阿里中间件团队博客

致力于成为中国第一，世界一流的中间件技术团队

从微服务治理的角度看RSocket,. Envoy和. Istio

2019-01-02 | 中间件小姐姐 | RSocket

很多同学看到这个题目，一定会提这样的问题：RSocket是个协议，Envoy是一个 proxy，Istio是service mesh control plane + data plane。 这三种技术怎么能放在一起比较呢?

的确，从技术定位的角度来讲，它们确实是有很大的差距。但是，如果我们用RSocket来治理微服务，会有哪些不同呢?

RSocket

RSocket是一种应用层协议，不是一个传输层的协议。一方面，它可以包容和支持不同的传输层协议和相关技术，比如tcp 和 proto buf。另一方面，它的重点是把反应流的实现，提升到应用层上来。

其实在底层的协议中，就有反应流的实现，tcp的滑动窗口就是很好的例子。但是往上，这种好的机制不见了，给编程的工作造成很多的麻烦。很大一部分的线上故障是由于阻塞链接造成的。另一方面，很多应用层的网络软件，从设计的时候就开始避免这样的麻烦，造成结构臃肿，通讯效率底下。简单的例子是如果所有的通讯都是反应式的，那就不用熔断了。

基于RSocket 的应用不止是端到端通讯，Broker也是对这个协议水到渠成的应用。作为一个反应式的Broker，它同样是异步，非阻塞的通讯方式，主要维护与就近的各个应用的链接以及和其它Broker的链接。与其它协议相比，它是多路复用，同时支持长链接。

经过这样的解释，不难理解，本文主要是针对RSocket应用通过RSocket Broker联结而形成的Mesh，与其它Service Mesh项目在不同层次和方面的对比。

RSocket vs .Envoy

Envoy作为一个proxy，它主要是基于HTTP2/HTTP1.1的协议，当然这样做是符合市场的口味，但是这个协议的局限性也限制了Envoy的性能。这就是我们比较的第一点，

1. Envoy不支持多路复用，非阻塞和有限支持长链接。说是有限，其实就是不支持，因为你的链接只要不能一直开着，就得依靠第三方做health check。这绝对增加开发难度。不支持多路复用，就无法对每个服务都开个链接，那么就要靠第三方作service registry。
   这样的限制，不但使得Envoy必须依靠一个control plane,自己无法独立担负weave mesh的重担，而且也大大限制了它的性能，比如新版本Istio Proxy（就是Envoy）用的联接池管理就占了很多的内存。

2. RSocket的主要障碍是应用程序之间必须要用RSocket通讯。随着Spring Cloud的推出，Spring Framework 5.2 即将要把RSocket作为缺省的反应通讯协议，以及Dubbo和RSocket 的整合，大家接触RSocket的机会也会越来越多。

3. 很多场合中会听到Envoy支持Polygoat,好像用了Envoy就不用SDK了。这种说法显然是错觉。SDK是一定要的，为了支持Polygoat，就要选多语言支持的SDK。因为调用另一个服务的代码还是发生在自己的程序中，这不是Envoy可以替代的。Envoy所说的省却SDK开发，是指所谓的“胖SDK”， 就是包括了服务发现和路由功能的SDK，类似大家现在用的Dubbo，那的确是会让SDK瘦身的。但是如果用了RSocket的Broker，这些SDK同样也不用再“胖”了，而且RSocket协议也有不同语言的SDK。

RSocket vs .Istion

除了上述的简化和高效等特性外，相比Istio，RSocket Broker 有一个主要的优势，那就是不依赖Kubernets。虽然Istio也号称不依赖Kubernets，但是在Kubernets外部署和管理sidecar proxy可不是一件容易的事，而RSocket Broker却是哪里都能部署。

作为一个Service Mesh solution, Istio其实是很难在 data center外应用的。那么对于众多的IoT设备怎么办？每一台手机上装个sidecar？而RSocket是很小且高效的SDK，这也是像Facebook这样的主要手机应用商选择RSocket的原因。

Istio主打的特性是observability, security and control。从observability和control方面来说，RSocket Broker虽然有接口，但是实现还不够，特别是API的部分。这也是社区要努力的一个方向。从security来说，如果是单纯RSocket的服务是不用开端口的，这是又一项由先进协议带来的对特性的简化，以后会有更多的介绍。

结论

很早以前，在分布程序中访问另一个服务是很直观，透明的事。微服务普及后，其为了“简化”微服务之间的通讯，引入了很多层的技术栈。这当然是好事，但是很多的决定是由于收到上一代的通讯协议的技术所限制。

RSocket的反应流技术，简化了程序间通讯对其它部件的依赖。我们可以享受Service Mesh提供的便利而不用那么复杂的技术栈。当然RSocket带来的好处不只是简单。在我们的初步实验中，RSocket Broker的service mesh比Istio带来将近10倍的速度提升。如果大家有兴趣，可以去了解一下RSocket。

> Andy Shi：阿里巴巴中间件硅谷团队 Istio 技术专家，Andy长期关注Service Mesh，在Cloud Foundry，Kubernetes，Envoy上有着丰富的实践和开发经验。

#RSocket

‹ 从微服务的角度看，如何 Be Cloud Native

Dubbo Ecosystem - 从微服务框架到微服务生态 ›