

Spring Cloud Alibaba

Spring Cloud + Nacos

Spring Cloud + Netflix (Netflix)

Spring Cloud + Alibaba (阿里公司)

Alibaba 开源生态

- Dubbo (RPC框架 - Apache)
- Nacos (服务注册/配置中心)
- Sentinel (限流/熔断)
- Seata (分布式事务解决方案)
- RocketMQ (分布式消息中间件 - Apache) 、 Kafka、 RabbitMQ、 Pulsar

未开始讲解的部分

- Nacos作为注册中心
- Sentinel
- Seata

Apache Dubbo

Dubbo是阿里巴巴内部使用的一个分布式服务治理框架，2012年开源，因为Dubbo在公司内部经过了很多的验证相对来说比较成熟，所以在很短的时间里就被很多互联网公司使用，再加上阿里出来的很多技术大牛进入各个创业公司担任技术架构以后，都以Dubbo作为主推的RPC框架使得dubbo很快成为了很多互联网公司的首要选择。

并且很多公司在应用dubbo时，会基于自身业务特性进行优化和改进，所以也衍生了很多版本，比如京东的JSF、比如新浪的Motan、比如当当的dubbox.

在2014年10月份，Dubbo停止了维护。后来在2017年的9月份，阿里宣布重启Dubbo，并且对于Dubbo做好了长期投入的准备，并且在这段时间Dubbo进行了非常多的更新，目前的版本已经到了2.7.

2018年1月8日，Dubbo创始人之一梁飞在Dubbo交流群里透露了Dubbo 3.0正在动工的消息。Dubbo 3.0内核与Dubbo2.0完全不同，但兼容Dubbo 2.0。Dubbo 3.0将支持可选Service Mesh

2018年2月份，Dubbo捐给了Apache。另外，阿里巴巴对于Spring Cloud Alibaba生态的完善，以及Spring Cloud团队对于alibaba整个服务治理生态的支持，所以Dubbo未来依然是国内绝大部分公司的首要选择。

Dubbo Spring Cloud

Dubbo并不仅仅是一个RPC框架，它其实是自成一个生态。

- 面向接口代理的高性能RPC调用，提供高性能的基于代理的远程调用能力，服务以接口为粒度，为开发者屏蔽远程调用底层细节。

- 智能负载均衡，内置多种负载均衡策略，智能感知下游节点健康状况，显著减少调用延迟，提高系统吞吐量。
- 服务自动注册与发现，支持多种注册中心服务，服务实例上下线实时感知。
- 可视化的服务治理与运维，提供丰富服务治理、运维工具：随时查询服务元数据、服务健康状态及调用统计，实时下发路由策略、调整配置参数。
- 运行期流量调度，内置条件、脚本等路由策略，通过配置不同的路由规则，轻松实现灰度发布，同机房优先等功能。
- 高度可扩展能力，遵循微内核+插件的设计原则，所有核心能力如Protocol、Transport、Serialization被设计为扩展点，平等对待内置实现和第三方实现。

Dubbo实践

本次案例，使用的spring cloud 版本是Hoxton.SR12，alibaba版本2.2.6.RELEASE，Spring Boot版本，2.3.12.RELEASE

Dubbo一些特性

负载均衡

Random LoadBalance

权重随机算法，根据权重值进行随机负载

它的算法思想很简单。假设我们有一组服务器 servers = [A, B, C]，他们对应的权重为 weights = [5, 3, 2]，权重总和为10。现在把这些权重值平铺在一维坐标值上，[0, 5) 区间属于服务器 A, [5, 8) 区间属于服务器 B, [8, 10) 区间属于服务器 C。接下来通过随机数生成器生成一个范围在 [0, 10) 之间的随机数，然后计算这个随机数会落到哪个区间上。比如数字3会落到服务器 A 对应的区间上，此时返回服务器 A 即可。权重越大的机器，在坐标轴上对应的区间范围就越大，因此随机数生成器生成的数字就会有更大的概率落在此区间内。只要随机数生成器产生的随机数分布性很好，在经过多次选择后，每个服务器被选中的次数比例接近其权重比例。

RoundRobin LoadBalance

加权轮询算法

所谓轮询是指将请求轮流分配给每台服务器。举个例子，我们有三台服务器 A、B、C。我们将第一个请求分配给服务器 A，第二个请求分配给服务器 B，第三个请求分配给服务器 C，第四个请求再次分配给服务器 A。这个过程就叫做轮询。轮询是一种无状态负载均衡算法，实现简单，适用于每台服务器性能相近的场景下。但现实情况下，我们并不能保证每台服务器性能均相近。如果我们将等量的请求分配给性能较差的服务器，这显然是不合理的。因此，这个时候我们需要对轮询过程进行加权，以调控每台服务器的负载。经过加权后，每台服务器能够得到的请求数比例，接近或等于他们的权重比。比如服务器 A、B、C 权重比为 5:2:1。那么在8次请求中，服务器 A 将收到其中的5次请求，服务器 B 会收到其中的2次请求，服务器 C 则收到其中的1次请求

ConsistentHash LoadBalance

一致性hash算法，在前面我们有讲到过，就是基于hash环的方式来减少扩容带来的数据建议问题。

一致性hash算法采用的是hash环的方式来实现，原本我们使用的hash算法是对服务器或者对某一个目标表进行取模，但是一致性hash算法是对 2^{32} 取模，什么意思呢？

简单来说，一致性Hash算法将整个哈希值空间组织成一个虚拟的圆环，如假设某哈希函数H的值空间为 $0\sim 2^{32}-1$ （即哈希值是一个32位无符号整形）。

LeastActive LoadBalance

最少活跃调用数算法，活跃调用数越小，表明该服务提供者效率越高，单位时间内可处理更多的请求这个是比较科学的负载均衡算法。

每个服务提供者对应一个活跃数 active。初始情况下，所有服务提供者活跃数均为0。每收到一个请求，活跃数加1，完成请求后则将活跃数减1。在服务运行一段时间后，性能好的服务提供者处理请求的速度更快，因此活跃数下降的也越快，此时这样的服务提供者能够优先获取到新的服务请求

shortestresponse LoadBalance

最短响应时间负载均衡算法，

筛选成功调用响应时间最短的调用程序的数量，并计算这些调用程序的权重和数量。然后根据响应时间的长短来分配目标服务的路由权重。

容错机制

Failover Cluster (默认)

失败自动切换，当出现失败，重试其它服务器。通常用于读操作，但重试会带来更长延迟。可通过 `retries="2"` 来设置重试次数(不含第一次)。

Failfast Cluster

快速失败，只发起一次调用，失败立即报错。

通常用于非幂等性的写操作，比如新增记录。

Failsafe Cluster

失败安全，出现异常时，直接忽略。

通常用于写入审计日志等操作。

Fallback Cluster

失败自动恢复，后台记录失败请求，定时重发。

通常用于消息通知操作。

Forking Cluster

并行调用多个服务器，只要一个成功即返回。

通常用于实时性要求较高的读操作，但需要浪费更多服务资源。

可通过 `forks="2"` 来设置最大并行数。

Broadcast Cluster

广播调用所有提供者，逐个调用，任意一台报错则报错。（2.1.0开始支持）

通常用于通知所有提供者更新缓存或日志等本地资源信息。

在实际应用中 查询语句容错策略建议使用默认Failover Cluster，而增删改 建议使用 Failfast Cluster 或者 使用 Failover Cluster (`retries="0"`) 策略 防止出现数据重复添加等等其它问题！建议在设计接口时候把查询接口方法单独做一个接口提供查询。

服务降级

当某个非关键服务出现错误时，可以通过降级功能来临时屏蔽这个服务。降级可以有几个层面的分类：自动降级和人工降级；按照功能可以分为：读服务降级和写服务降级；

1. 对一些非核心服务进行人工降级，在大促之前通过降级开关关闭哪些推荐内容、评价等对主流程没有影响的功能
2. 故障降级，比如调用的远程服务挂了，网络故障、或者RPC服务返回异常。那么可以直接降级，降级的方案比如设置默认值、采用兜底数据（系统推荐的行为广告挂了，可以提前准备静态页面做返回）等等
3. 限流降级，在秒杀这种流量比较集中并且流量特别大的情况下，因为突发访问量特别大可能会导致系统支撑不了。这个时候可以采用限流来限制访问量。当达到阀值时，后续的请求被降级，比如进入排队页面，比如跳转到错误页（活动太火爆，稍后重试等）

那么，Dubbo中如何实现服务降级呢？Dubbo中提供了一个mock的配置，可以通过mock来实现当服务提供方出现网络异常或者挂掉以后，客户端不抛出异常，而是通过Mock数据返回自定义的数据

Nacos注册中心 (Eureka)



