内容提要

- 网络I/O模型
- 基于服务治理的RPC解决方案
- 消息机制
- 分布式数据存储
- 互联网分布式架构设计与开发

现有的RPC应用相关的方案

- 流行的RPC框架：Thrift、Grpc、Dubbo、Srpring
- 服务资源注册与协调zookeeper、Eureka

rpc需解决的问题

- 远程通讯与调用机制
- 服务注册与发现
- 强大的服务治理功能

Apache Thrift

- 最初由facebook开发用做系统内各语言之间的RPC框架
- 2007年由facebook贡献到apache基金
- 08年5月进入apache孵化器

Thrift特点

- 跨语言，支持的语言多（C++, Java, Python, PHP, Ruby, Erlang, Perl, Haskell, C#, Cocoa, Smalltalk等多种语言）
- 支持多种信息格式：Thrift私有的二进制编码规则、LVQ消息格式，还有常规的JSON格式
- 支持阻塞式IO模型和多路IO复用模型
- 并发性能高

Thrift IDL

- IDL（接口定义语言）并不是RPC实现中所必须的。但是需要跨语言的RPC框架一定会有IDL部分的存在。
- 为了支持多种语言，Apache Thrift有一套自己的接口定义语言，并且通过Apache Thrift的代码生成程序，能够生成各种编程语言的代码。

- IDL语法：http://thrift.apache.org/docs/idl

Thrift的不足-缺乏服务治理

当业务发生变化后，要重新编写IDL，重新生成接口代码，重新部署的工作量会很大

为保证生产环境的服务的可用性，可能会出现一部分接口是新部署的，另外一部分接口是还未更新的。接口的稳定得不到保证

假如生产环境下一共有20个相对独立运行的系统：计费系统、客户系统、订单系统、库存系统、物流系统、税务联动系统，等等。负责他们的开发团队都是不一样的。如何在某个系统的接口发生变动后，通知到其它系统？如何做到之前的接口也一样可以使用呢？

问题引入

- 服务的治理

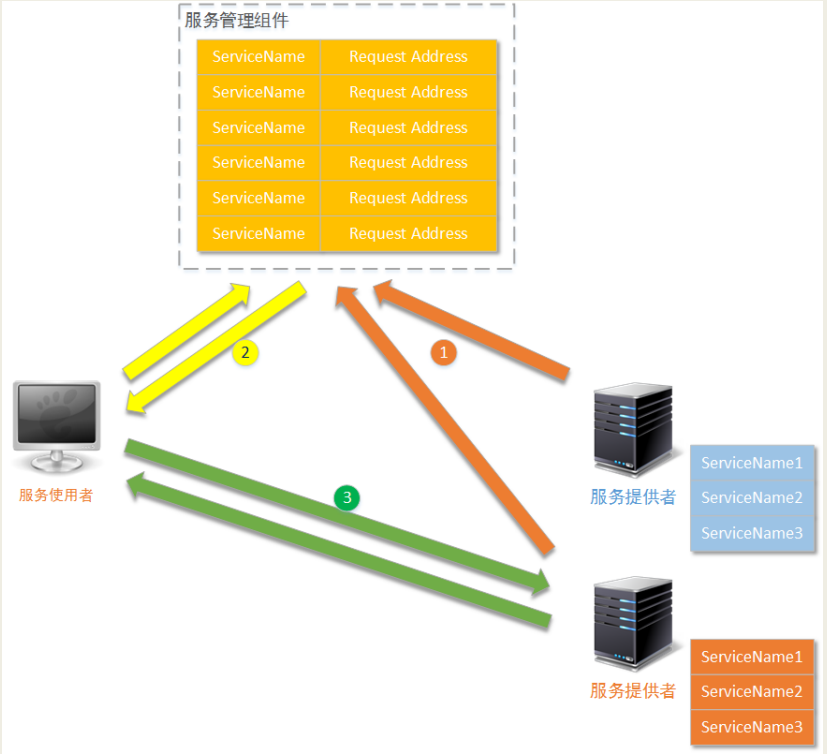
—不仅仅是服务的注册和发现

状态追踪
调用关系维护
安全权限
版本控制
集群路由和负载

。。。。。。

服务的治理

1.当服务提供者能够向外部系统提供调用服务时，它会首先向“服务管理组件”注册这个服务，包括服务名、访问权限、优先级、版本、参数、真实访路径、有效时间等等基本信息。
2.当某一个服务使用者需要调用服务时，首先会向“服务管理组件”询问服务的基本信息。当然“服务管理组件”还会验证服务使用者是否有权限进行调用、是否符合调用的前置条件等等过滤。最终“服务管理组件”将真实的服务提供者所在位置返回给服务使用者。
3.服务使用者拿到真实服务提供者的基本信息、调用权限后，再向真实的服务提供者发出调用请求，进行正式的业务调用过程。



zookeeper

- ZooKeeper是一个分布式的，开放源码的分布式应用程序协调服务，是Google的Chubby一个开源的实现，是Hadoop的重要组件，CDH版本中更是使用它进行Namenode的协调控制。
- 目标：封装好复杂易出错的关键服务，将简单易用的接口和性能高效、功能稳定的系统提供给用户。

Zookeeper主要功能

- 管理系统中独特的/统一的信息
- 集群状态监控和通知
- 协调资源抢占（锁）
- 分派计算任务

Zookeeper的应用

- 大数据分布式计算平台
- 用于Rpc的服务注册与发现
- 数据发布/订阅、负载均衡、命名服务、分布式协调/通知、集群管理、Master 选举、配置维护，名字服务、分布式同步、分布式锁和分布式队列。。。。

一个完整的基于服务的分布式架构实现方案

- Thrift+zookeeper
- 缺点：服务治理机制的实现比较复杂，如：访问权限、版本控制、服务时效控制、次数控制、性能措施等各类细节问题。
- 优点：跨语言跨平台

国内流行的RPC服务框架：DUBBO

- Dubbo是阿里巴巴SOA服务化治理方案的核心框架，每天为2,000+个服务提供3,000,000,000+次访问量支持，并被广泛应用于阿里巴巴集团的各成员站点。

Dubbo功能

- 透明化的远程方法调用，就像调用本地方法一样调用远程方法，只需简单配置，没有任何API侵入。
- 软负载均衡及容错机制，可在内网替代硬件负载均衡器，降低成本，减少单点。
- 服务自动注册与发现，不再需要写死服务提供方地址，注册中心基于接口名查询服务提供者的IP地址，并且能够平滑添加或删除服务提供者。
- 其网络通信模型采用Netty
- **其服务注册中心采用的ZK**

Spring Cloud

- Spring Cloud是一系列框架的集合，其基于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，为我们提供一整套企业级分布式云应用的完美解决方案。
  - *服务治理(发现注册)*
  - *配置中心*
  - *消息总线*
  - *负载均衡*
  - *断路器*
  - *数据监控*
  - *分布式会话和集群状态管理等功能*

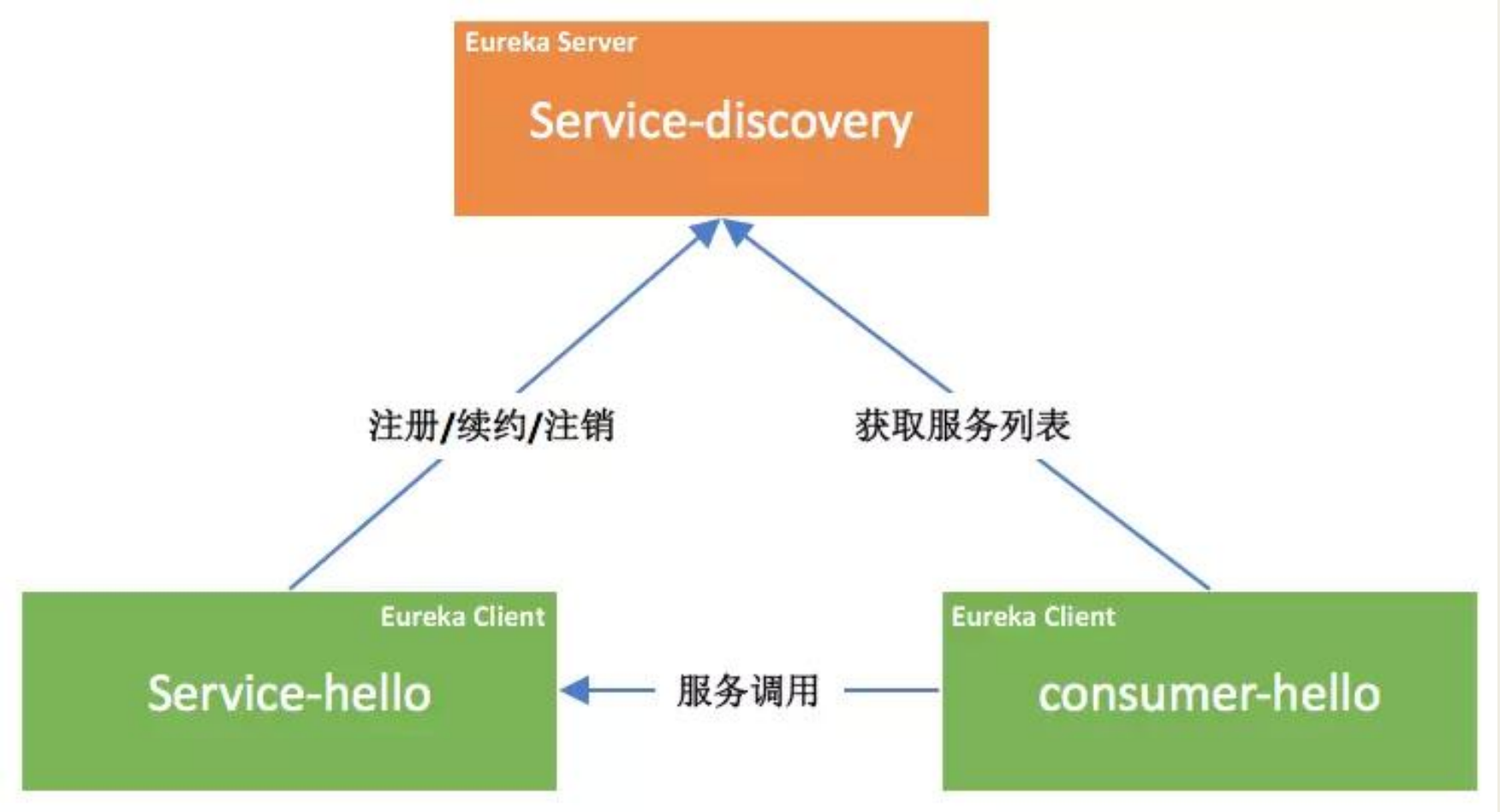
Spring Cloud子项目

- Spring Cloud包含了多个子项目：Spring Cloud Config、Spring Cloud Netflix、Spring Cloud CloudFoundry、Spring Cloud AWS、Spring Cloud Security、Spring Cloud Commons、Spring Cloud Zookeeper、Spring Cloud CLI等。
- 这些项目是Spring将目前各家公司开发的比较成熟、经得起实际考验的服务框架组合起来，通过Spring Boot风格进行再封装屏蔽掉了复杂的配置和实现原理，最终给我们开发者留出了一套简单易懂、易部署和易维护的分布式系统开发工具包。

Eureka

- Spring Cloud的核心是服务治理
- 服务治理主要通过整合Netflix的相关产品来实现这方面的功能，也就是Spring Cloud Netflix
- 在该项目中包括用于服务注册和发现的Eureka

Eureka服务注册管理



两大服务发现与注册实现方案

- ZK
- Eureka

基于服务的分布式系统开发

- 自己定制
- 纯粹的JAVA：RMI
- Thrift+zookeeper:高效快速，跨语言平台
- Dubbo+zookeeper：服务治理功能强大
- Spring 解决方案(含Eurika服务注册中心):Spring庞大生态圈,功能非常完善