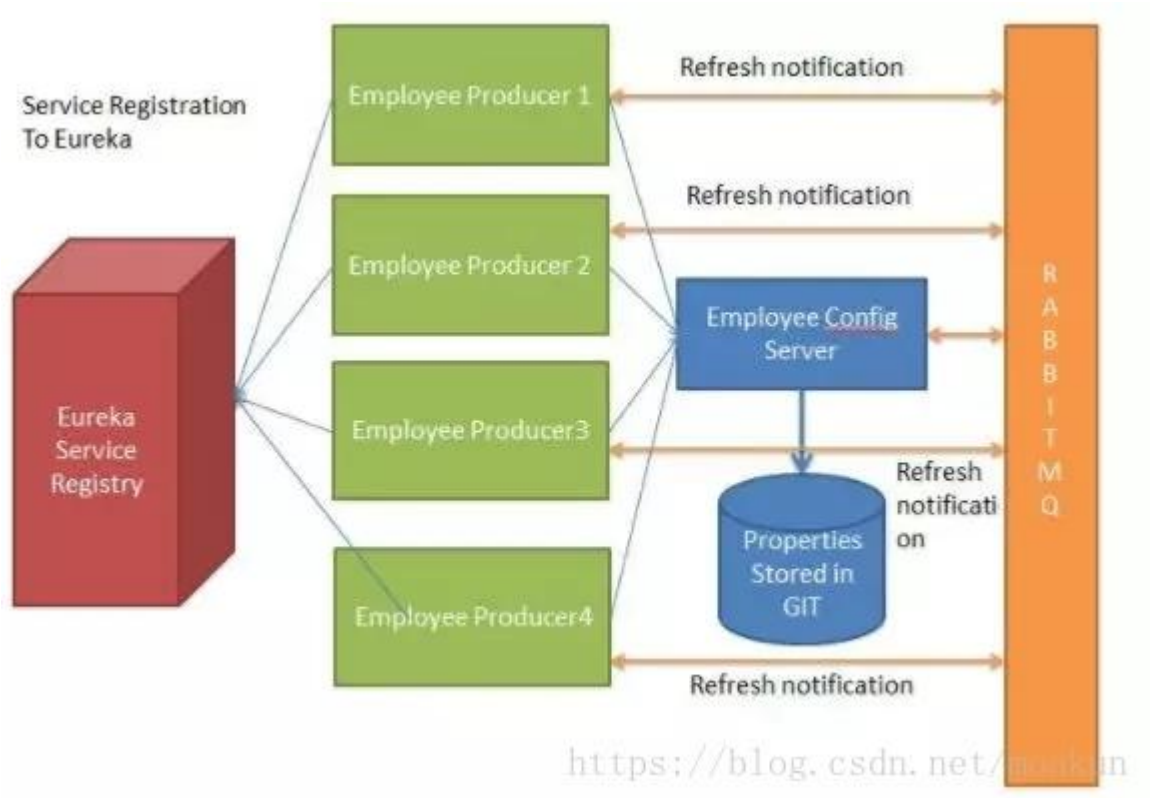
考虑以下情况：我们有多个应用程序使用 Spring Cloud Config 读取属性，而 Spring Cloud Config 从 GIT 读取这些属性。

下面的例子中多个员工生产者模块从 Employee Config Module 获取 Eureka 注册的财产。



如果假设 GIT 中的 Eureka 注册属性更改为指向另一台 Eureka 服务器，会发生什么情况。在这种情况下，我们将不得不重新启动服务以获取更新的属性。

还有另一种使用执行器端点/刷新的方式。但是我们将不得不为每个模块单独调用这个 url。例如，如果 Employee Producer1 部署在端口 8080 上，则调用 http：// localhost：8080 / refresh。同样对于 Employee Producer2 http：// localhost：8081 / refresh 等等。这又很麻烦。这就是 Spring Cloud Bus 发挥作用的地方。



Spring Cloud Bus 提供了跨多个实例刷新配置的功能。因此，在上面的示例中，如果我们刷新 Employee Producer1，则会自动刷新所有其他必需的模块。如果我们有多个微服务启动并运行，这特别有用。这是通过将所有微服务连接到单个消息代理来实现的。无论何时刷新实例，此事件都会订阅到侦听此代理的所有微服务，并且它们也会刷新。可以通过使用端点/总线/刷新来实现对任何单个实例的刷新。

11.SpringCloud 和 Dubbo

SpringCloud 和 Dubbo 都是现在主流的微服务架构

SpringCloud 是 Apache 旗下的 Spring 体系下的微服务解决方案

Dubbo 是阿里系的分布式服务治理框架

从技术维度上,其实 SpringCloud 远远的超过 Dubbo,Dubbo 本身只是实现了服务治理,而 SpringCloud 现在以及有 21 个子项目以后还会更多

所以其实很多人都会说 Dubbo 和 SpringCloud 是不公平的

但是由于 RPC 以及注册中心元数据等原因,在技术选型的时候我们只能二者选其一,所以我们常常为用他俩来对比

服务的调用方式 Dubbo 使用的是 RPC 远程调用,而 SpringCloud 使用的是 Rest API,其实更符合微服务官方的定义

服务的注册中心来看,Dubbo 使用了第三方的 ZooKeeper 作为其底层的注册中心,实现服务的注册和发现,SpringCloud 使用 Spring Cloud Netflix Eureka 实现注册中心,当然 SpringCloud 也可以使用 ZooKeeper 实现,但一般我们不会这样做

服务网关,Dubbo 并没有本身的实现,只能通过其他第三方技术的整合,而 SpringCloud 有 Zuul 路由网关,作为路由服务器,进行消费者的请求分发,SpringCloud 还支持断路器,与 git 完美集成分布式配置文件支持版本控制,事务总线实现配置文件的更新与服务自动装配等等一系列的微服务架构要素

2.技术选型

目前国内的分布式系统选型主要还是 Dubbo 毕竟国产,而且国内工程师的技术熟练程度高,并且 Dubbo 在其他维度上的缺陷可以由其他第三方框架进行集成进行弥补

而 SpringCloud 目前是国外比较流行,当然我觉得国内的市场也会慢慢的偏向 SpringCloud,就连刘军作为Dubbo 重启的负责人也发表过观点,Dubbo 的发展方向是积极适应 SpringCloud 生态,并不是起冲突

3.Rest 和 RPC 对比

其实如果仔细阅读过微服务提出者马丁福勒的论文的话可以发现其定义的服务间通信机制就是 Http Rest

RPC 最主要的缺陷就是服务提供方和调用方式之间依赖太强,我们需要为每一个微服务进行接口的定义,并通过持续继承发布,需要严格的版本控制才不会出现服务提供和调用之间因为版本不同而产生的冲突

而 REST 是轻量级的接口,服务的提供和调用不存在代码之间的耦合,只是通过一个约定进行规范,但也有可能出现文档和接口不一致而导致的服务集成问题,但可以通过 swagger 工具整合,是代码和文档一体化解决,所以 REST 在分布式环境下比 RPC 更加灵活

这也是为什么当当网的 DubboX 在对 Dubbo 的增强中增加了对 REST 的支持的原因

4.文档质量和社区活跃度

SpringCloud 社区活跃度远高于 Dubbo,毕竟由于梁飞团队的原因导致 Dubbo 停止更新迭代五年,而中小型公司无法承担技术开发的成本导致 Dubbo 社区严重低落,而 SpringCloud 异军突起,迅速占领了微服务的市场,背靠 Spring 混的风生水起

Dubbo 经过多年的积累文档相当成熟,对于微服务的架构体系各个公司也有稳定的现状

12.SpringBoot 和 SpringCloud

SpringBoot 是 Spring 推出用于解决传统框架配置文件冗余,装配组件繁杂的基于 Maven 的解决方案,旨在快速搭建单个微服务

而 SpringCloud 专注于解决各个微服务之间的协调与配置,服务之间的通信,熔断,负载均衡等

技术维度并相同,并且 SpringCloud 是依赖于 SpringBoot 的,而 SpringBoot 并不是依赖与 SpringCloud,甚至还可以和 Dubbo 进行优秀的整合开发

总结:

SpringBoot 专注于快速方便的开发单个个体的微服务

SpringCloud 是关注全局的微服务协调整理治理框架,整合并管理各个微服务,为各个微服务之间提供,配置管理,服务发现,断路器,路由,事件总线等集成服务

SpringBoot 不依赖于 SpringCloud,SpringCloud 依赖于 SpringBoot,属于依赖关系

SpringBoot 专注于快速,方便的开发单个的微服务个体,SpringCloud 关注全局的服务治理框架

13.Eureka 和 ZooKeeper 都可以提供服务注册与发现的功能,请说说两个的区别

1.ZooKeeper 保证的是 CP,Eureka 保证的是 AP

ZooKeeper 在选举期间注册服务瘫痪,虽然服务最终会恢复,但是选举期间不可用的.

Eureka 各个节点是平等关系,只要有一台 Eureka 就可以保证服务可用,而查询到的数据并不是最新的,

自我保护机制会导致 Eureka 不再从注册列表移除因长时间没收到心跳而应该过期的服务

Eureka 仍然能够接受新服务的注册和查询请求,但是不会被同步到其他节点(高可用)

当网络稳定时,当前实例新的注册信息会被同步到其他节点中(最终一致性)

Eureka 可以很好的应对因网络故障导致部分节点失去联系的情况,而不会像 ZooKeeper 一样使得整个注册系统瘫痪

2.ZooKeeper 有 Leader 和 Follower 角色,Eureka 各个节点平等

3.ZooKeeper 采用过半数存活原则,Eureka 采用自我保护机制解决分区问题

4.Eureka 本质上是一个工程,而 ZooKeeper 只是一个进程

14.微服务之间是如何独立通讯的

1、远程过程调用（Remote Procedure Invocation）

也就是我们常说的服务的注册与发现

直接通过远程过程调用来访问别的 service。

优点：

简单，常见,因为没有中间件代理，系统更简单

缺点：

只支持请求/响应的模式，不支持别的，比如通知、请求/异步响应、发布/订阅、发布/异步响应

降低了可用性，因为客户端和服务端在请求过程中必须都是可用的

2、消息

使用异步消息来做服务间通信。服务间通过消息管道来交换消息，从而通信。

优点:

把客户端和服务端解耦，更松耦合

提高可用性，因为消息中间件缓存了消息，直到消费者可以消费

支持很多通信机制比如通知、请求/异步响应、发布/订阅、发布/异步响应

缺点:

消息中间件有额外的复杂性

15.什么是服务熔断?什么是服务降级

在复杂的分布式系统中,微服务之间的相互调用,有可能出现各种各样的原因导致服务的阻塞,在高并发场景下,服务的阻塞意味着线程的阻塞,导致当前线程不可用,服务器的线程全部阻塞,导致服务器崩溃,由于服务之间的调用关系是同步的,会对整个微服务系统造成服务雪崩

为了解决某个微服务的调用响应时间过长或者不可用进而占用越来越多的系统资源引起雪崩效应就需要进行服务熔断和服务降级处理。

所谓的服务熔断指的是某个服务故障或异常一起类似显示世界中的“保险丝”当某个异常条件被触发就直接熔断整个服务，而不是一直等到此服务超时。

服务熔断就是相当于我们电闸的保险丝,一旦发生服务雪崩的,就会熔断整个服务,通过维护一个自己的线程池,当线程达到阈值的时候就启动服务降级,如果其他请求继续访问就直接返回 fallback 的默认值

15.微服务的优缺点分别是什么?说下你在项目开发中碰到的坑

优点

- 每一个服务足够内聚,代码容易理解
- 开发效率提高,一个服务只做一件事
- 微服务能够被小团队单独开发
- 微服务是松耦合的,是有功能意义的服务
- 可以用不同的语言开发,面向接口编程
- 易于与第三方集成
- 微服务只是业务逻辑的代码,不会和 HTML,CSS 或者其他界面组合
    - 开发中,两种开发模式
        - 前后端分离
        - 全栈工程师
- 可以灵活搭配,连接公共库/连接独立库

缺点

- 分布式系统的复杂性
- 多服务运维难度,随着服务的增加,运维的压力也在增大
- 系统部署依赖
- 服务间通信成本
- 数据一致性
- 系统集成测试
- 性能监控

16.你所知道的微服务技术栈有哪些?请列举一二

多种技术的集合体

我们在讨论一个分布式的微服务架构的话,需要哪些维度

维度(SpringCloud)

服务开发

- SpringBoot
- Spring
- SpringMVC

服务配置与管理

- Netfilx 公司的 Archaiusm,阿里的 Diamond

服务注册与发现

- Eureka,ZooKeeper

服务调用

- Rest,RPC,gRPC

服务熔断器

- Hystrix

服务负载均衡

- Ribbon,Nginx

服务接口调用

- Feign

消息队列

- Kafka,RabbitMq,ActiveMq

服务配置中心管理

- SpringCloudConfing

服务路由(API 网关)

- Zuul

事件消息总线

SpringCloud Bus