微信搜一搜 搜云库技术团队

第二版：SpringCloud 22 道

目录

公众号：架构师专栏

微信搜一搜 搜云库技术团队

我们的网站：https://tech.souyunku.com

关注我们的公众号：搜云库技术团队，回复以下关键字

回复:【进群】邀请您进「技术架构分享群」

回复:【内推】即可进：北京，上海，广周，深圳，杭州，成都，武汉，南京，郑州，西安，长沙「程序员工作内推群」

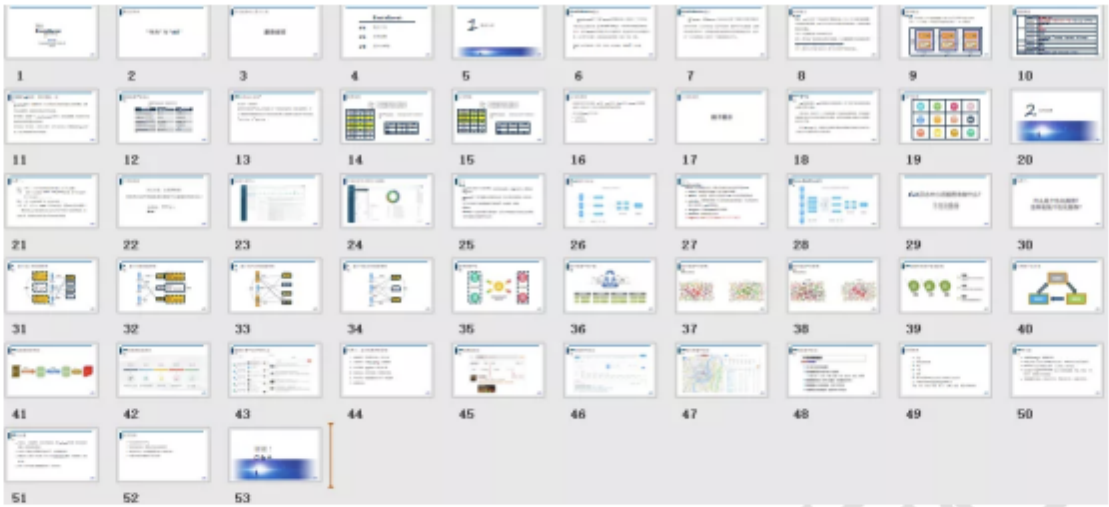
回复 【1024】送 4000G 最新架构师视频

回复 【PPT】即可无套路获取，以下最新整理调优 PPT!

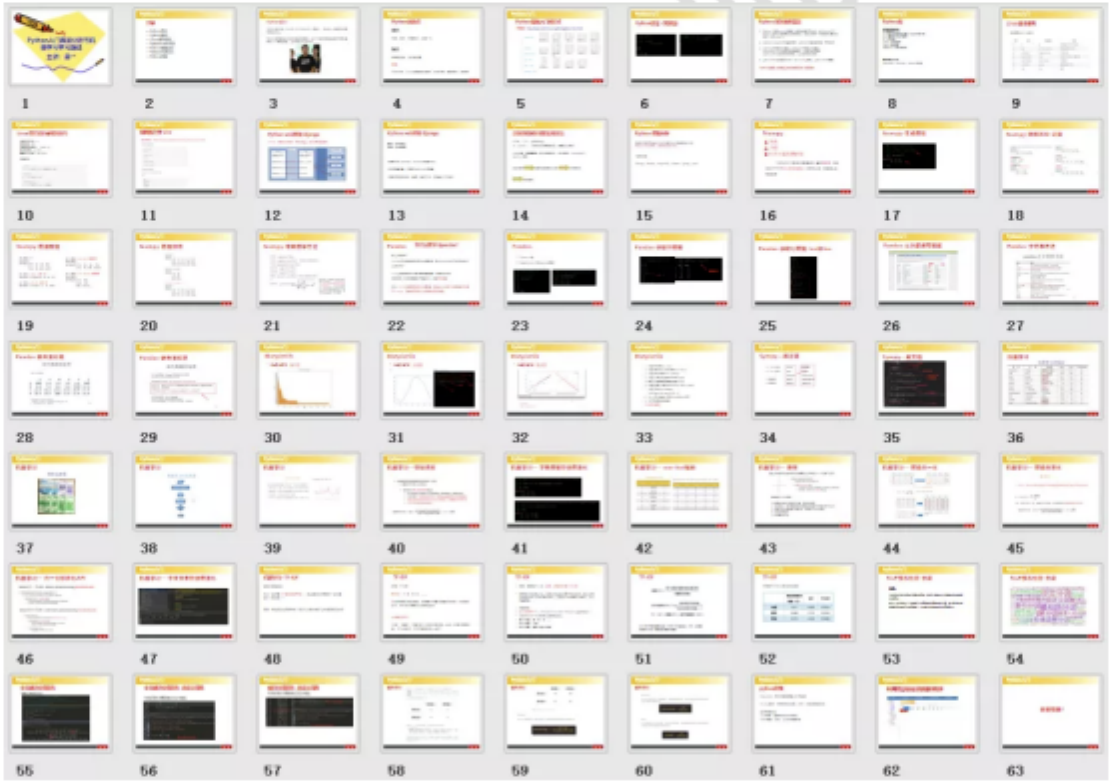
46 页《JVM 深度调优，演讲 PPT》

53 页《Elasticsearch 调优演讲 PPT》

微信搜一搜 搜云库技术团队



63 页《Python 数据分析入门 PPT》



微信搜一搜 搜云库技术团队

微信扫一扫
https://tech.souyunku.com
技术、架构、资料、工作 、内推
专注于分享最有价值的互联网技术干货文章

什么是 Spring Cloud?

spring cloud 是一系列框架的有序集合。它利用 spring boot 的开发便利性巧妙地简化了分布式系统基础设施的开发，如服务发现注册、配置中心、消息总线、负载均衡、断路器、数据监控等，都可以用 spring boot 的开发风格做到一键启动和部署。

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。

微信搜一搜 搜云库技术团队

- 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。

Spring Cloud 实现服务注册和发现的原理是什么?

- 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 Spring Cloud 自动实现的，只需要在 main 方法添加 @EnableDisscoveryClient 即可，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。

为什么要使用 Spring Cloud 熔断器?

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，断路器就是解决这一问题的。

断路器的状态有以下几种：

- 完全打开：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，那么下次的请求不会请求到该服务。
- 半开：短时间内有恢复迹象，断路器会将部分请求发送给服务，当能正常调用时，断路器关闭。
- 关闭：服务一直处于正常状态，能正常调用，断路器关闭。

微信搜一搜 搜云库技术团队

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。

spring cloud 和 dubbo 区别？

- 服务调用方式 dubbo 是 RPC springcloud Rest Api
- 注册中心,dubbo 是 zookeeper springcloud 是 eureka，也可以是 zookeeper
- 服务网关,dubbo 本身没有实现，只能通过其他第三方技术整合，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 支持断路器，与 git 完美集成配置文件支持版本控制，事物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。

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：

微信搜一搜 搜云库技术团队

- 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

负载均衡的意义是什么?

在计算中，负载均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。负载均衡旨在优化资源使用，最大吞吐量，最小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均

公众号·架构师专栏

微信搜一搜 搜云库技术团队

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进程。

微服务之间是如何独立通讯的?

1、远程调用，比如 feign 调用，直接通过远程过程调用来访问别的 service。 2.消息中间件

springcloud 如何实现服务的注册?

1、服务发布时，指定对应的服务名,将服务注册到 注册中心(eureka zookeeper)

2、注册中心加@EnableEurekaServer,服务用@EnableDiscoveryClient，然后用 ribbon 或 feign 进行服务直接的调用发现。

spring cloud 断路器的作用是什么?

在分布式架构中，断路器模式的作用也是类似的，当某个服务单元发生故障（类似用电器发生短路）之后，通过断路器的故障监控（类似熔断保险丝），向调用方返回一个错误响应，而不是长时间的等待。这样就不会使得线程因调用故障服务被长时间占用不释放，避免了故障在分布式系统中的蔓延。

什么是 Hystrix?

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，

微信搜一搜 搜云库技术团队

随着微服务数量的增加，这个问题变得更加复杂。我们将使用 Hystrix 的 Fallback 方法来处理，假设由于某种原因，公开的服务接口抛出异常，我们在这种情况下使用 Hystrix 定义一个回退方法。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，回退方法将返回对应信息。

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

- 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。
- ZooKeeper 有 Leader 和 Follower 角色,Eureka 各个节点平等
- 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
- Eureka 本质上是一个工程,而 ZooKeeper 只是一个进程

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。

微信搜一搜 搜云库技术团队

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。前面的代码如下

```
@Controller
public class ConsumerControllerClient {

@Autowired
private LoadBalancerClient loadBalancer;

public void getEmployee() throws RestClientException, IOException {

    ServiceInstance
serviceInstance=loadBalancer.choose("employee-producer");

    System.out.println(serviceInstance.getUri());

    String baseUrl=serviceInstance.getUri().toString();

    baseUrl=baseUrl+"/employee";

    RestTemplate restTemplate = new RestTemplate();
    ResponseEntity<String> response=null;
    try{
    response=restTemplate.exchange(baseUrl,
            HttpMethod.GET, getHeaders(),String.class);
    }catch (Exception ex)
    {
```

微信搜一搜 搜云库技术团队

```
            System.out.println(ex);
        }
        System.out.println(response.getBody());
    }
```

之前的代码，有像 NullPointer 这样的例外的机会，并不是最优的。我们将看到如何使用 Netflix Feign 使呼叫变得更加轻松和清洁。如果 Netflix Ribbon 依赖关系也在类路径中，那么 Feign 默认也会负责负载平衡。

REST 和 RPC 对比

1、RPC 主要的缺陷是服务提供方和调用方式之间的依赖太强，需要对每一个微服务进行接口的定义，并通过持续继承发布，严格版本控制才不会出现冲突。

2、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，只需要一个约定进行规范。

什么是 feigin？它的优点是什么？

- feign 采用的是基于接口的注解
- feign 整合了 ribbon，具有负载均衡的能力
- 整合了 Hystrix，具有熔断的能力

使用:

1、添加 pom 依赖。

2、启动类添加@EnableFeignClients

微信搜一搜 搜云库技术团队

3、定义一个接口@FeignClient(name=“xxx”)指定调用哪个服务

Ribbon 和 Feign 的区别?

- Ribbon 都是调用其他服务的，但方式不同。
- 启动类注解不同，Ribbon 是@RibbonClient feign 的是@EnableFeignClients
- 服务指定的位置不同，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。
- 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 需要将调用的方法定义成抽象方法即可。

什么是 Spring Cloud Bus?

spring cloud bus 将分布式的节点用轻量的消息代理连接起来，它可以用于广播配置文件的更改或者服务直接的通讯，也可用于监控。

如果修改了配置文件，发送一次请求，所有的客户端便会重新读取配置文件。

使用:

1、添加依赖

2、配置 rabbimq

公众号·架构师老样

微信搜一搜 搜云库技术团队

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?

zookeeper 是 CP 原则，强一致性和分区容错性。

eureka 是 AP 原则 可用性和分区容错性。

zookeeper 当主节点故障时，zk 会在剩余节点重新选择主节点，耗时过长，虽然最终能够恢复，但是选取主节点期间会导致服务不可用，这是不能容忍的。

eureka 各个节点是平等的，一个节点挂掉，其他节点仍会正常保证服务。

你所知道微服务的技术栈有哪些？列举一二

微服务条目	落地技术
服务开发	SpringBoot、Spring、SpringMVC
服务配置与管理	Netfix公司的Archaius、阿里的Diamond等
服务注册与发现	Eurka、Consul、Zookeeper等
服务调用	Rest（服务通信）、RPC（Dubbo）、GRpc
服务熔断器	Hystrix、Envoy等
负载均衡	Nginx、Ribbon等
服务接口调用（客户端简化工具）	Fegin等
消息队列	Kafka、RabbitMQ、ActiveMQ等
服务配置中心管理	SpringCloudConfig、Chef等
服务路由（API网关）	Zuul等
服务监控	Zabbix，Nagios，Metrics，Spectator等
全链路追踪	Zipkin，Brave，Dapper等
服务部署	Docker，OpenStack，Kubernetes等
数据流操作开发包	SpringCloud Stream（封装与Redis，Rabbit，kafka等发送接收消息）
事件消息总线	Spring Cloud Bus

架构师专栏

微信搜一搜 搜云库技术团队

服务网关的作用

- 简化客户端调用复杂度，统一处理外部请求。
- 数据裁剪以及聚合，根据不同的接口需求，对数据加工后对外。
- 多渠道支持，针对不同的客户端提供不同的网关支持。
- 遗留系统的微服务化改造，可以作为新老系统的中转组件。
- 统一处理调用过程中的安全、权限问题。

链路跟踪 Sleuth

当我们项目中引入 Spring Cloud Sleuth 后，每次链路请求都会添加一串追踪信息，格式是[server-name, main-traceId,sub-spanId,boolean]：

- server-name：服务结点名称。
- main-traceId：一条链路唯一的 ID，为 TraceID。
- sub-spanId：链路中每一环的 ID，为 SpanID。
- boolean：是否将信息输出到 Zipkin 等服务收集和展示。

Sleuth 的实现是基于 HTTP 的，为了在数据的收集过程中不能影响到正常业务，Sleuth 会在每个请求的 Header 上添加跟踪需求的重要信息。这样在数据收集时，只需要将 Header 上的相关信息发送给对应的图像工具即可，图像工具根据上传的数据，按照 Span 对应的逻辑进行分析、展示。

好了各位， 本文到这里就结束了！ 如果本文有任何错误，请批评指教，不胜感激 ！

公众号：架构师专栏

微信搜一搜 搜云库技术团队

我是提莫！ 一个节操泛滥，一身凛然正气，刚正不阿的 Java 程序员

公众号：架构师书栏