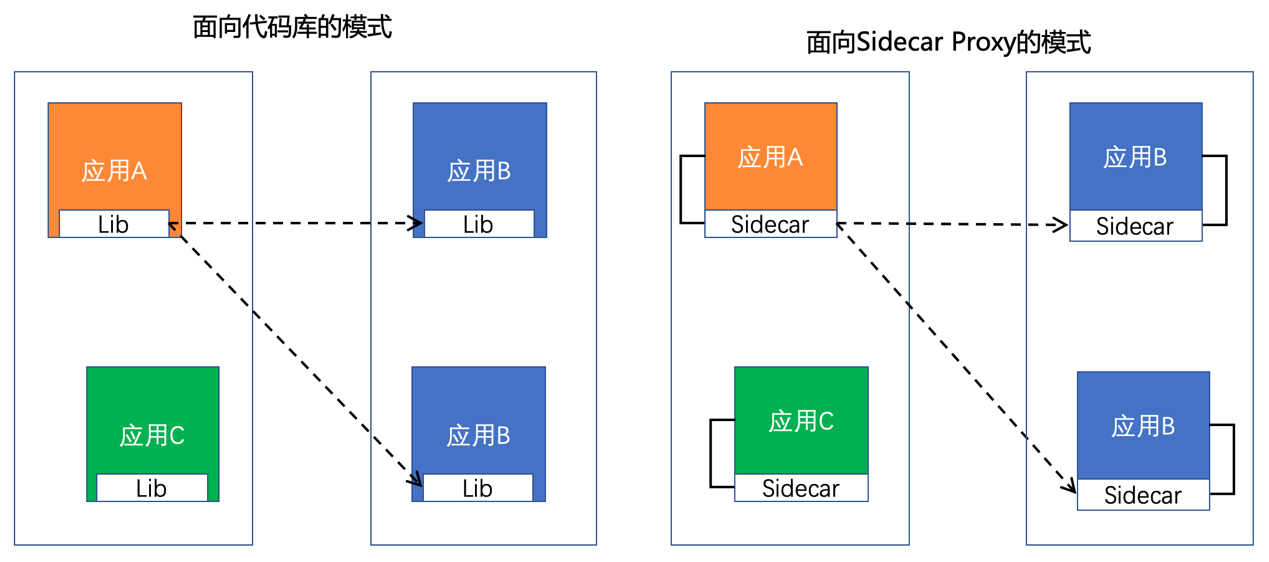
**[从单体到混乱的微服务，阿里云托管式服务网格是如何诞生的？](https://www.kubernetes.org.cn/8194.html)**

在服务网格技术使用之前，为了更快更灵活地进行业务创新, 我们常常会把现有应用进行现代化改造, 把单体应用程序分拆为分布式的微服务架构。通常来说,  在微服务架构模式的变迁过程中, 最初都是面向代码库的模式。

对这些微服务治理的实现, 往往是以代码库的方式把这些服务治理的逻辑构建在应用程序本身中，这些代码库中包括了流量管理、熔断、重试、客户端负载均衡、安全以及可观测性等这样的一些功能。这些代码库随着功能的不断增强, 版本也随之变更，因为版本不同导致的冲突问题处处可见。此外，库的版本一旦变更，即使你的应用逻辑并没有任何变化，整个应用也要随之全部变更。由此可见, 随着应用的增长和团队数量的增加，跨服务一致地使用这些服务治理功能会变得非常复杂。



服务治理的能力 Sidecar 化

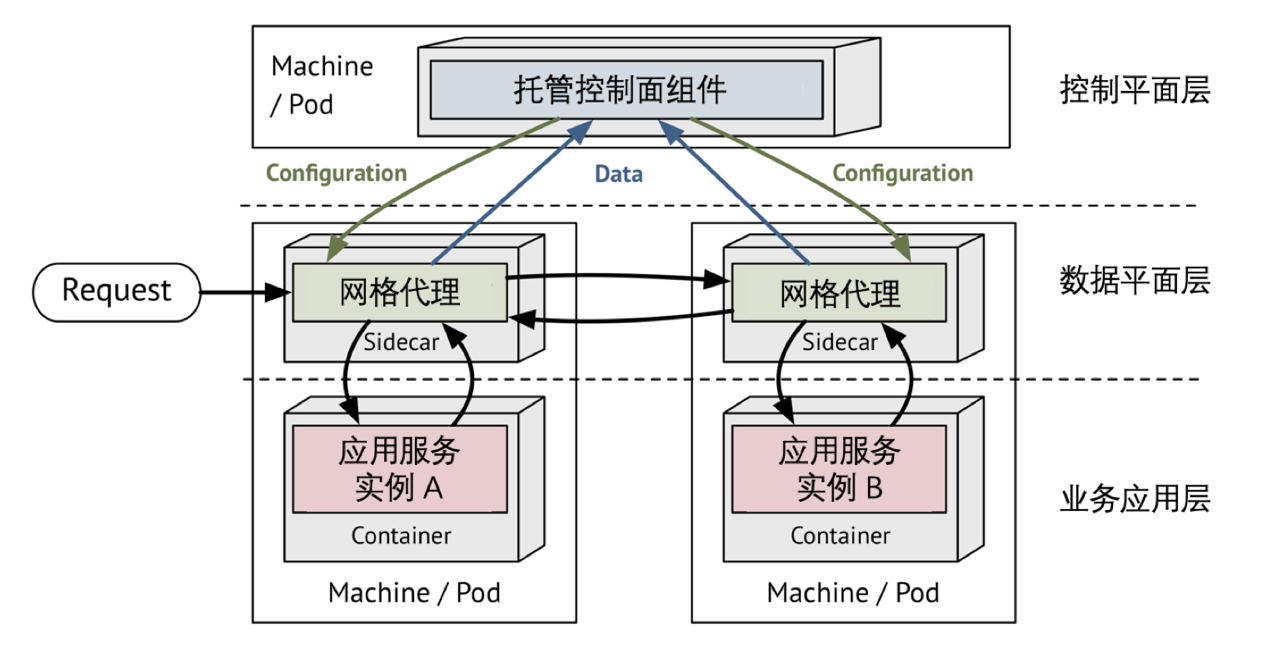
通过把这些服务治理的能力 Sidecar 化，就能够把服务治理的能力与应用程序本身进行了解耦，可以较好地支持多种编程语言、同时这些 Sidecar 能力不需要依赖于某种特定技术框架。这就是我们常说的面向 Sidecar proxy 的架构模式。

随着这些 Sidecar 代理功能的增强，原本需要在代码库中实现的服务治理功能被抽象化为一个个通用组件, 并被逐渐地下沉到代理中。这些服务治理能力的标准化、统一化，可以解决复杂系统微服务实现中面临的差异大、缺少共性的问题，可以很好地支持不同的编程语言、不同的框架。

通过把应用服务通信能力抽象下沉到基础设施,   使得开发人员可以更加聚焦于业务应用本身开发,  而让基础设施来提供这些通用的能力。

与此同时， 容器编排技术的更加成熟，也加速了 Sidecar 代理的普及与使用的便捷。Kubernetes 作为一个出色的容器部署和管理平台、Istio 作为应用服务治理的平台，两者的结合成为了将这些应用服务通信能力下沉到基础设施的载体。

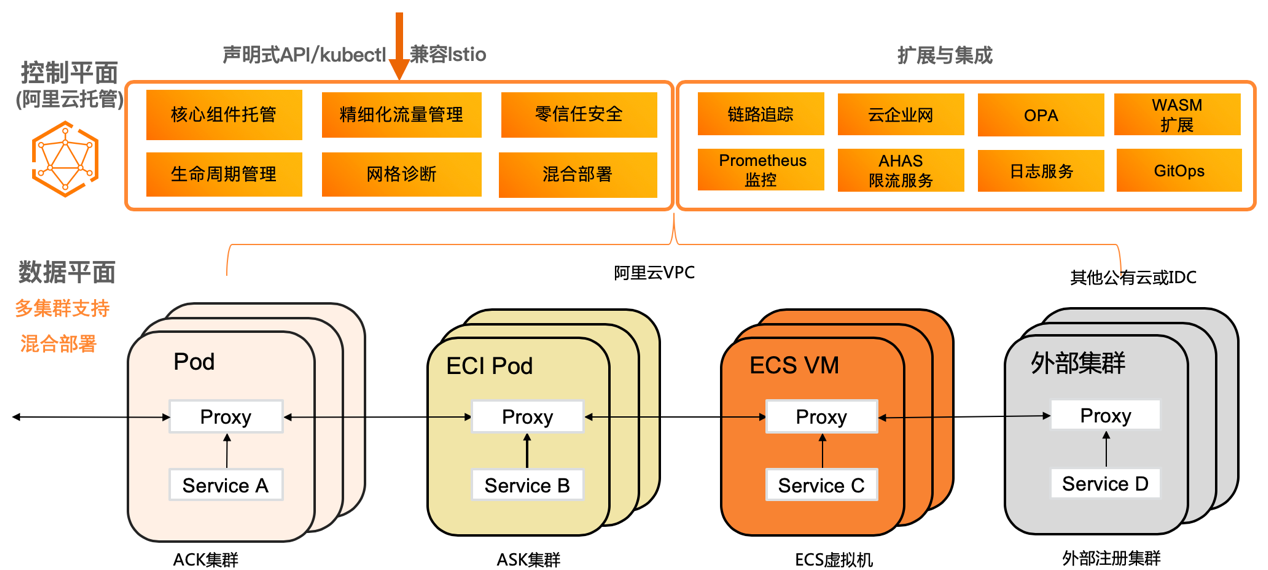
在云原生应用模型中，一个应用程序可能会包含数百个服务，每个服务又有数百个实例构成，那么这些成百上千个应用程序的 Sidecar 代理如何统一管理，这就是服务网格中定义的控制平面部分要解决的问题。作为代理，Envoy 非常适合服务网格的场景，但要发挥 Envoy 的最大价值，就需要使它很好地与底层基础设施或组件紧密配合。



这些 Sidecar 代理形成一个网状的数据平面，通过该数据平面处理和观察所有服务间的流量。数据平面扮演了一个用来建立、保护和控制通过网格的流量的角色。

负责数据平面如何执行的管理组件称为控制平面。控制平面是服务网格的大脑，并为网格使用人员提供公开 API，以便较容易地操纵网络行为。

启用服务网格之后,  开发人员、运维人员以及 SRE 团队将以统一的、声明的方式解决应用服务管理问题。



服务网格 ASM 产品架构

作为业内首个全托管 Istio 兼容的服务网格产品 ASM，一开始从架构上就保持了与社区、业界趋势的一致性，控制平面的组件托管在阿里云侧，与数据面侧的用户集群独立。ASM 产品是基于社区开源的 Istio 定制实现的，在托管的控制面侧提供了用于支撑精细化的流量管理和安全管理的组件能力。通过托管模式，解耦了 Istio 组件与所管理的 K8s 集群的生命周期管理，使得架构更加灵活，提升了系统的可伸缩性。

* 在深入分析服务网格方面，提供了网格诊断能力，把过去一年多来客户遇到的问题以及如何解决这些问题的手段变成产品能力, 帮助用户快速定位遇到的问题；
* 在扩展与集成方面，ASM 产品整合阿里云服务包括可观测性服务链路追踪/日志服务/Prometheus 监控等、跨 VPC 网络互连 CEN 能力等，同时也优化整合了社区开源软件包括 OPA 的支持、授权服务的定制化能力、限流服务等。

此外，随着 Istio 新架构的优化，将 WebAssembly 技术引入服务网格，解决代理扩展的问题。这样一来,  ASM 架构就变成了“托管的高可用弹性控制平面 + 可扩展的插件式的数据平面“的模式。

在数据平面的支持上，ASM 产品可以支持多种不同的计算基础设施，这包括了阿里云提供的公有云 ACK 集群（其中包括了托管的 K8s 集群和专有 K8s 集群）、也包括对的无服务器 Kubernetes 容器服务 ASK 集群的支持。同时, 对非容器化应用例如运行在 ECS 虚拟机上的应用服务网格化的支持。

此外，ASM 也推出了一个支持多云混合云的能力，能够针对外部的非阿里云 K8s 集群进行支持，不论这个集群是在用户自建的 IDC 机房，还是在其他的公有云之上，都可以通过 ASM 进行统一的服务治理。

多种类型计算服务统一管理的基础设施

接下来,  将会介绍托管式服务网格在成为多种类型计算服务统一管理的基础设施中, 如何提供了统一的流量管理能力、统一的服务安全能力、统一的服务可观测性能力、以及如何基于 WebAssembly 实现统一的数据面可扩展能力。

**1. 统一的流量管理能力**

关于统一的流量管理能力方面, 重点讲述 2 个方面。

**第一个是基于位置实现流量路由请求。**在大规模服务场景下, 成千上万个服务运行在不同地域的多种类型计算设施上, 这些服务需要相互调用来完成完整的功能。为了确保获得最佳性能，应当将流量路由到最近的服务，使得流量尽可能在同一个区域内，而不是只依赖于 Kubernetes 默认提供的轮询方式进行负载均衡。服务网格应当提供这样的基于位置的路由能力，一方面, 可以将流量路由到最靠近的容器, 实现本地优先的负载均衡能力, 并在主服务出现故障时可以切换到备用服务。另一方面, 提供局部加权的负载平衡能力, 能够根据实际需要, 将流量按比例拆分到不同的地域。

**第二个是关于非 K8s 工作负载的网格化统一管理**。在一个托管的服务网格实例中, 我们可以添加若干个 K8s 集群, 并在控制面定义路由规则的配置, 也可以定义网关服务等。为了能够统一地管理非 K8s 工作负载, 我们通过一个 WorkloadEntry 的 CRD 来定义工作负载的标签以及该工作负载运行的 IP 地址等信息。然后通过 ServiceEntry CRD 将这个工作负载注册到服务网格中, 并提供类似于 K8s Pod 的处理机制来处理这些非 K8s 工作负载。譬如可以通过 selector 机制路由到对应的Pod或者这个非容器应用上。

**2. 统一的服务安全能力**

关于统一的服务安全能力, 托管服务网格为多种不同计算基础设施上的应用服务提供统一的主子账户支持 / RAM 授权支持。在此基础上, 提供统一的 TLS 认证与 JWT 认证,  支持启用与禁用 TLS 认证的简易切换、支持以渐进方式逐步实现双向 TLS 认证; 支持以细粒度的认证范围, 包括 namespace 与 workload 级别。此外, 服务网格也提供对 JWT 认证能力的支持, 使得这种 TOKEN 认证机制不再依赖于某种特定实现框架就可以统一透出。

* 在 RBAC 授权方面, 针对不同协议提供了统一的授权策略, 可以在不同粒度上支持, 包括 namespace / service / port 级别的授权；
* 在审计方面, 可以灵活开启网格审计功能, 并可以查看审计报表、查看日志记录以及设置告警规则等；
* 在策略管理方面，提供了开放策略代理（OPA）的集成, 用户可以使用描述性策略语言定义相应的安全策略。此外也提供了自定义授权服务 external\_auth grpc 的对接。只要遵循这一接口定义, 任意授权服务都可以集成到服务网格中。

**3. 统一的服务可观测性**

统一的服务可观测性, 分为 3 个方面。

* **一是日志分析能力**：通过对数据平面的 AccessLog 采集分析, 特别是对入口网关日志的分析, 可以分析出服务请求的流量情况、状态码比例等, 从而可以进一步优化这些服务间的调用；
* **二是分布式追踪能力**：为分布式应用的开发者提供了完整的调用链路还原、调用请求量统计、链路拓扑、应用依赖分析等工具，可以帮助开发者快速分析和诊断分布式应用架构下的性能瓶颈，提高微服务时代下的开发诊断效率；
* **三是监控能力**：根据监视的四个维度（延迟，流量，错误和饱和度）生成一组服务指标, 来了解、监视网格中服务的行为。

**4. 统一的数据面可扩展能力**

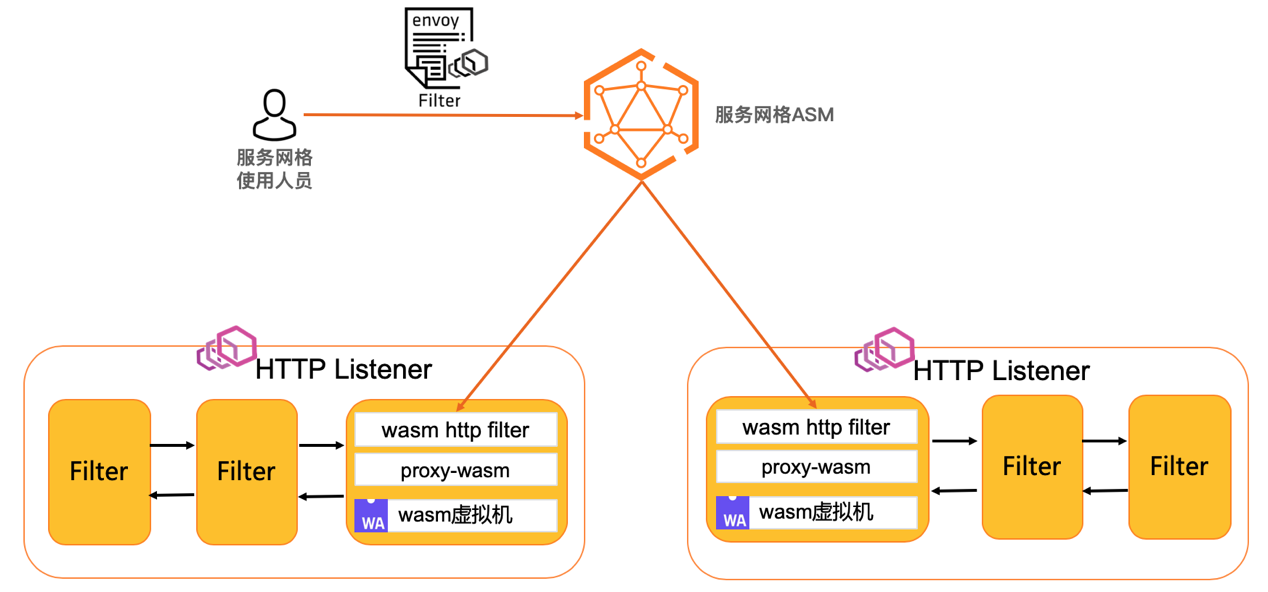
尽管 sidecar 代理已经把服务治理过程中常用的一些功能进行了封装实现，但它的可扩展能力一定是必须具备的，譬如如何与已有的后端系统做对接，如何解决用户的一些特定需求。这个时候，一个 Sidecar 代理的可扩展性显得尤为重要，而且在一定程度上会影响 Sidecar 代理的普及。

在 Istio 之前的架构中，对 Sidecar 能力的扩展主要集中在 Mixer 组件上。Sidecar 代理的每个服务到服务连接都需要连接到 Mixer，以进行指标报告和授权检查，这样会导致服务之间的调用延迟更长，伸缩性也变差。同时，Envoy 要求使用代理程序的编程语言 C++ 编写，然后编译为代理二进制文件。对于大多 Istio 用户而言，这种扩展能力具有一定的挑战性。

而在采用了新架构之后，Istio 把代理的扩展能力从 Mixer 下移到了数据平面的 Envoy 本身中, 并且使用 WebAssembly 技术将其扩展模型与 Envoy 进行了合并。WebAssembly 支持几种不同语言的开发，然后将扩展编译为可移植字节码格式。这种扩展方式既简化了向 Istio 添加自定义功能的过程，又通过将决策过程转移到代理中而不是将其种植到 Mixer 上来减少了延迟。使用 WebAssembly（WASM）实现过滤器 Filter 的扩展, 可以获得以下好处：

* 敏捷性：过滤器 Filter 可以动态加载到正在运行的 Envoy 进程中，而无需停止或重新编译；
* 可维护性：不必更改 Envoy 自身基础代码库即可扩展其功能；
* 多样性：可以将流行的编程语言（例如 C / C ++ 和 Rust）编译为 WASM，因此开发人员可以使用他们选择的编程语言来实现过滤器 Filter；
* 可靠性和隔离性：过滤器 Filter 会被部署到 VM 沙箱中，因此与 Envoy 进程本身是隔离的；即使当 WASM Filter 出现问题导致崩溃时，它也不会影响 Envoy 进程；
* 安全性：过滤器 Filter 通过预定义 API 与 Envoy 代理进行通信，因此它们可以访问并只能修改有限数量的连接或请求属性。

阿里云服务网格 ASM 产品中提供了对 WebAssembly（WASM）技术的支持，服务网格使用人员可以把扩展的 WASM Filter 通过 ASM 部署到数据面集群中相应的 Envoy 代理中。通过自研的 ASMFilterDeployment 组件,  可以支持动态加载插件、简单易用、以及支持热更新等能力。



通过这种过滤器扩展机制，可以轻松扩展 Envoy 的功能并将其在服务网格中的应用推向了新的高度。

服务网格实践之成熟度模型



服务网格作为应用服务通信的统一基础设施,  可以（并且应该）逐步采用。在此, 我们推出了它的实践之成熟度模型, 分为了 5 个层次, 分别为**一键启用**、**可观测提升**、**安全加固**、**多种基础设施的支持**, 以及**多集群混合管理**。这 5 个方面分别涵盖了前面讲述的统一流量管理、统一可观测性、统一服务安全以及支持不同的计算基础设施和多集群非容器化应用的混合管理。