面向多租户的微服务架构实践

洪增林

字节跳动 / 资深架构师

关于我

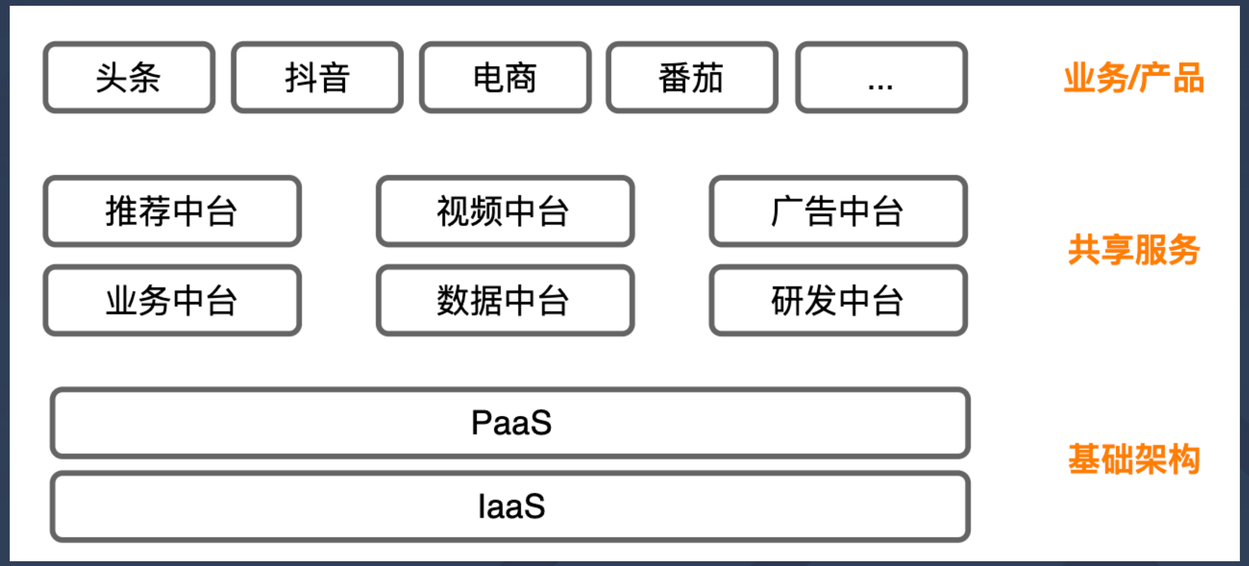
- 洪增林 / 字节跳动 架构师
- 14年从浙江大学硕士毕业，从业 8 年，致力于业务技术架构方向
- 19 年加入字节，现负责技术中台架构解决方案团队，先后推进多租户、单元化、数据访问保护和业务上云等技术架构演进

大纲

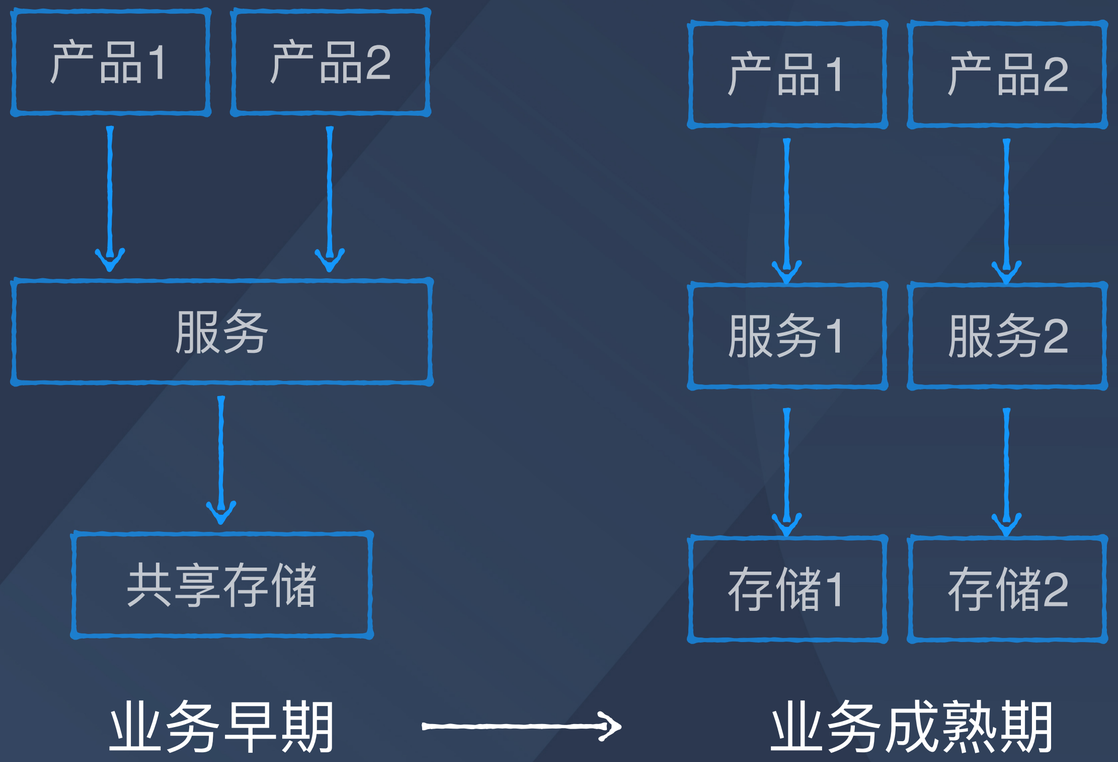
- 多租户的业务驱动力
- 面向多租户微服务架构
  - 多租户解决方案
  - 全链路流量身份票据
  - 计算层流量调度管理
  - 存储层隔离管理
  - 业务场景实践
- 后续的演进思考

业务形态


今日头条
抖音
轻颜相机
西瓜视频
懂车帝
剪映
番茄小说
飞书
火山引擎
...

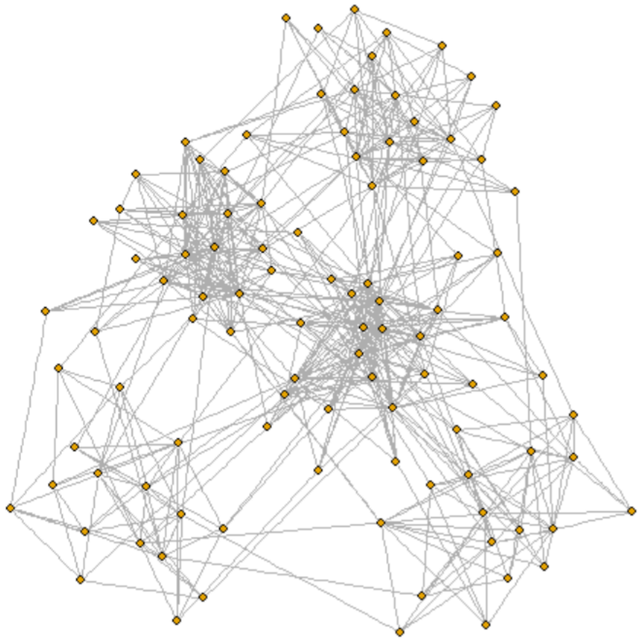

头条
抖音
电商
番茄
...
业务/产品
推荐中台
视频中台
广告中台
业务中台
数据中台
研发中台
共享服务
PaaS
IaaS
基础架构

以一个共享服务视角来看

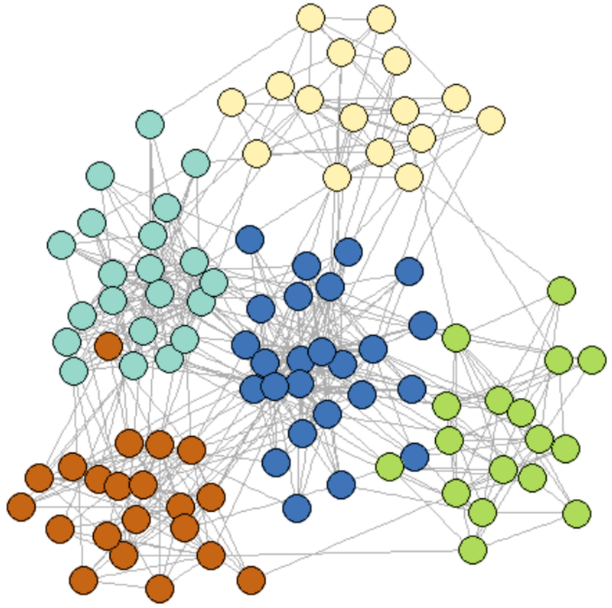


- 共享模式的问题
  - 不同租户共享一套服务，存在 Noisy Neighbor 问题
  - 数据合规和隔离需求
  - 分场景的隔离要求，比如Sandbox、压测环境、Chaos 环境等，一种泛化的租户

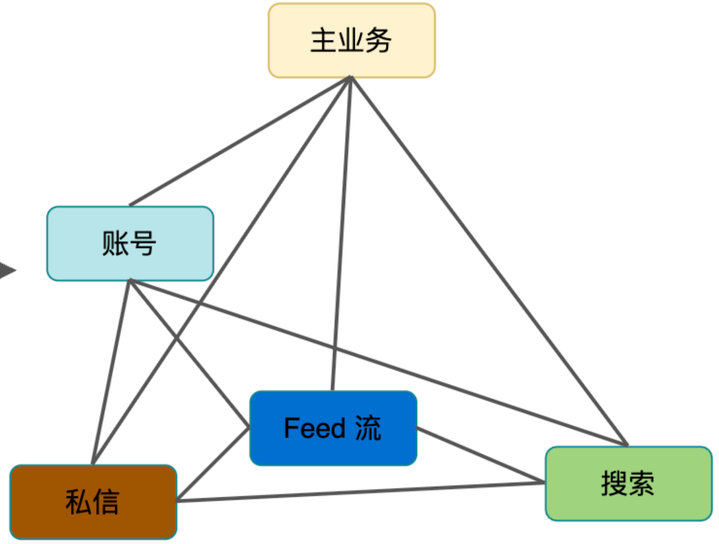
微服务调用拓扑



微服务视角服务调用关系



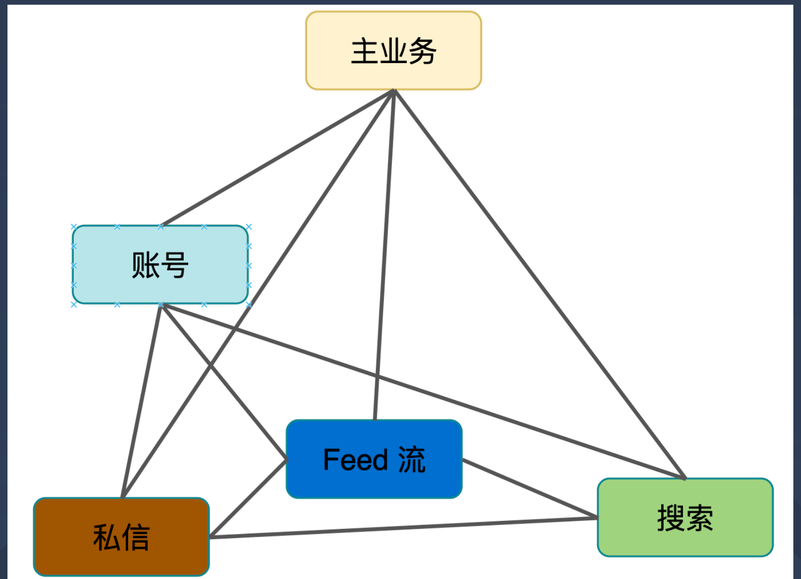
考虑服务域元信息的服务调用关系



服务域视角的调用关系
(图示关系仅做 Demo)

服务域

- 复杂微服务网络的治理不能仅关注单个微服务
- 服务域：一个内聚的微服务结合，对外提供能力
- 服务域下的微服务需要整体考虑对上游提供多租户治理能力
- 在字节跳动一个服务域会包含上千个微服务



服务域视角的调用关系
（图示关系仅做 Demo）

微服务下多租户挑战

- 服务域内包含大量微服务，上下游的服务域对于租户的拆分维度可能不同
- 微服务链路长，中间包含异步链路
- 共享服务对接租户初期倾向于用共享资源，后期面临拆分需求
- 租户数据的隔离按照不同的场景，需要支持 db/table/row 级别隔离

大纲

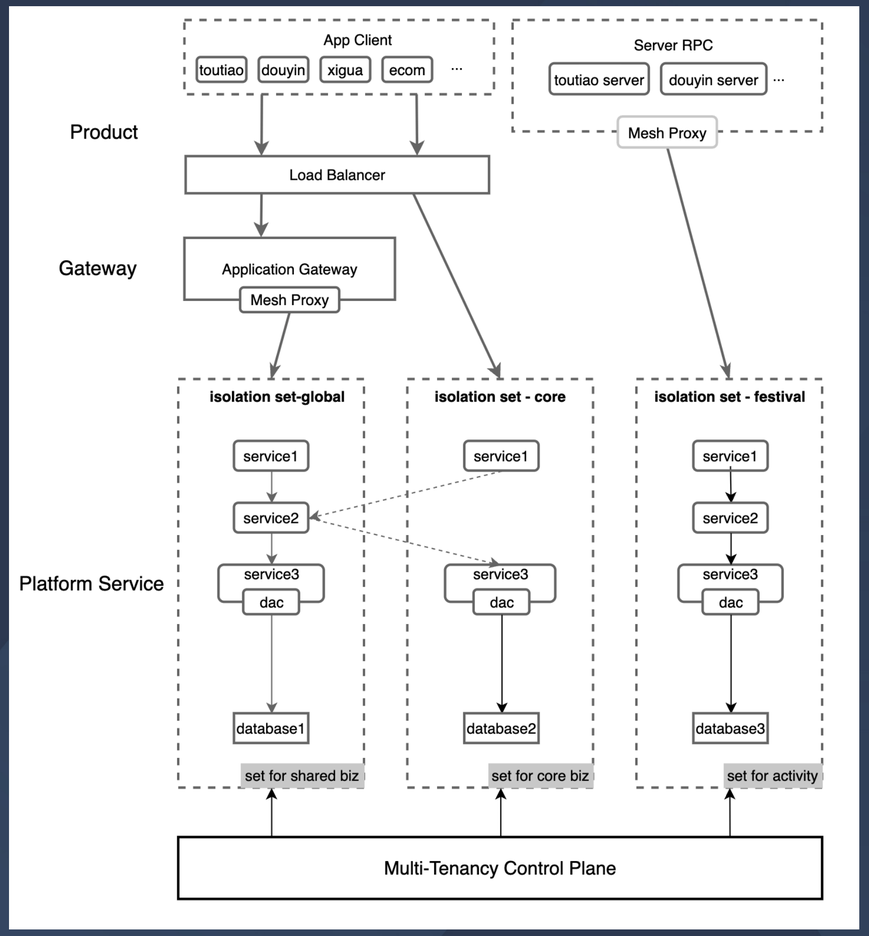
- 多租户的业务驱动力
- 面向多租户微服务架构
  - 多租户解决方案
  - 全链路流量身份票据
  - 计算层流量调度管理
  - 存储层隔离管理
  - 业务场景实践
- 后续的演进思考

我们面对的本质问题

- 容器化和微服务已经在业务中广泛落地，微服务的规模和链路越来越复杂
- 从租户隔离、数据保护、多机房部署等业务要求下，我们需要重新思考更精细的微服务治理手段
- 多租户是一个切入的场景，本质问题就是在复杂微服务架构下的流量和数据治理问题

在微服务架构下怎么解

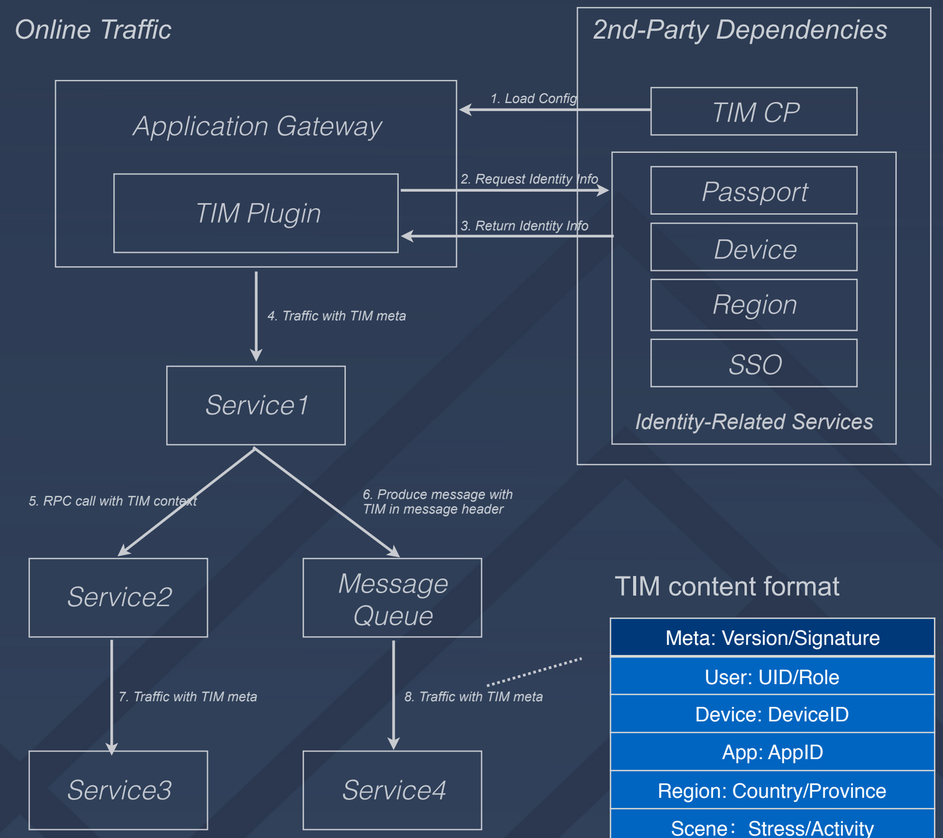
多租户解决方案



- Key1 – 流量身份标识，入口注入，全链路透传
- Key2 – 计算流量调度，不同租户的流量路由–> 隔离单元
- Key3 – 流量隔离保证，单元内部流量闭环，支持逃逸
- Key4 – 存储访问管理，不同单元的存储按需隔离，支持动态迁移

全链路流量身份票据

- TIM(Traffic Identity Mark)
- TIM 票据 – 基础身份信息 + 业务扩展信息
- 网关层注入，全链路透传
  - RPC – 框架/Mesh 支持，注入 RPC request header
  - MQ – 中间件支持，注入 Message header
- 应用 – 流量治理、数据访问控制、应用鉴权

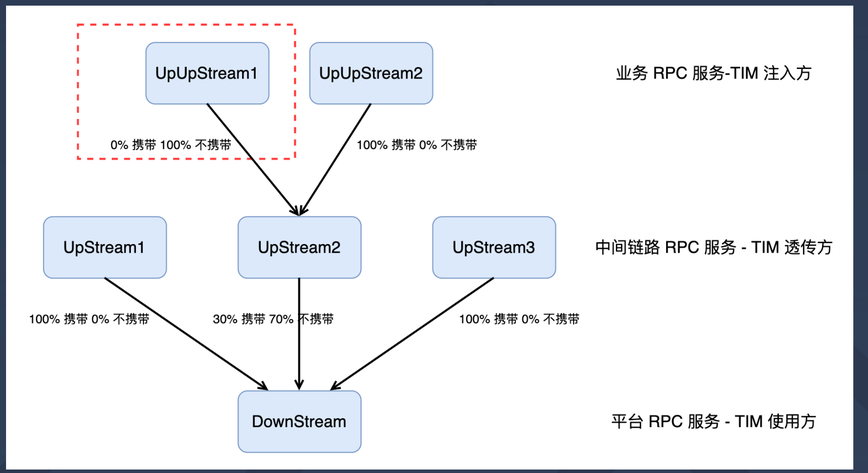


TIM content format

Meta: Version/Signature
User: UID/Role
Device: DeviceID
App: AppID
Region: Country/Province
Scene：Stress/Activity
Extended: Business Infos

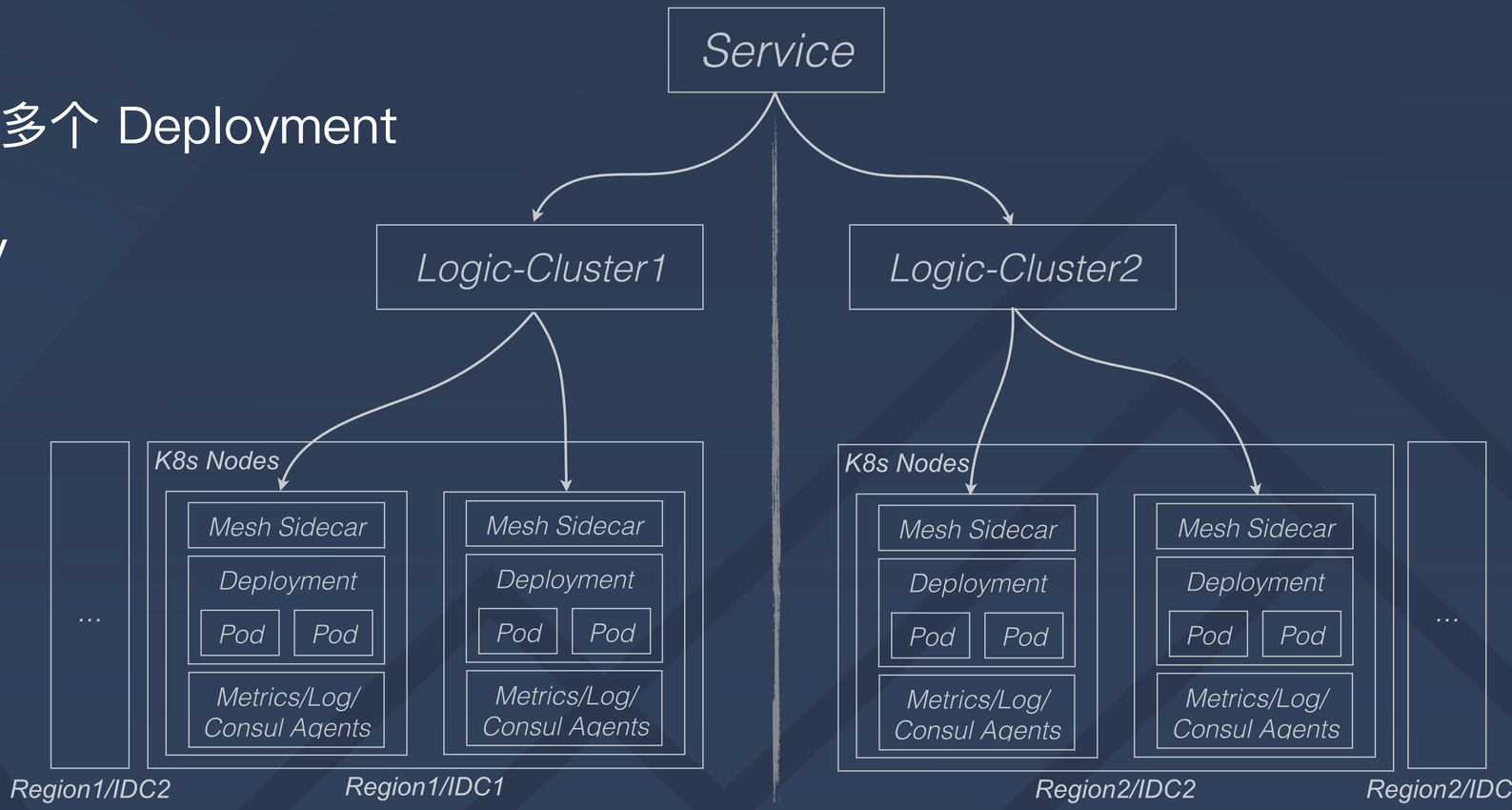
TIM 落地经验

- 统一注入 – 推荐在 AGW 层
- 可传递性 – 框架能力覆盖，KiteX/Mesh/MQ etc…
- 票据不丢 – 票据丢失/定位，解决断点识别
- 票据可信 – 保证票据不可伪造篡改
  - 注入方身份识别，基于容器粒度 ZTI Token
  - 非对称加签/验签
- 票据大小 – 注意检查 http/rpc header size 限制



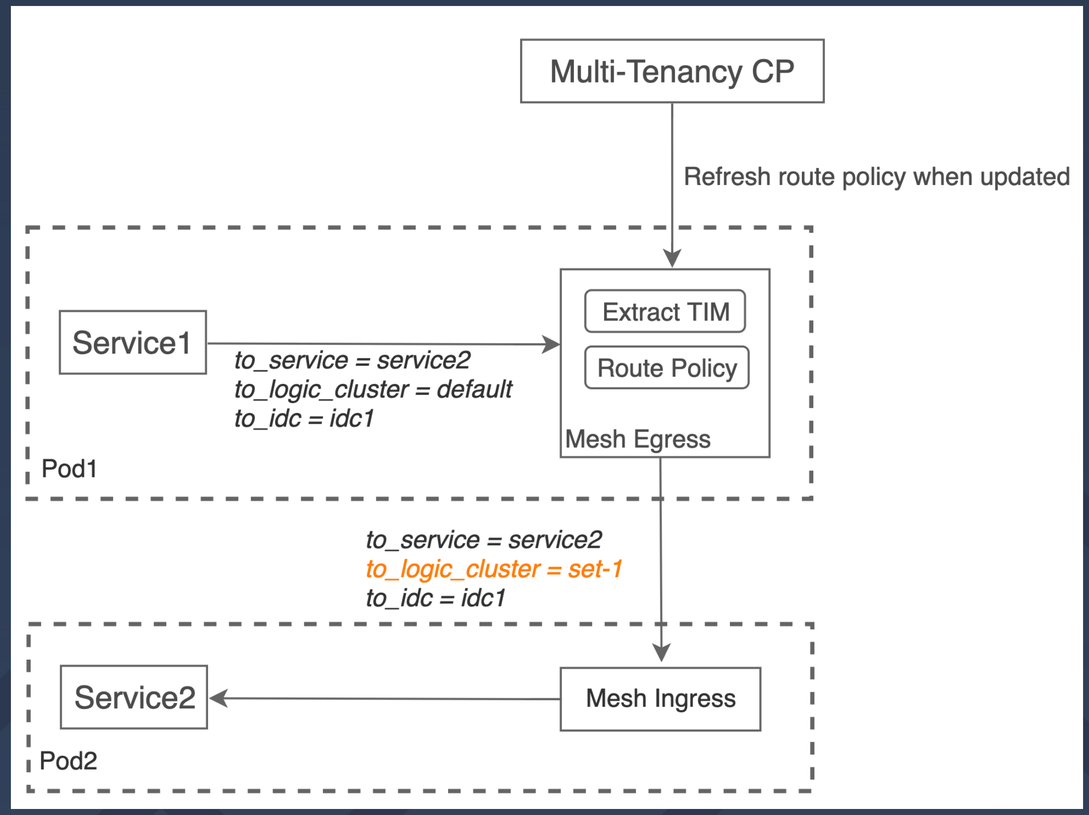
从一个微服务视角看

- Service – 单个微服务
- Cluster – 服务下的逻辑集群，对应多个 Deployment
- 绝大部分服务都启用了 Mesh Proxy



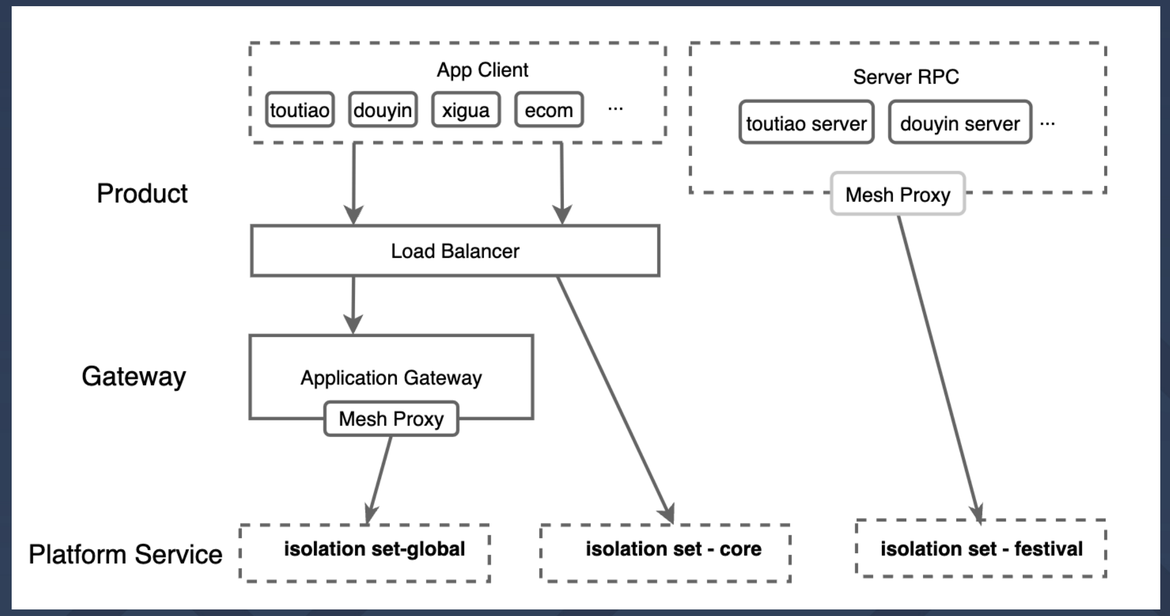
Mesh 流量策略路由

- 流量路由基本要求是对业务透明
- 租户路由策略下发到 Mesh Proxy，流量 Egress 中动态计算目标逻辑集群
- 路由规则 = TIM 变量 + 路由策略表达式



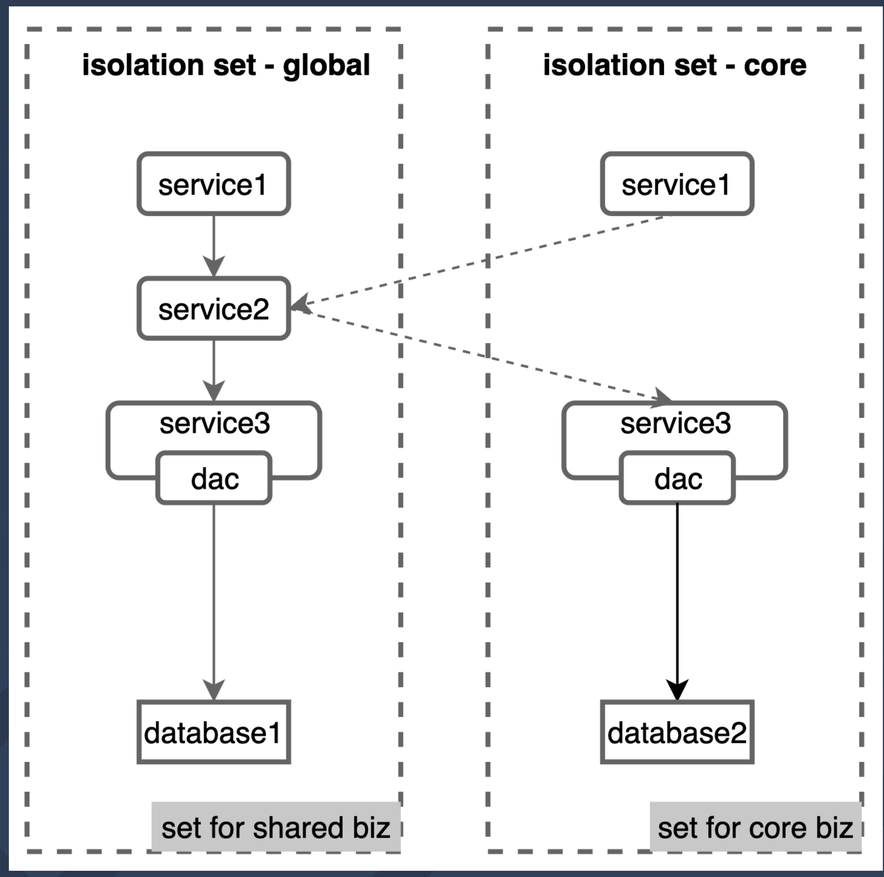
服务域入流路由

- RPC – MeshProxy 策略路由
- AGW – MeshProxy 策略路由
- LB – LB lua 插件策略路由
  - 能力和 MeshProxy 保持一致



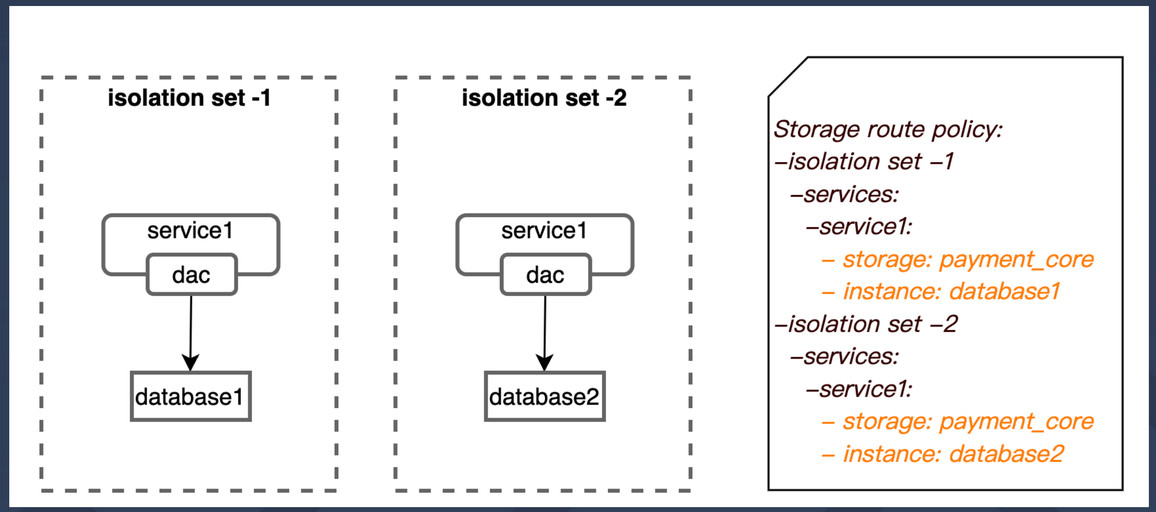
域内流量路由

- 流量优先在单元内闭环
- 支持单元内亲和性调度（调度兜底机制）
- 支持流量逃逸，不要求每个单元全量微服务
  - 单元有优先级，访问下游服务时优先访问到高优单元
  - 支持 RPC 调用直接指定单元



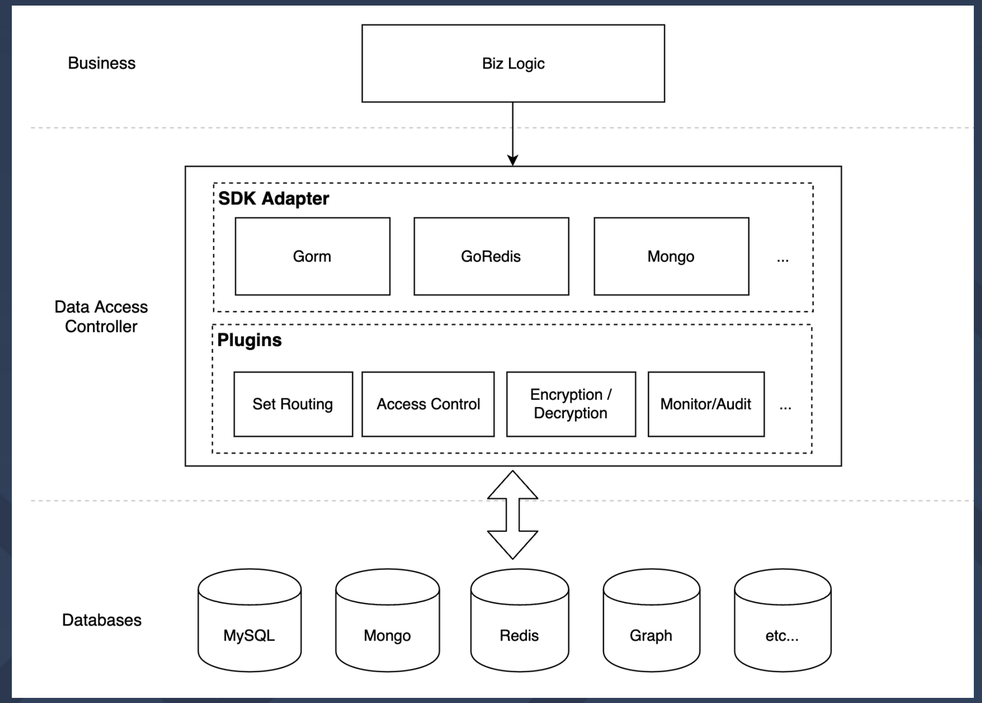
存储层流量路由

- 存储代号在不同单元指向不同实例
- 存储访问中间件层生效租户存储路由
- 跨单元访问拦截，兜底计算层流量调度 badcase



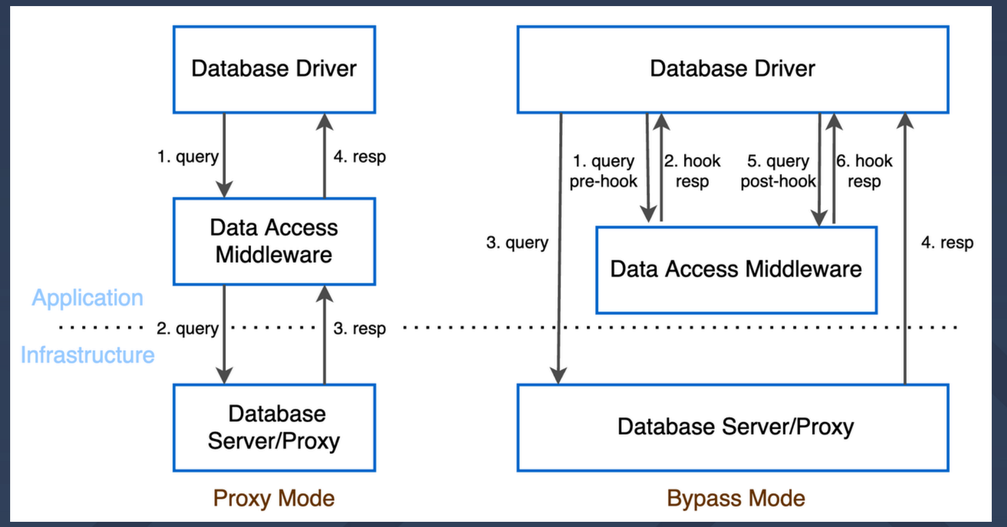
存储访问中间件

- 为异构存储提供一个统一的治理和管控中间件层
- 核心定位和能力：
  - 底层存储引擎适配
  - 租户请求路由
  - 数据访问管控
  - 租户数据在线拆分



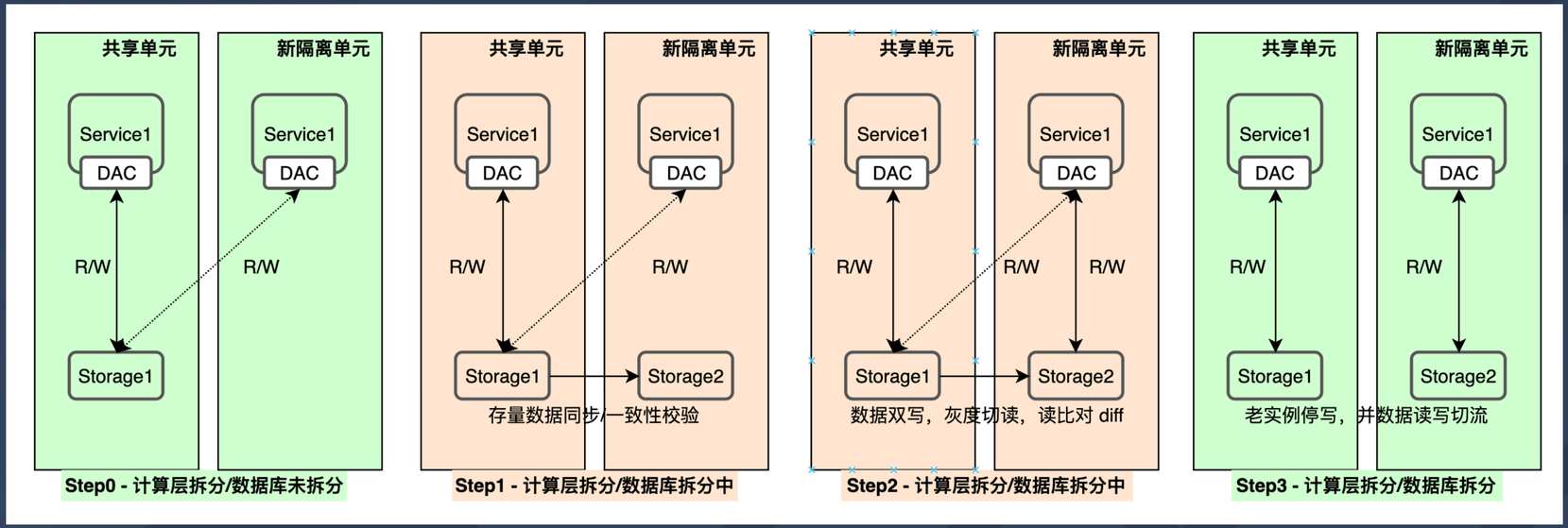
存储引擎对接

- 尽量减少对于业务研发的侵入
- 在 query hook 中动态改写 req/resp 完成租户数据路由
- 未授权的跨租户访问在 pre hook 中拦截，通过身份 + 访问规则来决策



延伸场景1–租户数据拆分

- 隔离性 vs. 成本的权衡，租户数据拆分不可避免
- 我们的目标：租户数据拆分标准化



平台化解决方案

- 一站式租户单元管理，计算/存储/路由规则/数据迁移...
- 关注标准化、效率和可靠性

解决了哪些问题

服务能力扩展

- 平台租户数量不断增长
- 单租户的流量持续增长

流量和数据隔离

- 租户间流量隔离
- 数据隔离（物理/逻辑）
- 数据访问管控

研发复杂度

- 按照单租户模型开发
- 不同租户版本和变更差异化

运维复杂度

- 垂直的隔离环境搭建
- 共享租户拆分到独享

User Case – 租户/场景流量隔离

- 多租户隔离：
  - 信息流、账号、IM、电商、支付 等
- 场景化隔离：
  - 封板核心链路隔离
  - 电商 Sandbox
  - UG 活动链路
  - etc.

这个事情是不是值得做

多租户架构投入考虑

服务复杂度

- 服务域内微服务规模大
- 微服务链路长
- 服务域内存储多
- 多 RD/SRE，需要标准

租户隔离场景

- 业务服务多个租户
- 租户间有隔离诉求
- 需要从共享拆分独立环境

垂直隔离链路

- 隔离压测链路
- 隔离活动链路

安全合规要求

- 租户的数据隔离有技术保证
- 跨租户数据访问有管控

大纲

- 多租户的业务驱动力
- 面向多租户微服务架构
  - 多租户解决方案
  - 全链路流量身份票据
  - 计算层流量调度管理
  - 存储层隔离管理
  - 业务场景实践
- 后续的演进思考

租户间调度到机房间调度

	场景	流量标识	计算层路由	存储层路由	数据同步	数据拆分	容灾管理
多租户	机房内租户隔离	需要业务标识	业务透明路由	• 业务投屏路由 • 跨单元拦截	租户数据隔离	租户拆分时需要	不存在租户间容灾
单元化	机房间的容灾	需要业务标识	业务透明路由	• 业务投屏路由 • 跨单元拦截	容灾机房间同步	不需要	• 容灾切流 • 强一致场景切流禁写

多租户和单元化面向不同场景，原子能力有很多重合

Key Takeaways

- 在微服务架构下构建面向多租户解决方案关注的核心问题
- 复杂微服务架构下的流量和数据治理思路
- 面向多租户的微服务架构的必要性、投入和收益
- 字节跳动在面向多租户场景的实践
- 后续业务架构的演进和投入

ArchSummit 全球架构师峰会 10th

想一想，我该如何把这些技术应用在工作实践中?

THANKS

InfoQ