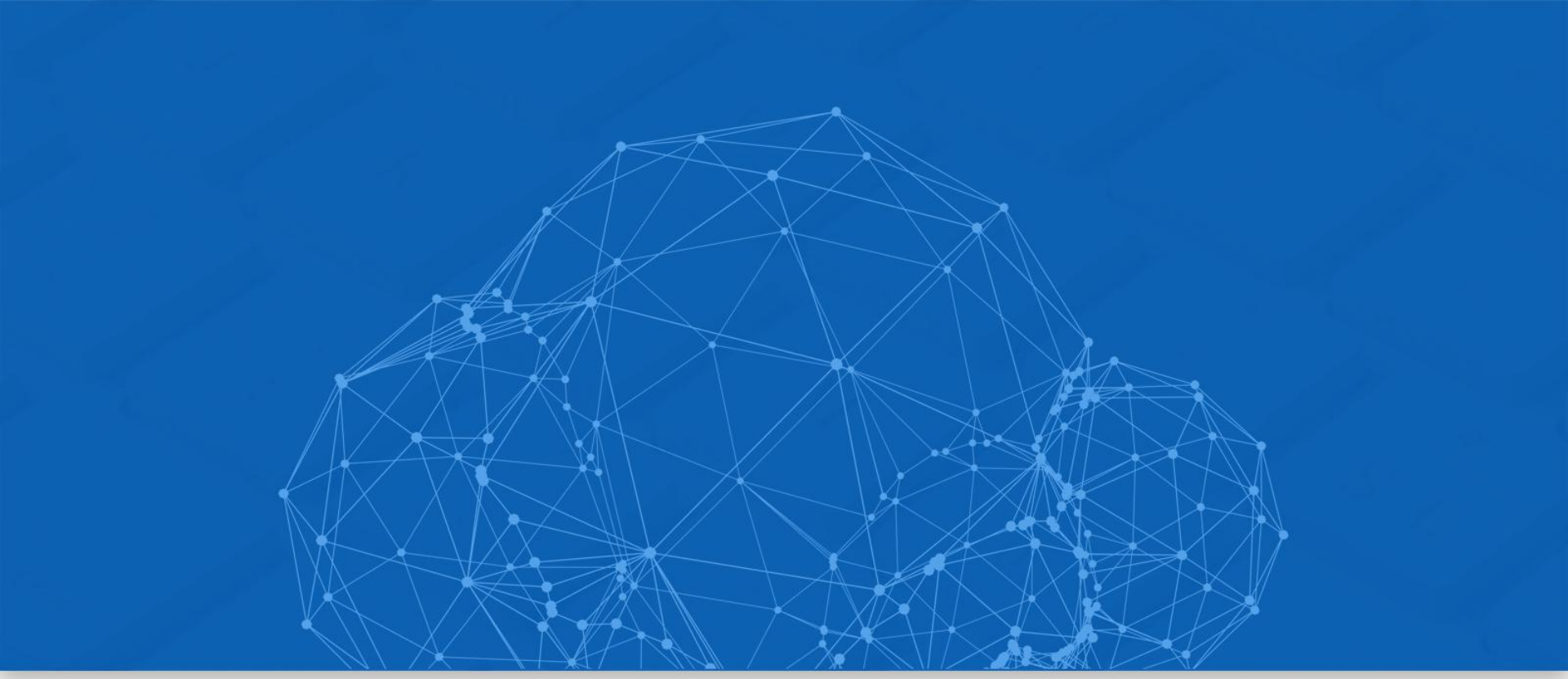
QCon全球软件开发大会

技术解析：构建高效开发容器云平台

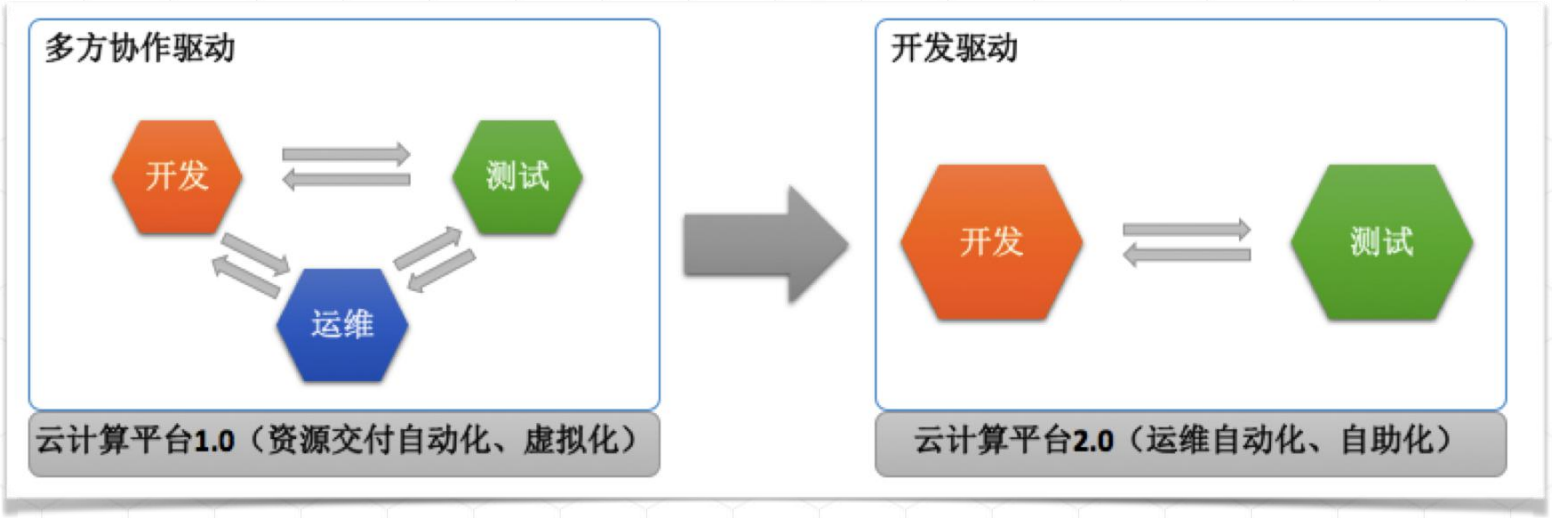
网易蜂巢 陈谔



从Docker容器技术到云平台

Docker容器带来的而改变

- 开发者角色
- 协作方式

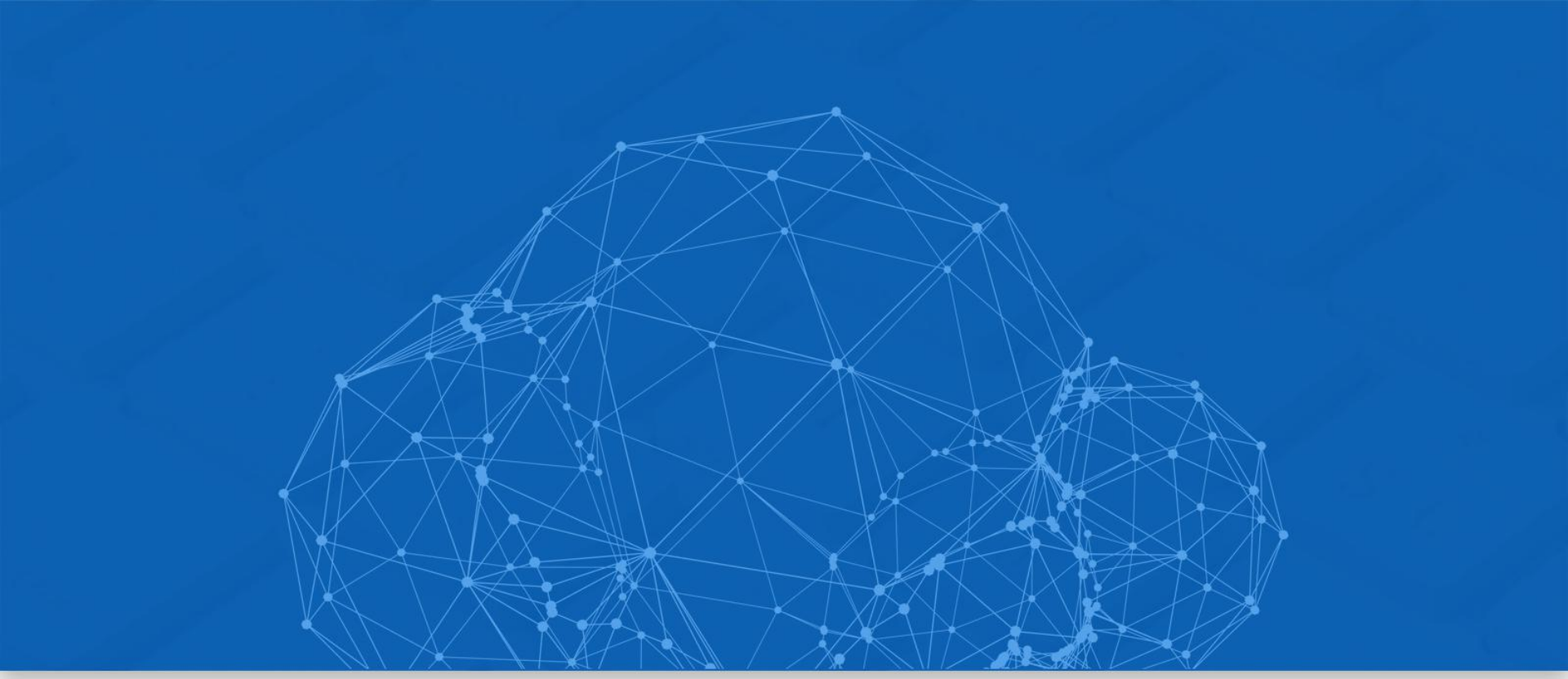


打造容器云的挑战

- 产品形态
- 技术栈的成熟度
  - Docker
  - kubernetes
  - OpenStack

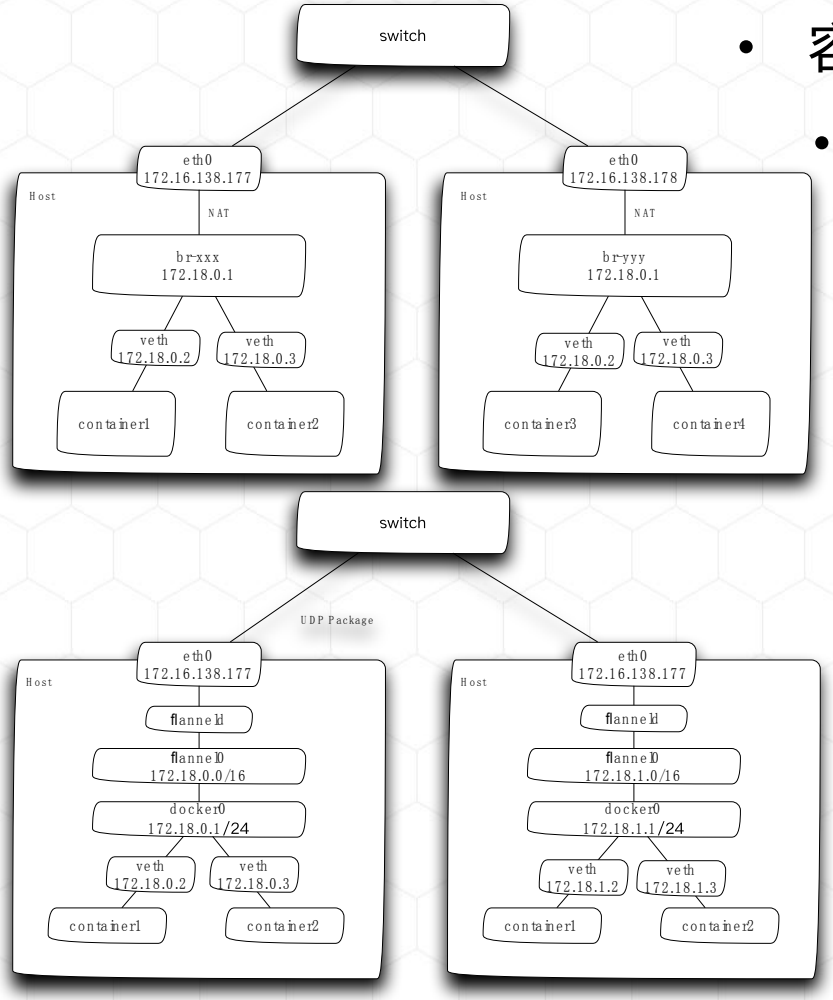
做一朵更好的云

—— 基于 Dev视角关注提升开发效率



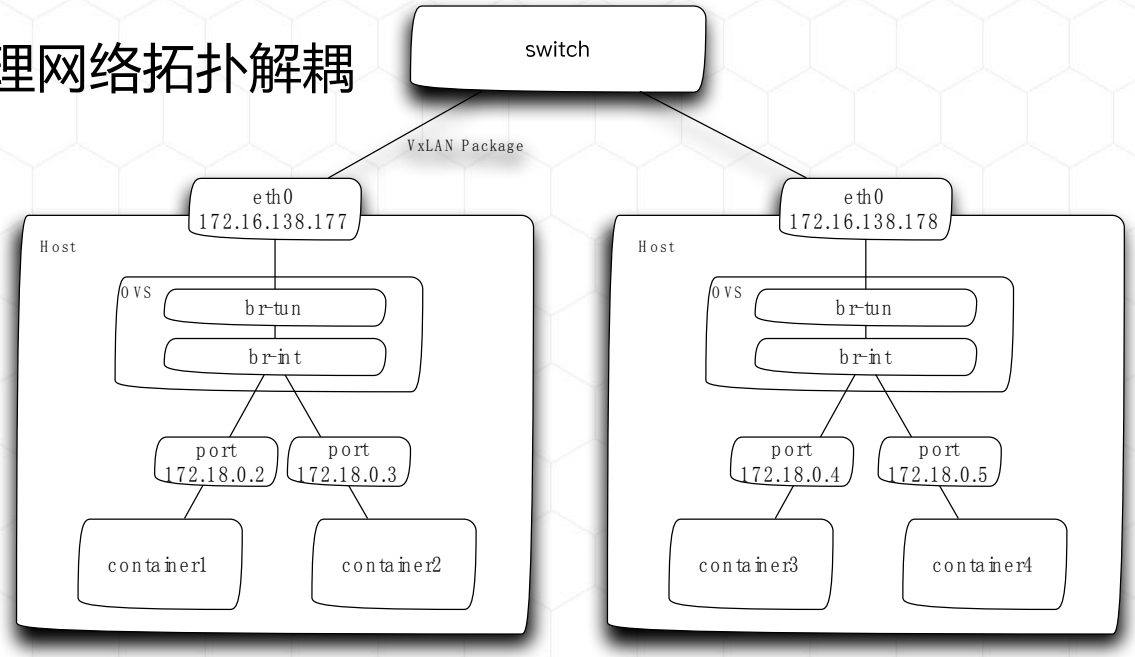
开发效率提升策略—消除系统复杂性

系统复杂性-容器网络



- 容器间互连
- 复杂：NAT、端口映射、层级网络
  - 遗留系统迁移的影响
    - 长连接状态问题
    - 基于 IP 注册的服务发现
  - 运维复杂度增加
    - 端口冲突
    - 内外IP 不一致
  - 不利于故障恢复
    - IP 变化

- 简单：虚拟化扁平二层网络
  - 遗留系统兼容性好
  - 控制 IP 分配
    - 利于故障恢复
    - 与物理网络拓扑解耦



蜂巢的解决方案

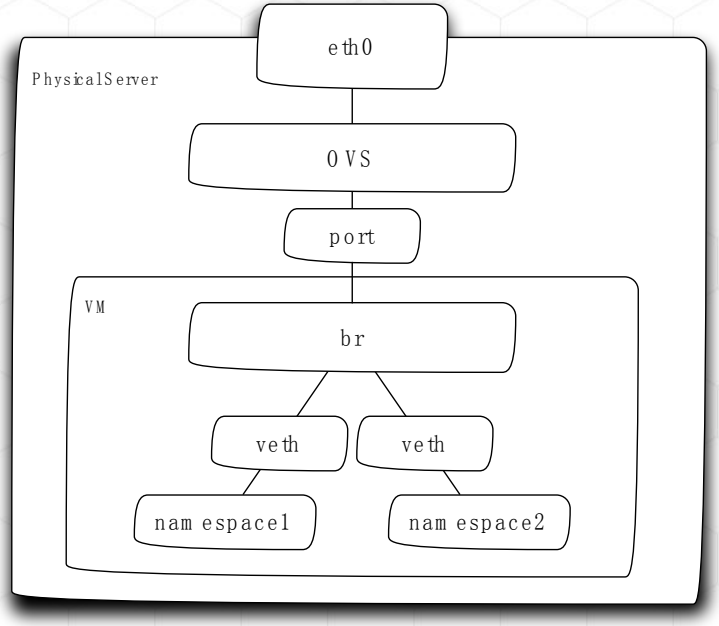
- VxLAN 网络，基于 Openstack Neutron
- 每租户一张独立私有二层网络
- 外网网卡直接挂载
- 私有网络 > 容器网络

网络性能的挑战

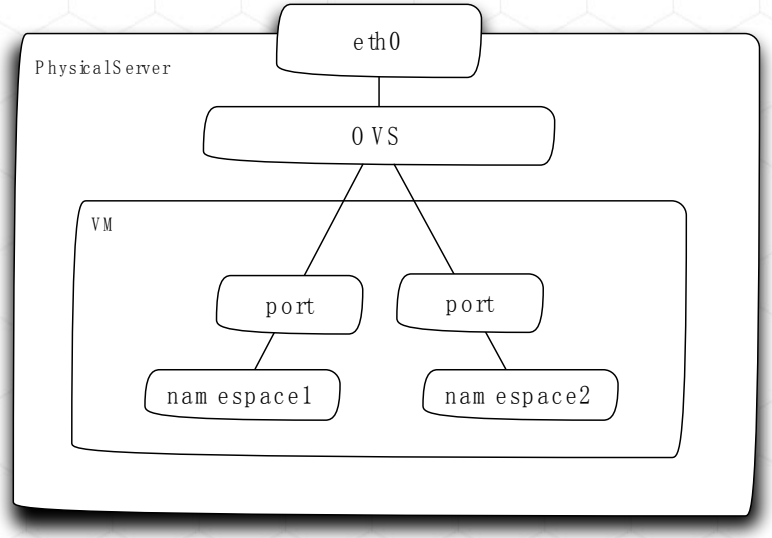
- VxLAN性能
  - L2 Population（避免 IP ）
  - RSS（利用多核）
  - GRO（接收端package 合并）
  - RCO（优化 checksum 开销）
  - 硬件 offload

架构优化

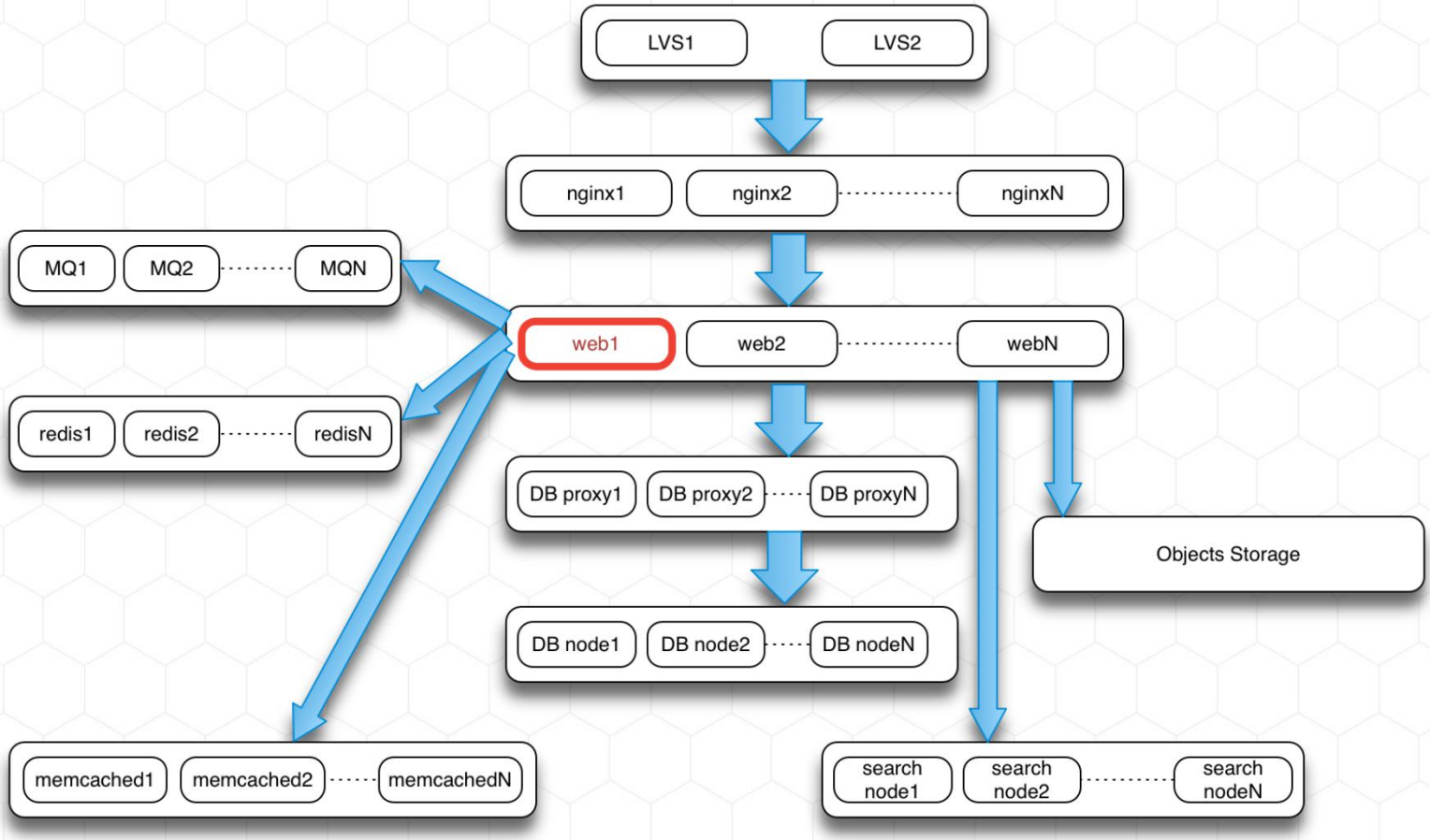
- 桥接、非受控



- Namespace 接入，受控

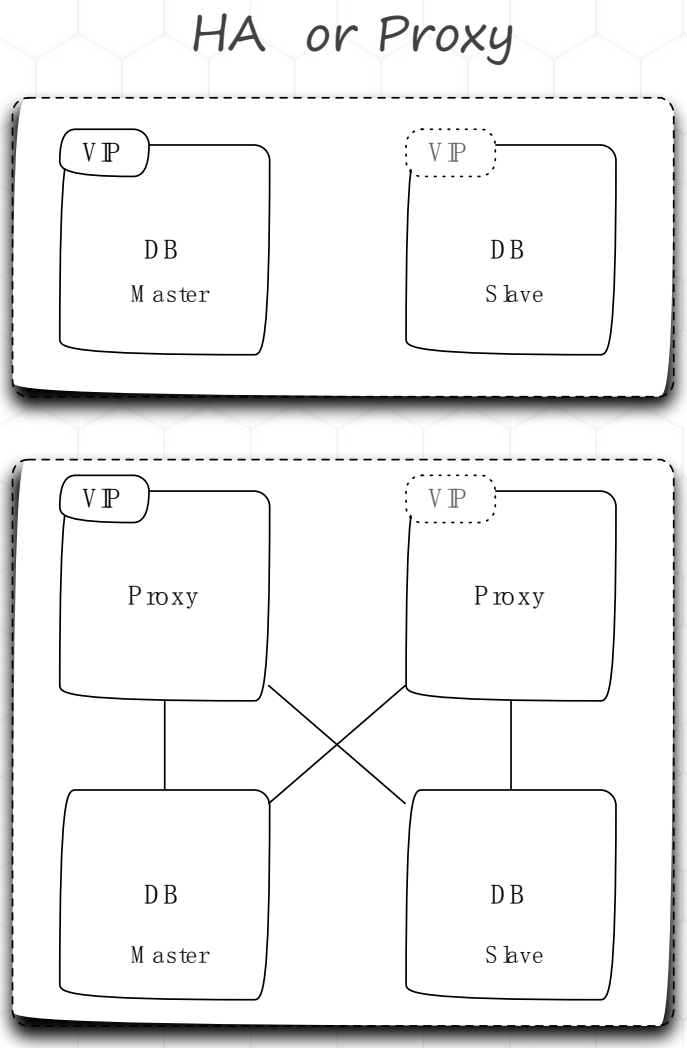


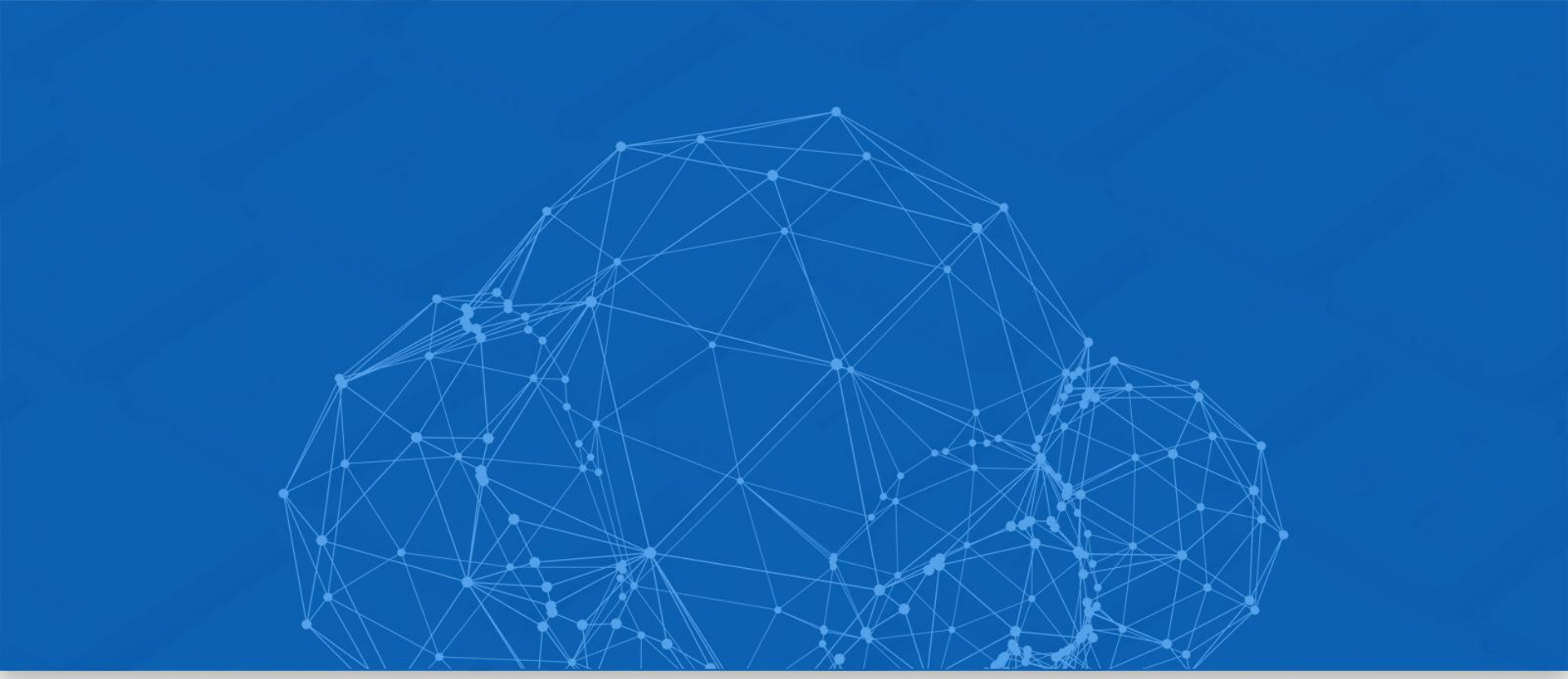
系统复杂性-架构


LVS1
LVS2
nginx1
nginx2
nginxN
MQ1
MQ2
MQN
web1
web2
webN
redis1
redis2
redisN
DB proxy1
DB proxy2
DB proxyN
Objects Storage
DB node1
DB node2
DB nodeN
memcached1
memcached2
memcachedN
search node1
search node2
search nodeN

基础服务云端化

- 数据库、消息队列、缓存等服务抽象为后端服务，对 Client 只暴露一个 URL
- 高可用、Scale 由后端服务在实现上保障，无需使用者介入





开发效率提升策略—DevOps

容器云该为 DevOps 做什么？

Build，Ship，Run

以容器作为资源的
交付单位

编排

容器与集群运维
管理自动化

诊断工具

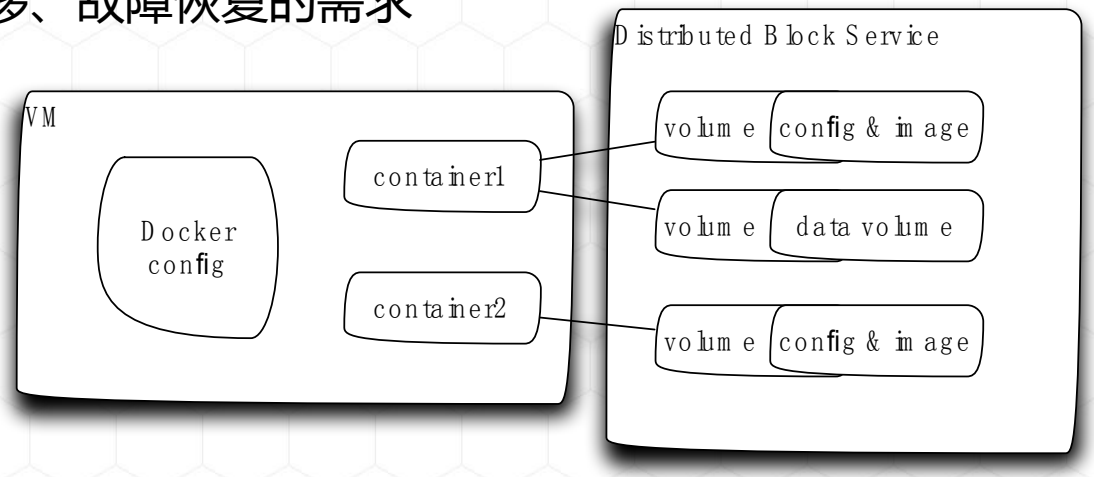
性能稳定性风险防
范故障快速定位

关于容器运行

- 保持状态
  - 网络、存储
- 是否基于Hypervisor
  - 安全隔离
  - 故障隔离

技术解析：本地盘 VS 远程盘

- 备份、停机、迁移、故障恢复的需求



- ***Docker 远程盘支持***
  - *Node 故障时将远程盘挂载到新 Node，Copy 容器配置信息到 Containers 目录，只有重启 Docker Daemon 才能加载配置启动容器。 解决方法：增加 reload 指令*
  - *Docker daemon 启动时可以通过—graph=指定docker运行时根目录， 但当一个node上运行多个容器时，所有容器的配置信息，文件系统相关数据，数据卷都存放在一个根目录下，导致容器无法独立迁移。解决方法：docker run 增加 container-home=dir将容器数据保存在dir目录*

- 容器迁移
  - 在一个node上挂载云硬盘到指定dir目录
  - 启动容器，设置container-home=dir将容器数据保存在dir目录
  - 当node宕机或需要迁移该容器时
  - 将该容器配置信息copy到新的node containers目录
  - 挂载云硬盘到新node dir目录
  - 执行docker reload 后docker ps –a可以看到容器
  - 执行docker start 就可以启动容器，迁移成功

关于容器编排

- 选择 kubernetes
- 有状态容器支持
- 多租户

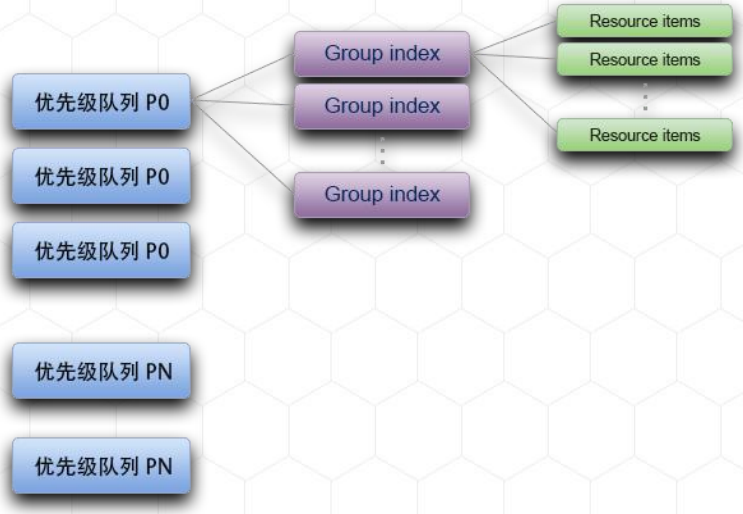
kubernetes存在的问题

- 多租户支持
  - namespace 只隔离 rc，pod 等资源，node 与存储、网络等是共享状态，实现真正的多租户隔离需要将所有资源隔离。不同租户不共享 node，每个租户的认证与授权独立（采用不同证书）。

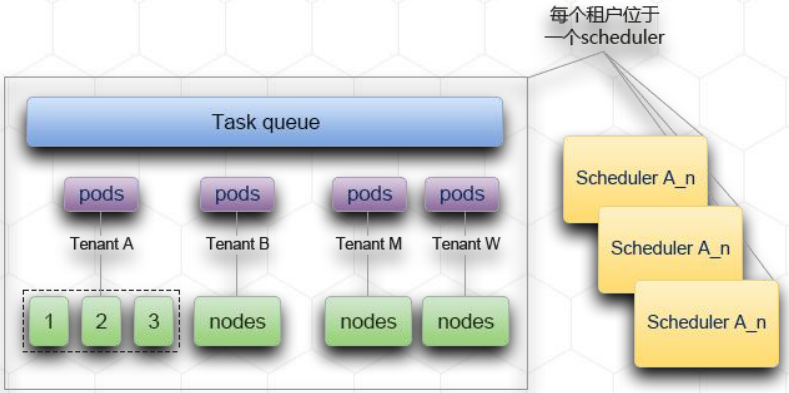
- 有状态支持
  - 原生 RC 面向无状态场景设计，当 Node 故障时会重建 Pod 导致状态数据丢失，需更改 kubernetes 的恢复策略进行支持。

性能

- 性能问题
  - 任务队列设计不合理，所有操作都在队列串行执行，scheduler、controller 等组件均存在此问题。改进为多优先级队列、deadline 机制，可有效解决这一问题。



- How to Scale
  - 关键在于扩展 etcd 集群与调度器，可通过LB 聚合多个 etcd 集群；按照调度器空闲情况将租户调度请求分散到不同调度器处理。

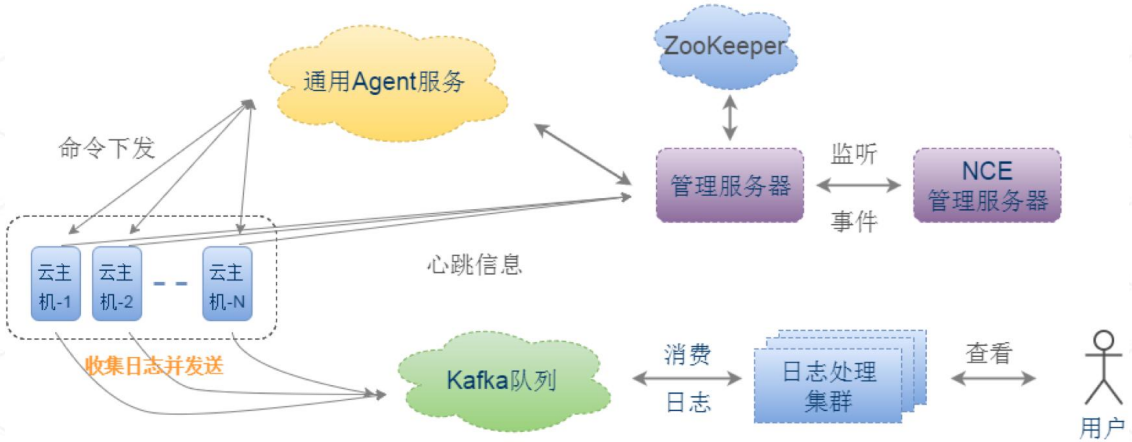


诊断工具

问题排查手段的缺失是影响业务可用率的最重要因素之一

统一日志服务

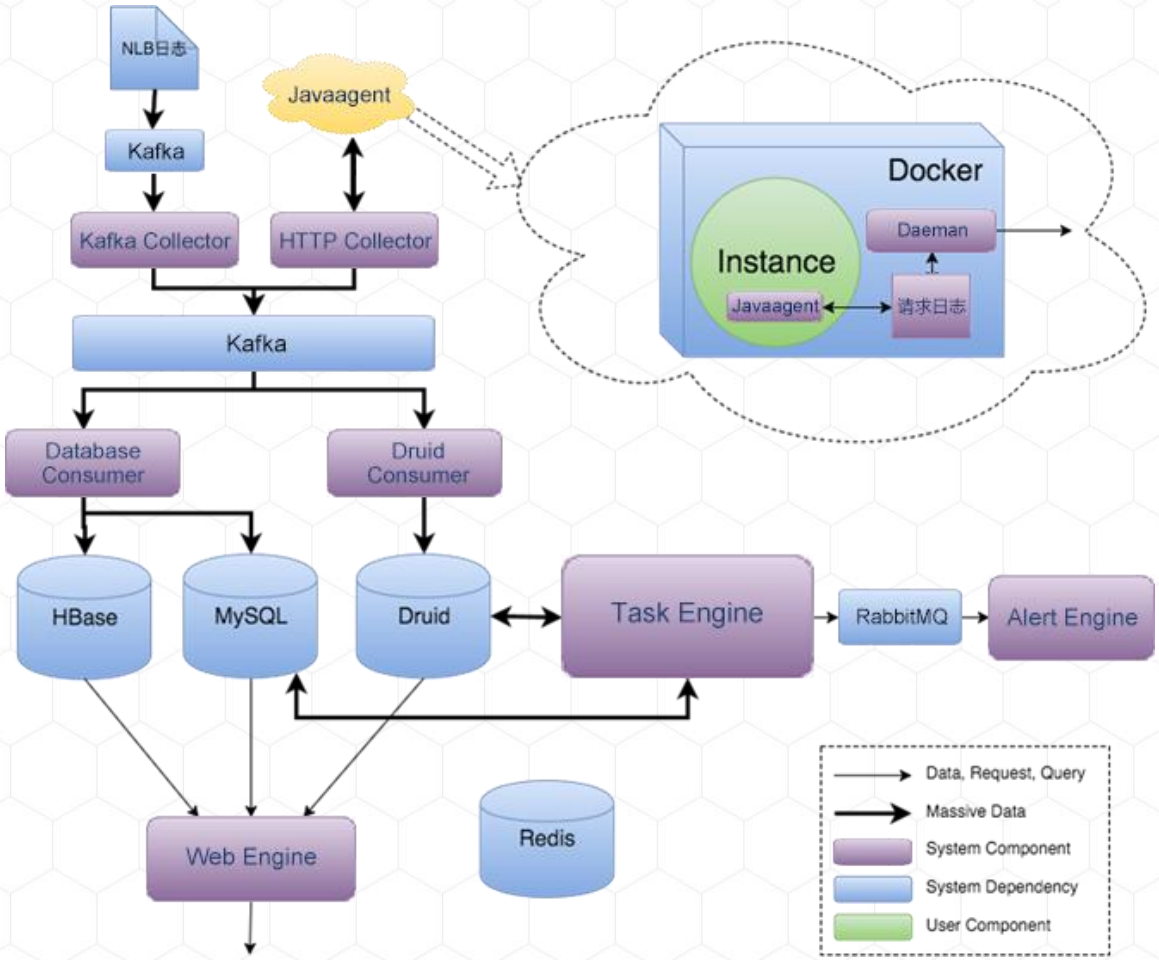
- 在多节点架构的业务系统下必须的服务，否则问题定位的速度将大打折扣
  - 典型场景：应用服务器集群问题排查需依次登录服务器
  - 典型场景：看不到其他服务模块的日志，排查问题时沟通过程复杂
- 在微服务架构下可进一步配合日志 trace 机制分析调用链
- 采用 rsyslog 收集日志降低Node 节点资源开销，采用 Kafka 支撑大吞吐量的同时保障可接受的日志收集延迟



性能管理

- 单纯的资源监控不利于性能问题的预判与诊断
  - 短暂的性能波动在常规监控方式下易被平均，性能监控至少要细化到服务 API 级别甚至细化到代码接口级别
  - 必须要与服务依赖、调用链相结合才能精确定位性能故障点，发生故障时调用链上的节点常规监控都会表现出故障现象，调用链跟踪配合日志服务能够快速精准得定位性能问题极大加速线上性能问题的解决
  - 数据库等基础服务的监控需要定制化，如慢查询统计、死锁记录、复制性能等

引入服务端 APM 解决细粒度性能分析



网易蜂巢

谢谢

扫一扫，关注我们