Dubbo 和 HSF 在阿里巴巴的实践：携手走向下一代云原生微服务

原创

阿里云云栖号 于 2021-09-13 11:03:13 发布 157 收藏
分类专栏： 云栖号技术分享 文章标签： zookeeper java 运维

本文链接：https://blog.csdn.net/yunqiinsight/article/details/120262671

简介： HSF 和 Dubbo 的融合是大势所趋。为了能更好的服务内外用户，也为了两个框架更好发展，Dubbo 3.0 和以 Dubbo 3.0 为内核适配集团内基础架构生态的 HSF 3 应运而生。

作者 | 郭浩

Dubbo 和 HSF 都是阿里巴巴目前在使用的微服务 RPC 框架。HSF 在阿里巴巴使用更多，承接了内部从单体应用到微服务的架构演进，支撑了阿里历年双十一的平稳运行；Dubbo 则在 2011 年开源后，迅速成为业界广受欢迎的微服务框架产品，在国内外均有着广泛应用。
自 2008 年 5 月发布第一个版本 1.1 后，经历数年迭代，HSF 从一个基础的 RPC 框架逐渐演变成为日支撑十万亿级别调用的易于扩展的微服务框架。内部场景中，用户既可以选择少量配置轻松接入微服务体系，获取高性能的稳定服务调用。也可以按照自身业务需求，对 HSF 进行扩展，获取整条链路的能力增强。

Dubbo 项目诞生于 2008 年，起初只在一个阿里内部的系统使用；2011 年，阿里 B2B 决定将整个项目开源，仅用了一年时间就收获了来自不同行业的大批用户；2014 年，由于内部团队调整，Dubbo 暂停更新；2017 年 9 月，Dubbo 3 重启开源，在 2019 年 5 月由 Apache 孵化毕业，成为第二个由阿里巴巴捐献至 Apache 毕业的项目。

Dubbo 和 HSF 在阿里巴巴的实践

2008 年的时候，集团内部淘系主要使用的服务框架是 HSF， 而 B2B 使用的则是 Dubbo。二者独立，各行其道，彼此不通。随着业务飞速发展，跨语言、跨平台、跨框架的需求日益明显，不同业务间彼此互联互通的呼声越来越高，而且很快演变成为几乎整个集团的需求。即淘系可以调用 B2B 的服务，反之亦然。

服务框架就像铁路的铁轨一样，是互通的基础，只有解决了服务框架的互通，才有可能完成更高层的业务互通，所以用相同的标准统一，共建新一代的服务框架是必然趋势。也就是最终的框架需要同时兼容 HSF1.x 和 Dubbo (包括 1.x 和 2.x) 的协议。对于集团内的需求而言，稳定和性能是核心，因此，当时选型了在电商这种高并发场景久经考验的 HSF 做为新一代服务框架核心。随后，HSF 推出了 2.0 的版本，并针对 HSF 之前版本的主要问题进行重构改造，降低了维护成本，进一步提高了稳定性和性能。HSF 2.0 解决了通讯协议支持不透明，序列化协议支持不透明等框架扩展性问题。基于 HSF 2.0 的 Java 版本，集团内也演进出了 CPP/NodeJs/PHP 等多语言的客户端。

由于兼容了 Dubbo 的协议，原有的 Dubbo 用户可以平滑地迁移到新版本上，所以 HSF 推出后很快就在集团全面铺开，部署的 server 数量达到数十万，基本完成了阿里巴巴内部微服务框架的统一，并经历了多年双十一零点流量洪峰的验证。

下一代微服务的挑战和机遇

然而，业务的发展和框架自身的迭代使得两个框架从协议层的简单兼容已经无法满足需要。随着云计算的不断发展和云原生理念的广泛传播，微服务的发展有着以下趋势：

1、K8s 成为资源调度的事实标准，Service Mesh 从提出到发展至今已经逐渐被越来越多用户所接受。屏蔽底层基础设施成为软件架构的一个核心演进目标，无论是阿里巴巴还是其他企业用户，所面临的问题都已经从是否上云变为如何平滑稳定地低成本迁移上云。

2、由于上云路径的多样以及由现有架构迁移至云原生架构的过渡态存在，部署应用的设施灵活异变，云上的微服务也呈现出多元化的趋势。跨语言、跨厂商、跨环境的调用必然会催生基于开放标准的统一协议和框架，以满足互通需求。

3、端上对后台服务的访问呈爆炸性的趋势增长，应用的规模和整个微服务体系的规模都随之增长。

这些趋势也给 HSF 和 Dubbo 带来了新的挑战。

第一，上云对内部闭源组件带来了冲击。微服务框架是基础组件，大部分公司在早期选型或业务发展到一定规模的时候都需要确定使用某一个框架。而一个稳定高效的自研框架通常需要较长时间的迭代来打磨优化。所以，大部分公司初期都会倾向于使用开源组件。对阿里云而言，这就带来了一个问题：内部使用的是 HSF 框架，而云上的用户大部分都是使用的开源 Dubbo 框架，两种框架在协议、内部模块抽象、编程接口和功能支持上都存在差异。如何能让使用了 HSF 的阿里集团内部组件的最优实践和前沿技术更简单直接地输出到云上，这是每一个做技术商业化的同学都会遇到和必须解决的问题。

第二，原有部门或公司的技术栈如何更快地融入到现有技术体系是一个绕不开的问题。一个典型的例子就是 2019 年加入阿里巴巴的考拉。考拉之前一直使用 Dubbo 作为微服务框架，基于 Dubbo 构建了大规模的微服务应用，迁移的成本高，风险也大。需要集团和考拉的基础架构部门耗费较长的时间进行迁移前调研、方案设计，确保基本可行后再开始改动。从分批灰度上线，再到最终全量上线。这种换血式的改动不仅需要耗费大量人力，时间跨度也很长，会影响到业务的发展和稳定性。

第三，由于历史原因，集团内部始终存在着一定数量的 Dubbo 用户。为了更好的服务这部分用户，HSF 框架对 Dubbo 进行了协议层和 API 层的兼容。但这种兼容仅限于互通，随着 Dubbo 开源社区的多年发展，这种基础的兼容在容灾、性能和可迭代性方面，都有着较大的劣势，同时很难对齐 Dubbo 的服务治理体系。在稳定性方面也存在风险，更无法享受到集团技术发展和 Dubbo 社区演进的技术红利。

产生这些问题的根本原因是闭源的 HSF 无法直接用于广大云上用户和外部其他用户，而开源产品对闭源产品的挑战会随着开源和云的不断发展愈演愈烈。越早解决这个问题，阿里巴巴和外部企业用户的云原生迁移成本越低，产生的价值也就越大。

最简单直接的方式是将 HSF 也开源出去。但这又会面临两个新问题。第一， Dubbo 是阿里较早开源的明星产品，如果 HSF 也开源，这两个同类框架的关系和适用场景如何划分，不仅外部用户会有疑惑，重复开源也不利于品牌建设。第二，国内外现有的 Dubbo 用户如果想上阿里云，则需要使用基于 HSF 的现有解决方案，需要花费巨大精力将所有用到 Dubbo 的应用迁移到 HSF，成本和稳定性都是不得不考虑的问题 。以上两点原因说明目前已经不是开源 HSF 的最好时机。

既然 HSF 不能走出去，那剩下的解决方式就是让 Dubbo 走进来。内部采用核心融合的方式，基于 Dubbo 内核重新构建 HSF 框架。

品牌建设上，融合可以使 Dubbo 现有的广泛影响力得以持续发展，Dubbo 在集团内大规模落地后，会产生良好的原厂品牌示范效应，外部用户也会有更多的信心在进行微服务框架选型时选择 Dubbo。同时，目前已经使用 Dubbo 的用户也有更充分的理由不断追随版本演进，享受阿里巴巴开源带来的技术红利。

工程实践上，使用 Dubbo 重构 HSF 这种从内部重新统一的可行性更高，迁移的复杂度可控，能够逐步地有序实现。内部完善的测试流程和丰富的场景是对 Dubbo 最好的功能回归测试。内外统一也是兼顾商业化和内部支持的最好方式。在重构的过程中，不断完善功能，提高性能，拥抱更新的更云原生的技术栈，这也是提升集团内部用户体验的最佳方式。

因此，HSF 和 Dubbo 的融合是大势所趋。为了能更好的服务内外用户，也为了两个框架更好发展，Dubbo 3.0 和以 Dubbo 3.0 为内核适配集团内基础架构生态的 HSF 3 应运而生。

下一代云原生微服务

首先总体上介绍一下 Dubbo 3.0 。

- Dubbo 3.0 支持全新的服务发现模型，Dubbo 3.0 尝试从应用模型入手，优化存储结构，对齐云原生主流设计模型，避免在模型上带来的互通问题。新模型在数据组织上高度压缩，能有效提高性能和集群的可伸缩性。
- Dubbo 3.0 提出了下一代 RPC 协议 —— Triple。这是一个基于 HTTP/2 设计的完全兼容 gRPC 协议的开放性新协议，由于是基于 HTTP/2 设计的，具有极高的网关友好型和穿透性；完全兼容 gRPC 协议是的天然在多语言互通方面上具有优势。
- Dubbo 3.0 面向云原生流量治理，提出了一套能够覆盖传统 SDK 部署、Service Mesh 化部署、VM 虚拟机部署、Container 容器部署等场景的统一治理规则，支持一份规则治理大部分场景，大大降低流量治理治理成本，使得异构体系下全局流量治理变得可能。
- Dubbo 3.0 将提供接入 Service Mesh 的解决方案，面向 Mesh 场景，Dubbo 3.0 提出了两种接入方式，一种是 Thin SDK 模式，部署模型和当前 Service Mesh 主流部署场景完全一样，而 Dubbo 将进行瘦身，屏蔽掉与 Mesh 相同的治理功能，仅保留核心的 RPC 能力；第二种是 Proxyless 模式，Dubbo 将接替 Sidecar 的工作职责，主动与控制面进行通信，基于 Dubbo 3.0 的统一治理规则应用云原生流量治理功能。

1、应用级注册发现模型

应用级注册发现模型的原型最早在 Dubbo 2.7.6 版本提出，经过数个版本的迭代最终形成了 Dubbo 3.0 中的稳定模型。在 Dubbo 2.7 及以前版本中，应用进行服务注册和发现时，都是以接口为粒度，每个接口都会对应在注册中心上的一条数据，不同的机器会注册上属于当前机器的元数据信息或者接口级别的配置信息，如序列化、机房，单元、超时配置等。所有提供此服务的服务端在进行重启或者发布时，都会在接口粒度独立的变更。

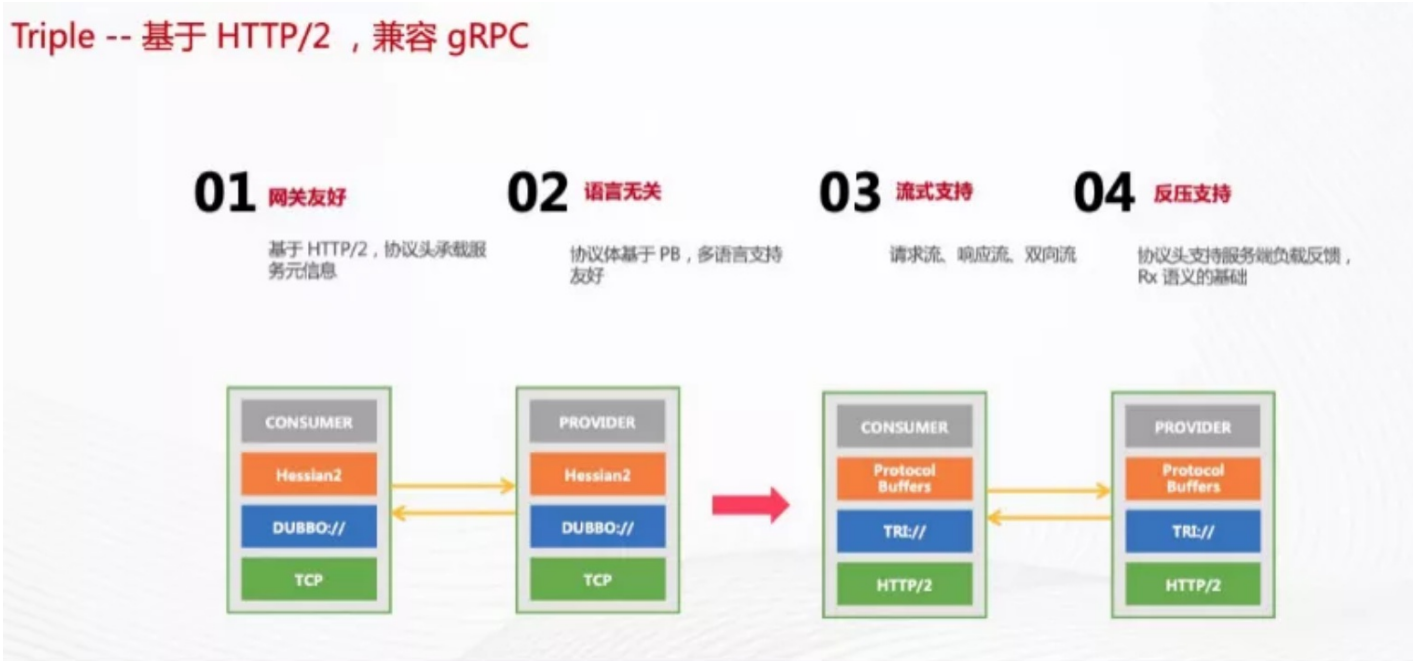
举个例子，一个网关类应用依赖了上游应用的 30 个接口，当上游应用在发布时，就有 30 个对应的地址列表在进行机器上线和下线。以接口作为注册发现第一公民的模型是最早的 SOA 或微服务的拆分方式，提供了灵活的根据单一服务单一节点独立动态变更的能力。随着业务的发展，单一应用依赖的服务数在不断增多，每个服务提供方的机器数量也由于业务或容量原因不断增长。客户端依赖的总服务地址数上涨迅速，注册中心等相关依赖组件的压力倍增。

我们注意到了微服务模型发展的两个趋势：首先，随着单体应用拆分为多微服务应用的基本完成，大规模的服务拆分和重组已经不再是痛点，大部分接口都只被一个应用提供或者固定几个应用提供。其次，大量的用于标志地址信息的 URL 都是存在极大冗余的，如超时时间，序列化，这些配置变更频率极低，却在每个 URL 中都出现。所以应用级注册发现应运而生。

应用级服务发现以应用作为注册发现的基本维度。和接口级的主要区别是，一个应用提供了 100 个接口，按照接口级粒度需要在注册中心上注册 100 个节点，如果这个应用有 100 台机器，那每次发布，对于它的客户端都是 10000 个虚拟节点的变更。而应用级注册发现则只需要 1 个节点， 每次发布只变更 100 个虚拟节点。对于依赖服务数多、机器多的应用而言，是几十到上百分之一数量级的规模下降。内存占用上也会至少下降一半。

最后，由于这个新的服务发现与 Spring Cloud、Service Mesh 等体系下的服务发现模型是一致的，因此 Dubbo 可以从注册中心层面与其他体系下的节点实现互发现，实现异构微服务体系的互联互通。

2、下一代 RPC 协议——Triple



作为 RPC 框架最基础的能力还是完成跨业务进程的服务调用，将服务组成链、组成网，这其中最核心的载体就是协议。Dubbo 2.0 提供了 RPC 的核心语义，包括协议头、标志位、请求 ID 以及请求 / 响应数据，他们被按照一定的顺序以二进制数据的方式组合在一起。

在云原生时代，Dubbo 2.0 协议主要面临两个挑战。一是生态不互通，用户很难直接理解二进制的协议。第二是对 Mesh 等网关型组件不够友好，需要完整的解析协议才能获取到所需要的调用元数据，如一些 RPC 上下文，从性能到易用性方面都会面临挑战。同时，老版本 Dubbo 2.0 RPC 协议的设计与实现，已在实践中被证实在□些□面限制了业务架构的发展，□如从终端设备到后端服务的交互、微服务架构中多语言的采用、服务间的数据传输模型等。那么，在支持已有的功能和解决存在的问题的前提下，下一代的协议需要有哪些特性呢？

首先，新协议应该易扩展，包括但不限于 Tracing/ Monitoring 等支持，也应该能被各层设备识别，降低用户理解难度。

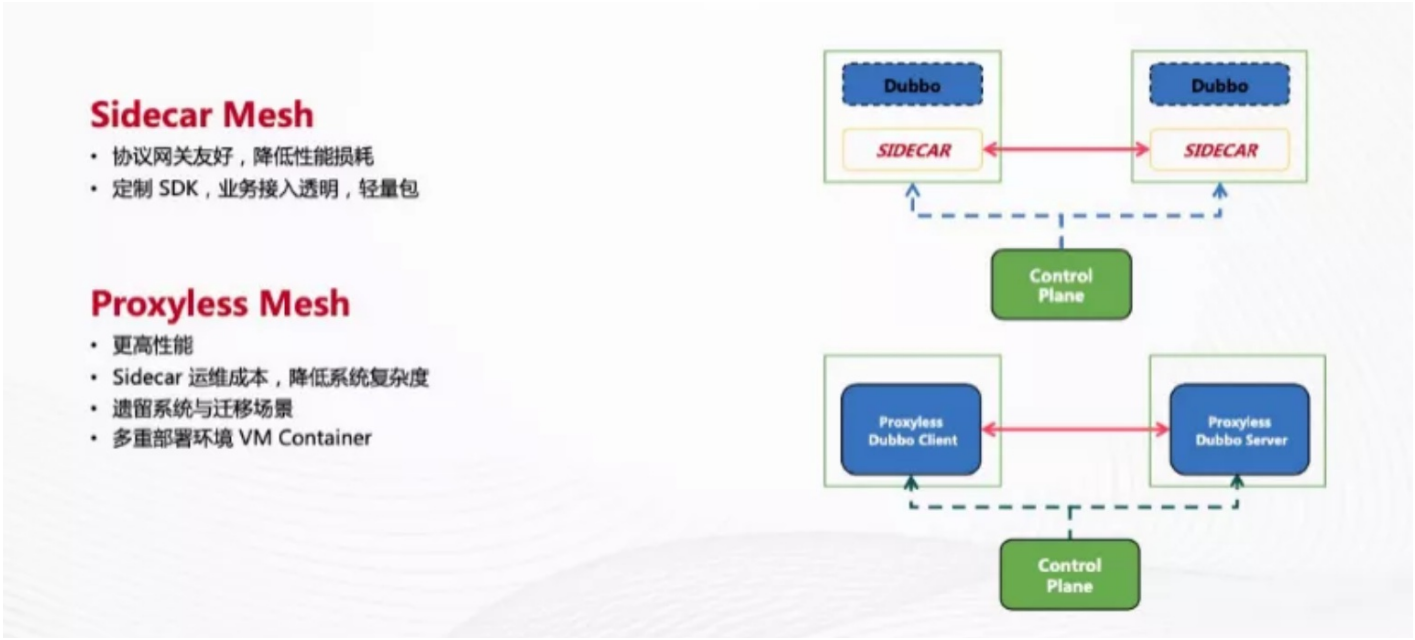
其次，协议需要解决跨语言互通的问题，传统的多语言多 SDK 模式和 Mesh 化跨语言模式都需要一种更通用易扩展的数据传输格式。

最后，协议应该提供更完善的请求模型，除了 Request/Response 模型，还应该支持 Streaming 和 Bidirectional 等模型。

基于这些需求，HTTP2/protobuf 的组合是最符合的。提到这两个，大家可能很容易想到 gRPC 协议。那新一代的协议和 gRPC 的关系是什么呢。

首先，Dubbo 新协议是基于 GRPC 扩展的协议，这也保证了在生态系统上新协议和 GRPC 是能够互通和共享的。其次，在这个基础上，Dubbo 新协议将更原生的支持 Dubbo 的服务治理，提供更大的灵活性。在序列化方面，由于目前大多数应用方还没有使用 Protobuf ，所以新协议会在序列化方面给予足够的支持，平滑的适配现有序列化，方便迁移到 Protobuf。在请求模型上，新协议将原生支持端到端的全链路 Reactive，这也是 gRPC 协议所不具备的。

3、原生接入 Service Mesh



如何让 Dubbo 在 Service Mesh 体系下落地，社区开发团队调研众多方案，最终确定了最适合 Dubbo 3.0 的两种形态的 Mesh 方案。

□种是经典的基于 Sidecar 的 Service Mesh，另□种是无 Sidecar 的 Proxyless Mesh。对于 Sidecar Mesh 方案，其部署方式和当前主流 Service Mesh 部署方案一致，Dubbo 3.0 的重点是尽量给业务应用提供完全透明的升级体验，这不止是编程视角的无感升级，还包括通过 Dubbo 3.0 轻量化、Triple 协议等，让整个调用链路上的损耗与运维成本也降低到最低。这个方案也被称为 Thin SDK 方案，而 Thin 的地方就是在去除所有不需要的组件。Proxyless Mesh 部署方案则是 Dubbo 3.0 规划的另□种 Mesh 形态，目标是不需要启动 Sidecar，由传统 SDK 直接与控制面交互。

我们设想这对以下□种场景会□常适用 Proxyless Mesh 部署方案：

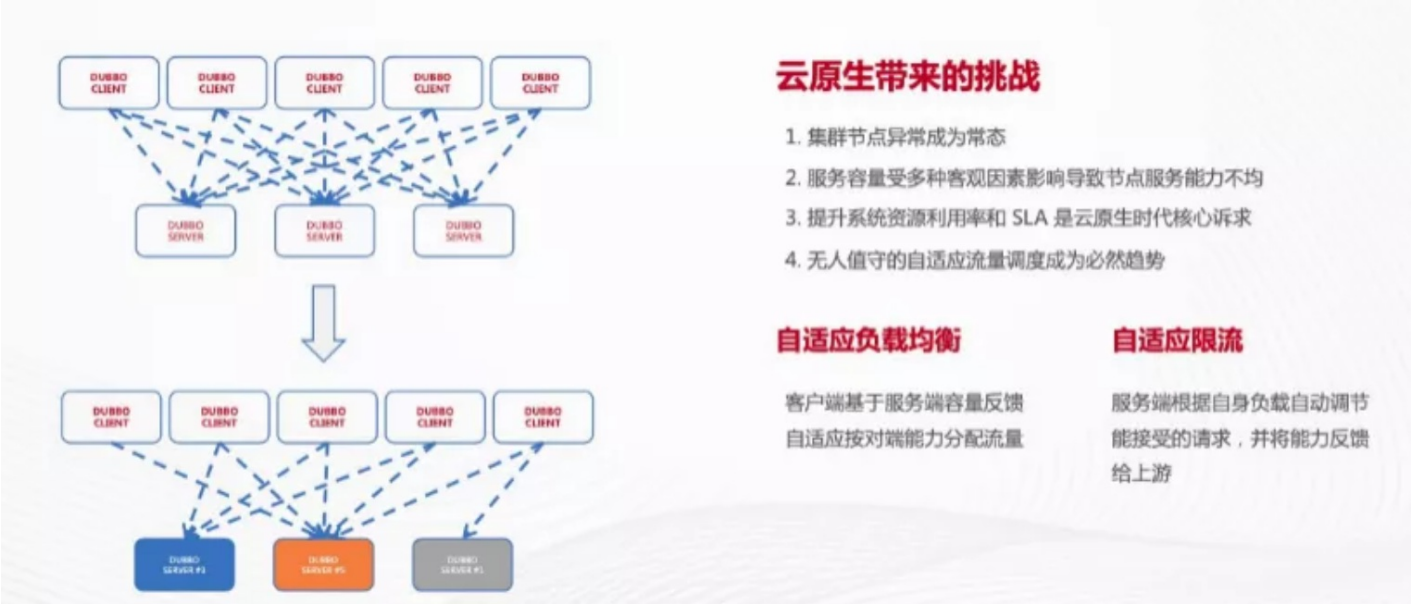
一是业务方期望升级 Mesh 方案，但却无法接受由于 Sidecar 进行流量劫持所带来的性能损耗，这种情况常见于核心业务场景；

二是期望降低由于部署 Sidecar 带来的运维成本，降低系统复杂度；三是遗留系统升级缓慢，迁移过程漫长，多种部署架构□期共存。

最后是，多种部署环境，这里的多种部署环境包括了如 VM 虚拟机、Container 容器等多种部署方式，也包括了多种类型应用混合部署，例如 Thin SDK 与 Proxyless 方案混合部署，对性能敏感应用部署 Proxyless 模式，对于周边应用采用 Thin SDK 部署方案，多种数据面共同由统一控制面进行调度。

将这两种形态统筹来看，在不同的业务场景、不同的迁移阶段、不同的基础设施保障情况下， Dubbo 都会有 Mesh □案可供选择。

4、柔性服务增强



云原生带来了技术标准化的重大变革，如何让应用在云上更简单地创建和运行，并具备可弹性扩展的能力，是所有云原生基础组件的核心目标。借助云原生技术带来的弹性能力，应用可以在极短时间内扩容出一大批机器以支撑业务需要。比如为了应对零点秒杀场景或者突发事件，应用本身往往需要数千甚至数万的节点数来以满足用户的需要，但是在扩容的同时也带来了许多在云原生场景下集群大规模部署的问题。比如由于集群节点极多导致的节点异常频发、服务容量受多种客观因素影响导致节点服务能力不均等。

Dubbo 期待基于一种柔性的集群调度机制来解决这些问题。这种机制主要解决的问题有两个方面：一是在节点异常的情况下，分布式服务能够保持稳定，不出现雪崩等问题；二是对于大规模的应用，能够以最佳态运行，提供较高的吞吐量和性能。从单一服务视角看，Dubbo 期望的目标是对外提供一种压不垮的服务，即是在请求数特别高的情况下，可以通过选择性地拒绝一些的请求来保证总体业务的正确性、时效性。

从分布式视角看，要尽可能降低因为复杂的拓扑、不同节点性能不一导致总体性能的下降，柔性调度机制能够以最优的方式动态分配流量，使异构系统能够根据运行时的准确服务容量合理分配请求，从而达到性能最优。

5、业务收益

对业务而言，可能更关心的是升级到 Dubbo 3.0 能带来哪些收益。总结提炼出两大关键词，分别是应用自身的性能稳定性的提升以及云原生的原生接入。

- 性能与稳定性方面，Dubbo 3.0 会着力关注大规模集群部署的场景，通过优化数据存储方式，来降低单机资源损耗，同时可以在应对超大规模集群的水平扩容的时候，保证整个集群的稳定性。另外，在 Dubbo 3.0 提出了柔性服务的概念，也能够在一定程度上有效保证和提高全链路总体可靠性和资源的利用率。
- 第二是关于云原生方面，Dubbo 3.0 是 Dubbo 全面拥抱云原生的里程碑版本，当前 Dubbo 在国内外有基数巨大的用户群体，随着云原生时代的到来，这些用户上云的需求越来越强烈，Dubbo 3.0 将提供完整可靠的解决方案、迁移路径与最佳实践帮助企业实现云原生转型，享受云原生带来的红利。

从已经落地的结果上看，Dubbo 3.0 能□幅降低框架带来的额外资源消耗，提升系统资源利用率。从单机视角，Dubbo 3.0 能节省约 50% 的内存占□；从集群视角，Dubbo3 能□持百万实例数的大规模集群，为未来更大规模的业务扩容打下基础；Dubbo3 对 Reactive Stream 等通信模型的支持，在大文件传输、流式等业务场景下能带来整体吞吐量的□幅提升。

架构方面，Dubbo 3.0 给业务架构升级带来了更多可能性。Dubbo 原来的协议在□定程度上束缚了微服务接□□式的，举个例子，移动端、前端业务要接入 Dubbo 后端服务，需要经过网关层的协议转换，再比如，Dubbo 只□持 request-response 模式的通信，这使得□些需要流式传输或反向通信的场景□法得到很好的支持。

在云原生转型过程中，业务最关心的就是改动和稳定性，能不能不改代码或者少改代码就能升级到云原生环境，在业务上云过程的选型中至关重要。Dubbo 3.0 给业务侧云原□生升级带来了整体的解决方案。不论是底层基础设施升级带动业务升级，还是为解决业务痛点而进行的主动升级，Dubbo 3.0 所提供的云原生解决方案都能帮助产品快速升级，进入云原生时代。

6、现状和 Roadmap

内部使用上，Dubbo 3.0 已经在考拉业务的数百应用的数万节点中全面落地，大量应用使用 Dubbo 3.0 轻松完成应用上云，目前正在电商核心应用中大规模试点和逐步落地，以及开启应用级注册发现、Triple 协议等新特性。开源用户和商业化应用目前也在从原有的 HSF2 或 Dubbo 2.0 迁移至 Dubbo 3.0 ，服务框架团队和社区正在整理和编写相关迁移的最佳实践，一段时间后这些文档就会和大家见面。

Dubbo 3.0 作为捐给 Apache 后的一个里程碑版本已经在今年 6 月份正式发布了，这也代表着 Apache Dubbo 全面拥抱云原生的一个节点。在 2021 年 11 月我们会发布 Dubbo 3.1 版本，届时会带来 Dubbo 在 Mesh 场景下部署的最佳实践。

2022 年 3 月会发布 Dubbo 3.2 版本，这个版本将带来服务柔性的全面支持，在大规模应用部署下实现智能流量调度，提高系统稳定性与资源利用率。

回顾过去，Dubbo 和 HSF 在阿里巴巴和微服务框架的发展的不同阶段都起到了至关重要的作用。立足现在，放眼未来，Dubbo 3.0 和基于 Dubbo 3.0 内核的 HSF 正在外部和内部齐头并进，做最稳定高性能的微服务框架，给用户最好的使用体验，继续在云原生时代引领微服务的发展。

作者介绍：郭浩，阿里巴巴服务框架负责人，Dubbo 3.0 架构师，专注分布式系统架构

原文链接