ServiceMesh 实践分享学习总结

随着 ServiceMesh 技术的演进，以及在当前以 SDK 方式提供给业务方的微服务系统中遇到的一些问题（业务强耦合、升级困难、多语言支持成本高等），越来越多的公司开始关注 ServiceMesh，并在生产环境进行 ServiceMesh 的实践。通过学习实践分享，了解了业界 ServiceMesh 具体的实现，以及遇到的问题。学习后发现，大家都比较认同 Istio 的设计方案，并在实践过程中参考 Istio 结合应用场景进行实践。根据场景的不同，具体实现各不相同。一般可以分为两类，一类以面向公司内部业务为主，例如：华为 Mesher；一类面向外部用户，以云平台的形式对外提供服务，例如阿里云 ServiceMesh。

阿里云 ServiceMesh 主要是面向外部用户，以 Istio 基础同，对 Istio 实现改动较少，主要侧重于结合阿里云的已有服务对 Istio 进行一些扩展与加强，例如整合了已有的智能流量路由、金丝雀发布、流量镜像等功能；另一方面主要是修改网络策略支持多区域多集群以及混合部署模式。腾讯云提供的 ServiceMesh 与阿里云思路大体相同，都是在提供 ServiceMesh 基础功能后，融合云平台已有功能，为用户提供更丰富、更多功能。

另一类是面向公司内部业务的实践，借鉴 Istio 的设计理念，采用自研+开源的方式，来更好的满足业务的需求。主要是因为在公司内部业务才是核心，新技术的推广是一个长期的过程，需要不停的慢慢演进。在新技术的推广过程中，不能影响到业务的稳写，因些 ServiceMesh 的实践需要重点考虑已有业务系统兼容与稳定，在已有

业务系统与新技术之间找到一个平衡点。例如：华为与阿里都是在充分考虑已有微服务框架(华为 ServiceComb,阿里 Dubbo)的基础上实现的 ServiceMesh。阿里 DubboMesh，在 Istio 的基础上结合业务情况，做了以下优化：1.Envoy 增加了 Dubbo 协议的解析（已合入主版本），2.Pilot 完成与内部服务注册发现服务 Nacos 的对接，3.将 pilot 功能下沉到 sidecar，在 ServiceMesh 全网化后，再将 pilot 上浮到原来位置，对过该方式来更好的推动 ServiceMesh 实践。华为 Mesher: 利用微服框架中的核心代理，用 go 语言实现了自研的 sidecar，在控制平台增加增加一层抽象接口，用来对接现存的服务治理及配置管理功能，最终形成了 SDK+sidecar 混合模式的 ServiceMesh，可以运行在任意的机器环境上。而对于 FreeWheel 其技术栈是 Golang 和 Kubernetes，特别适合实践 ServiceMesh，选择 Istio，只是在原来的 Istio 基础上整合了原有的自定义认证及授权机制，并取得了良好的实践效果。

通过业界公司的分享，大家对待新技术都是以一种务实的思想对待，都是为了解决业务遇到的一些痛点。具体实践中按照架构分层合理、下沉通用能力、结合具务现状，设计出满足内部业务场景的实现。在我们的实践中，也可结合行内的具体的情况，分析找出最好的切入点，将 ServiceMesh 最终在行内落地。

通过分享提供的测试数据中，也有一些与直观理解不一样的东西。例如：引入 sidecar 理论上增加了链路请求的跳数，但在实际测试过程中，并没有明显的导致性能的降低和资源的开销。