WOTD

51CTO

全球 World Of Tech 2017

2017年12月1日-2日 • 深圳中洲万豪酒店

软件开发技术峰会

DEVELOPMENT

WOTD

51CTO

微服务与容器技术

基于Docker的微博直播互动微服务架构

温情

新浪微博高级系统研发工程师

主要分享内容

01 背景与挑战

02 直播互动的微服务化架构

03 直播互动的弹性扩缩容

04 典型案例分享

Part.1
背景与挑战

01 背景 - 微博直播互动

基本需求：

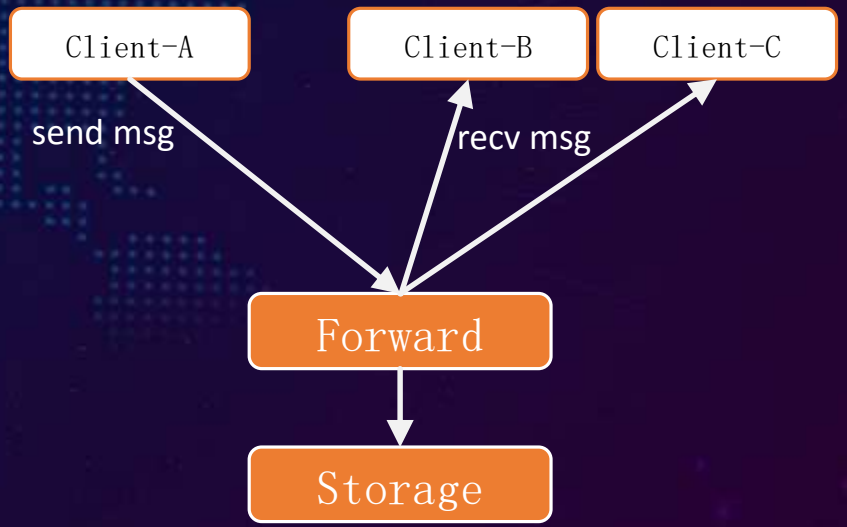
1. 消息的实时互动（评论，点赞，送礼等）
2. 支持消息回放
3. 支持三方接入（淘宝，一直播，咪咕，红豆等）
4. 房间人数无限大（支持百万在线）

01 直播互动 - 消息系统

基本功能：消息转发

特性：

1. 实时性：高
2. 可靠性：一般
3. 一致性：高



01 微博直播互动特点

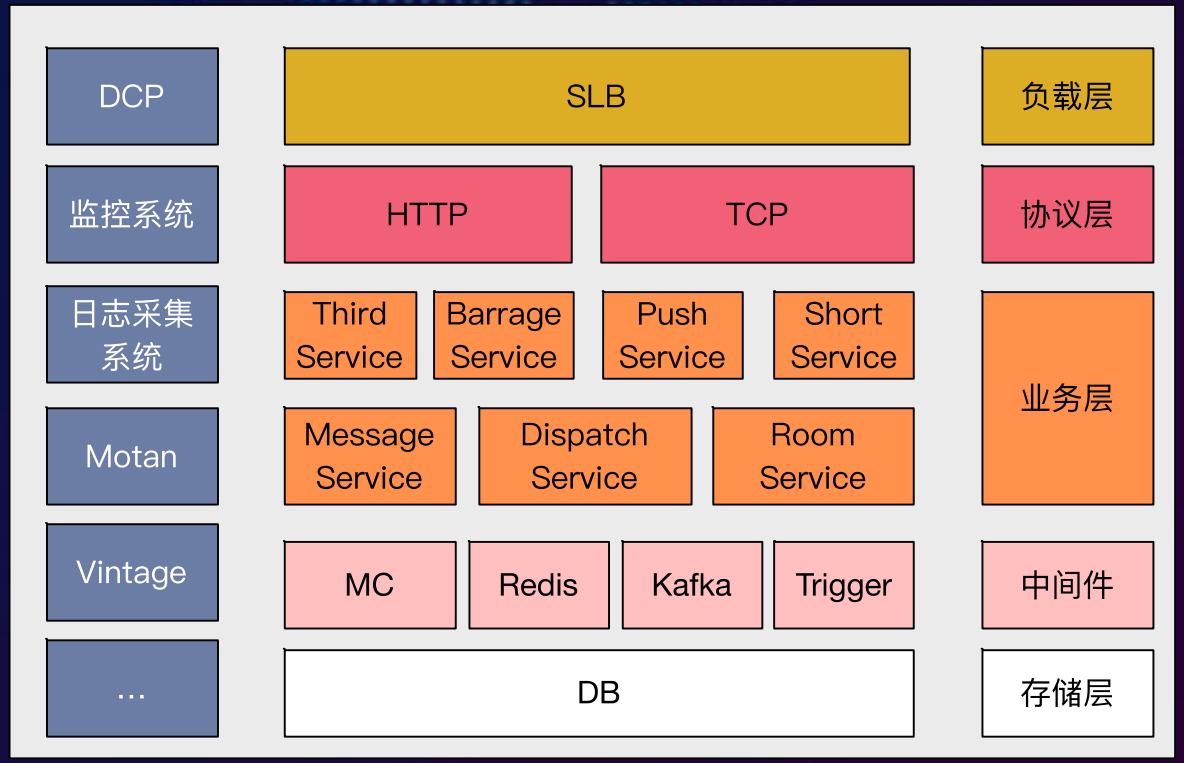
1. 微博特色的极端突发流量
2. 消息推送量大（每秒千万条消息推送）
3. 服务端持续压力（与短连服务对比）

01 挑战

1. 微博直播快速发展，如何快速响应需求
2. MAU 3.7亿的微博，如何应对全量Push引起的千倍流量激增
3. 直播流量的波峰波谷特性，如何实现成本最优化

Part.2
直播互动的微服务化架构

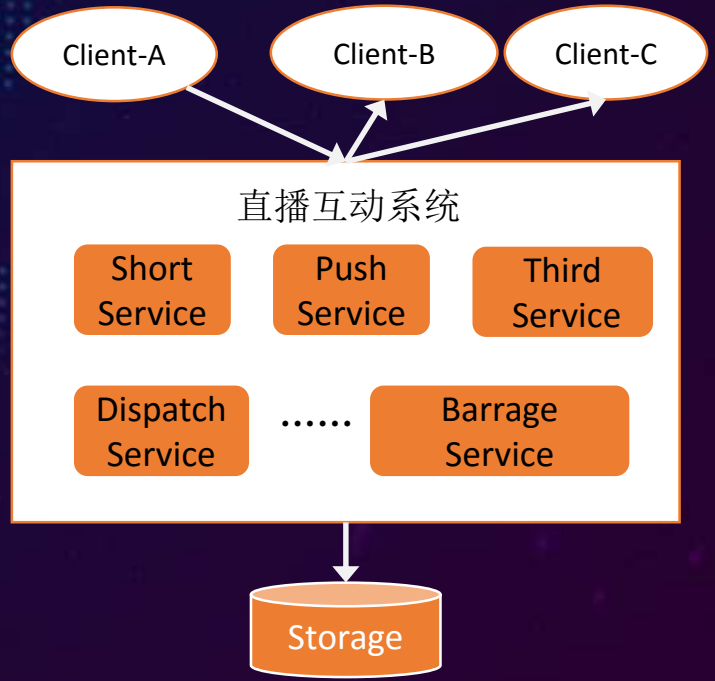
02 直播互动架构图


DCP
SLB
负载层
监控系统
HTTP
TCP
协议层
日志采集系统
Third Service
Barrage Service
Push Service
Short Service
业务层
Motan
Message Service
Dispatch Service
Room Service
Vintage
MC
Redis
Kafka
Trigger
中间件
...
DB
存储层

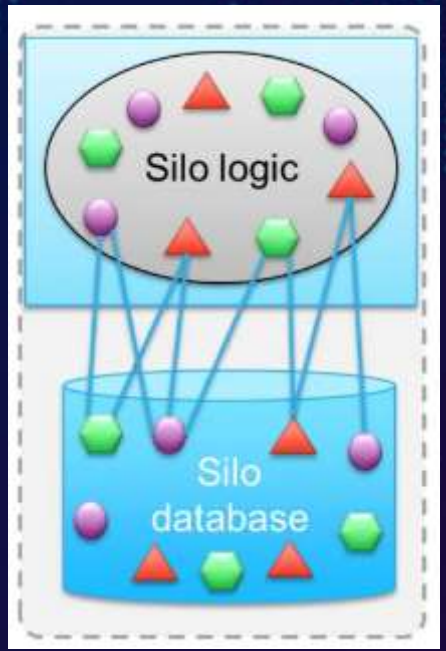
02 架构选型 – 单体式

面临的问题：

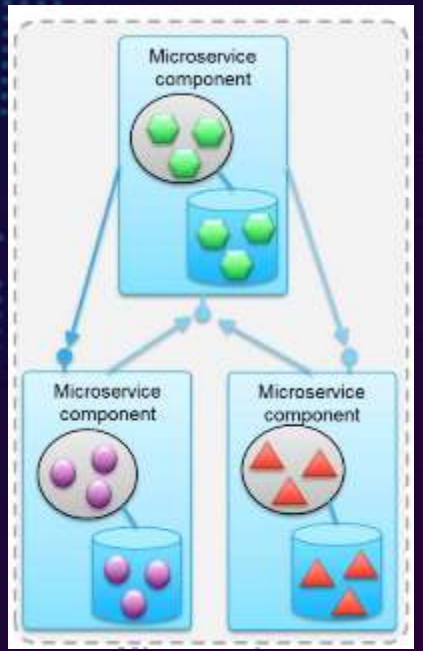
- 上线成本高，迭代速度慢
- 可伸缩性差
  1. DB/Redis连接数过多
  2. 扩展速度慢
- 鲁棒性差
  1. 任何一处的Bug，引起整体服务Crash



02 架构选型 – 单体式 vs 微服务



单体式

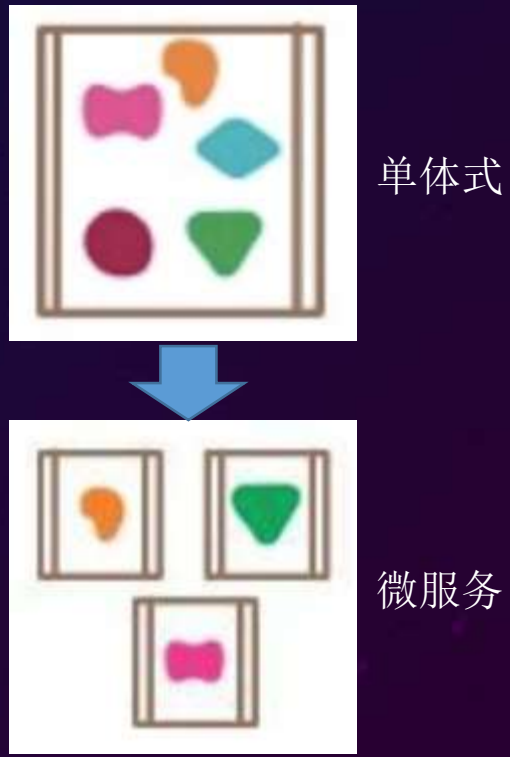


微服务

02 架构选型 – 微服务

微服务的好处：

1. 独立开发测试部署，提高迭代速度
2. 微服务依赖资源少，提高扩容速度
3. 扩容频繁的服务，移除对DB和Redis的直接依赖，解决连接数问题
4. 各个服务独立部署，避免无关联服务的Bug而引起整体服务Crash



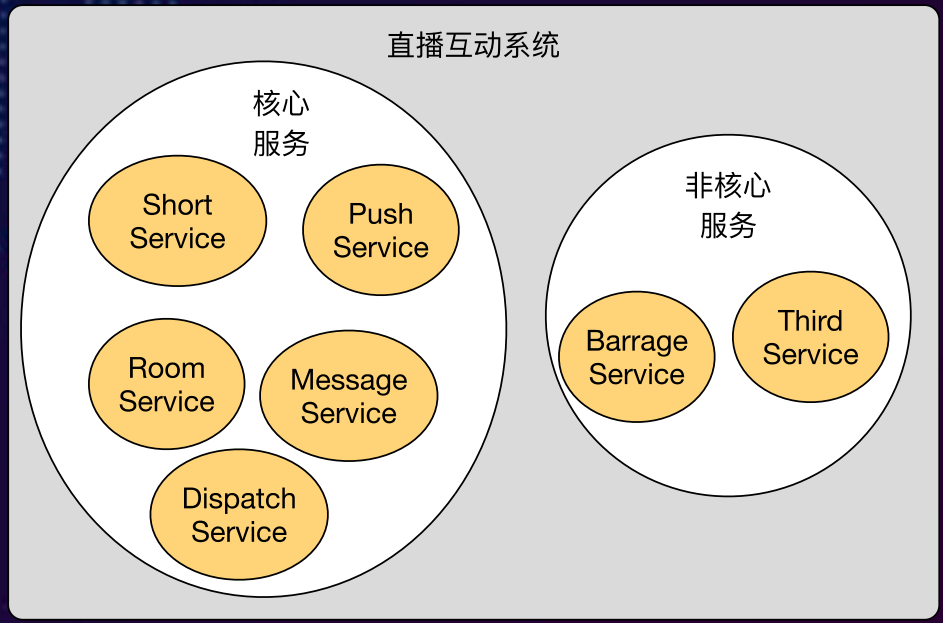
02 微服务 – 新问题

- 微服务如何合理拆分
- 微服务如何通信
- 微服务发现问题

02 微服务 – 服务拆分

核心与非核心服务拆分：

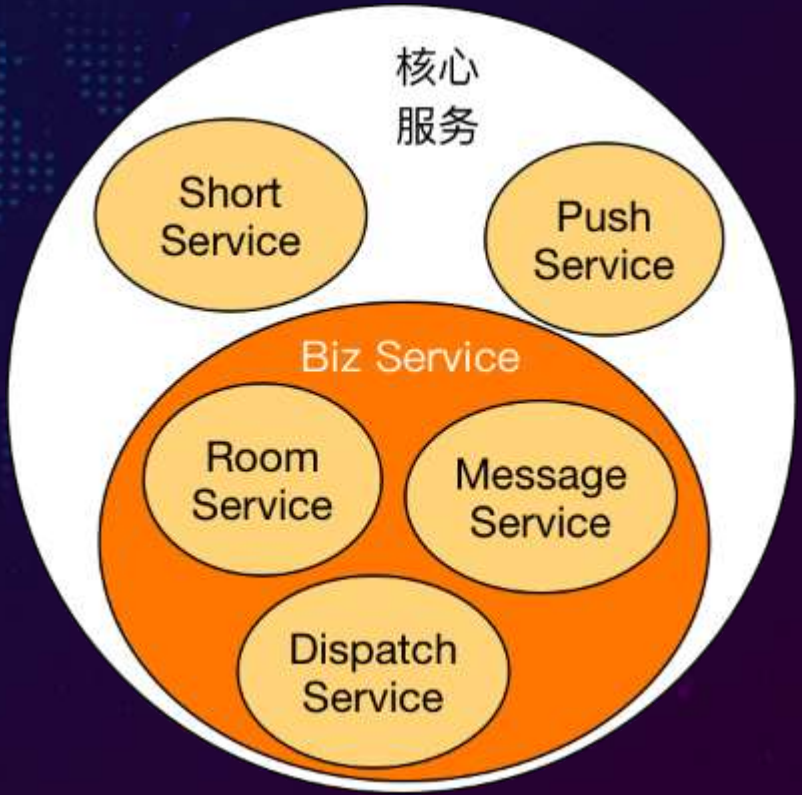
1. 服务隔离，避免非核心服务影响核心服务
2. 服务治理，可随时对非核心服务降级



02 微服务 – 服务拆分

核心服务拆分：

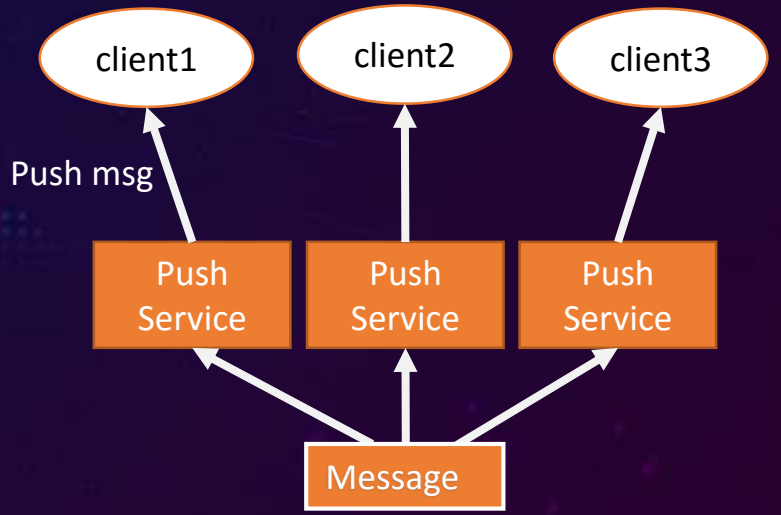
1. Short Service: Wesync 私有协议解析
2. Biz Service：房间，消息等业务处理
3. Push Service：维护与用户的长连，实时推送消息



02 微服务 – 服务拆分

Push Service 拆分的原因?

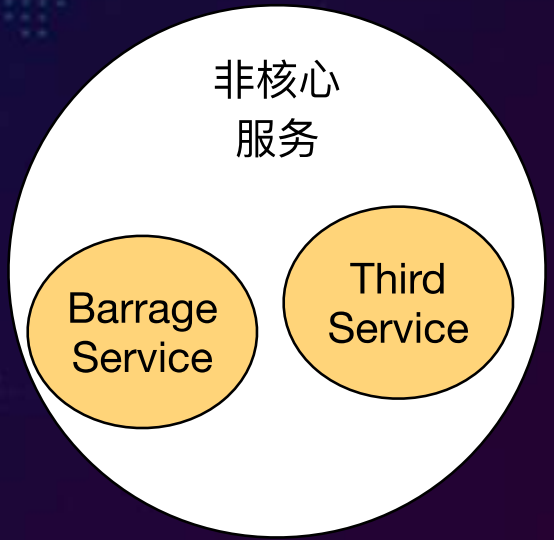
- Push Service 分析
  - 推送量大，扩容频率高
- Push Service 设计
  - 功能单一
  - 减少对资源的依赖，解决连接数问题



02 微服务 - 服务拆分

非核心服务拆分：

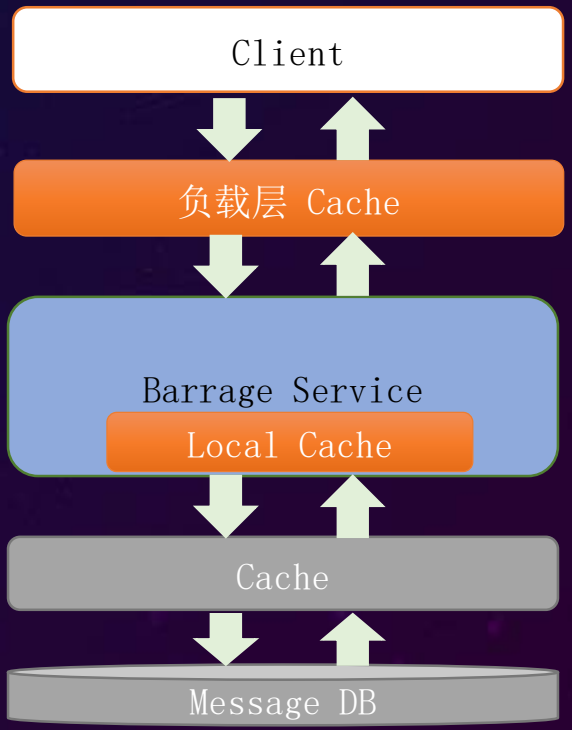
1. Barrage Service：消息回放服务，获取历史消息
2. Third Service：与其他三方平台接入服务



02 微服务 - 服务拆分

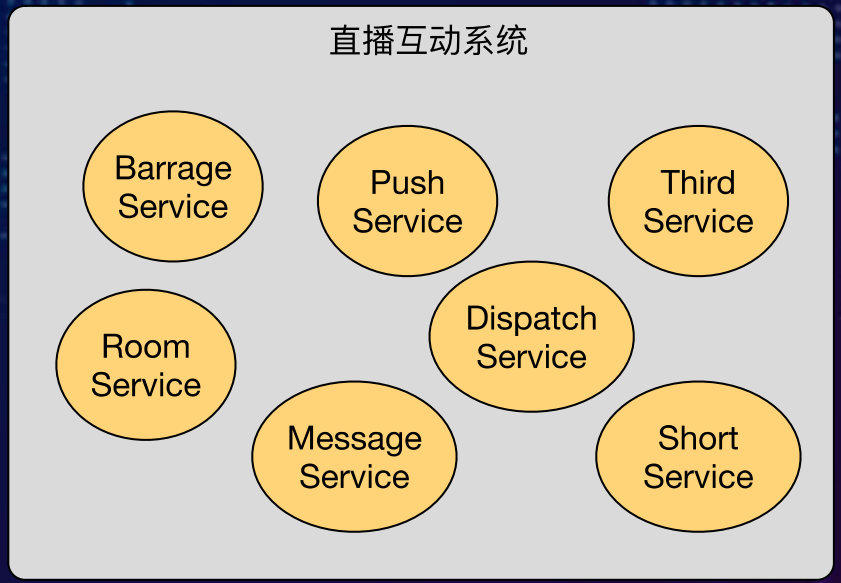
Barrage Service拆分的原因？

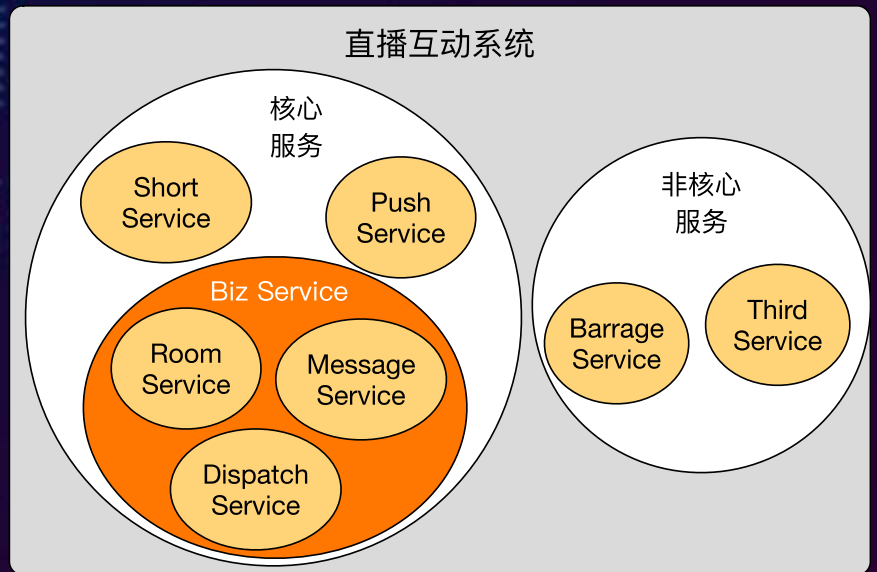
- Barrage Service分析：
  - 批量返回消息，带宽消耗型服务
  - 采用Gzip技术，CPU消耗型服务
  - 与Third Service拆分，减少相互影响
- Barrage Service优化：
  - 带宽：采用Gzip压缩及多层缓存策略
  - CPU：多层缓存，减少压缩次数



02 微服务 – 服务拆分

服务拆分前后对比





02 微服务 – 服务通信

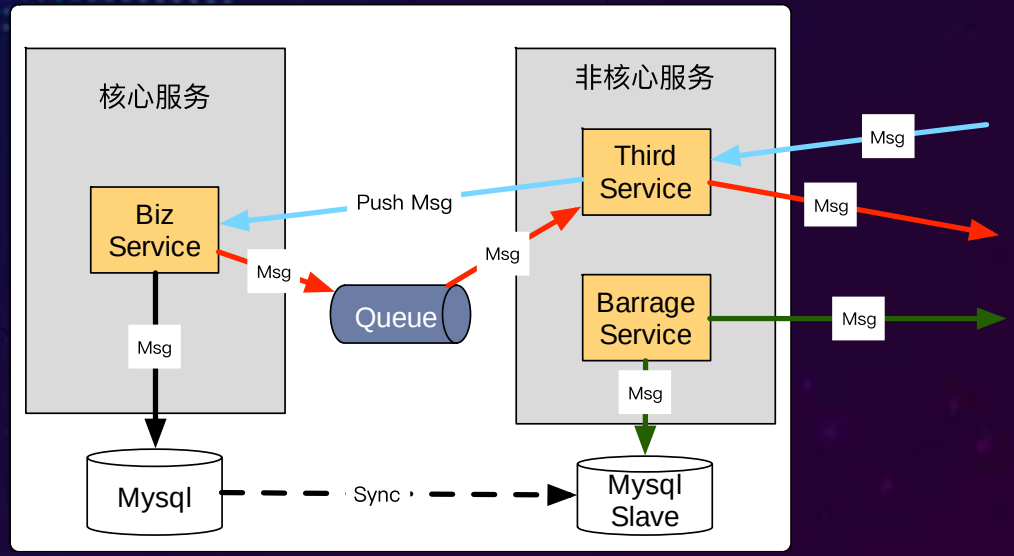
通信方式：

同步	异步
• REST • RPC	• 消息队列

02 微服务 - 服务通信

核心与非核心服务通信设计

- Third Service
  - Push：RPC
  - Pull：Queue
- Barrage Service
  - 共用数据库

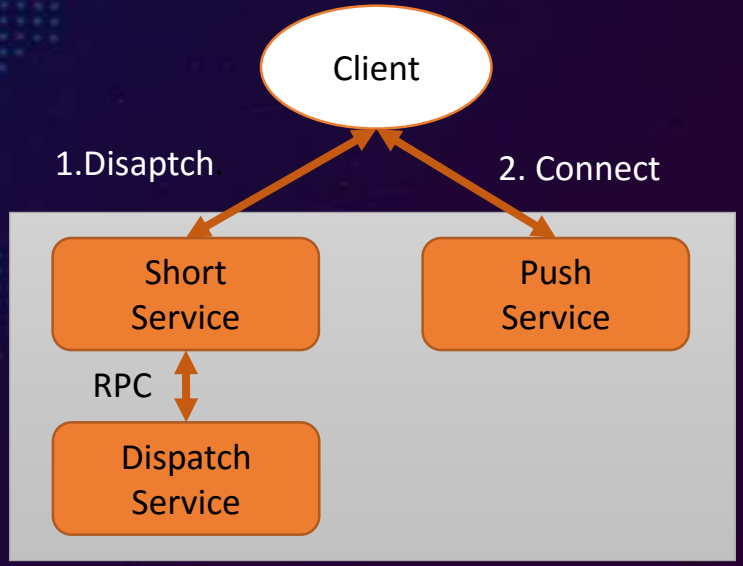


02 微服务 – 服务发现

Push Service 服务发现：

Dispatch Service

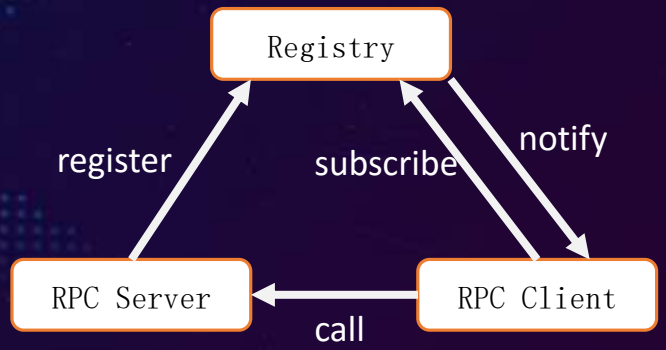
- 根据集群的负载情况动态下发长连地址
- 根据用户的地域动态下发
- 支持Push Service水平扩展



02 微服务 – 服务发现

Motan RPC 服务发现：

- Server：提供服务
- Client：使用服务
- Registry：可用服务列表



02 微服务化总结

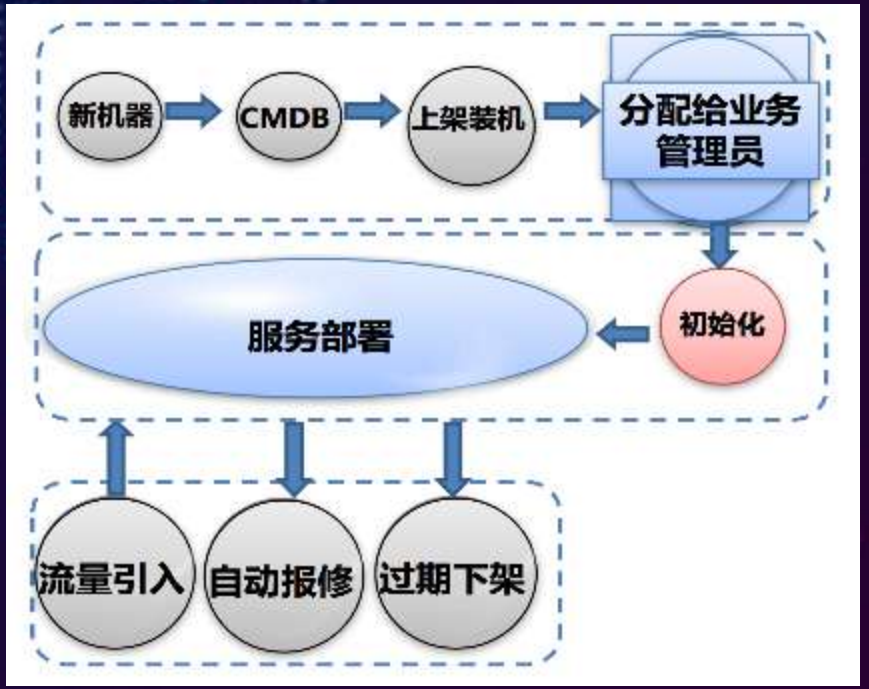
1. 独立开发测试，加快迭代速度
2. 通过服务拆分，减少无关联服务间相互影响
3. 提高启动速度，加快服务扩展速度
4. 增加了运维维护和部署难度

Part.3
直播互动的弹性扩缩容

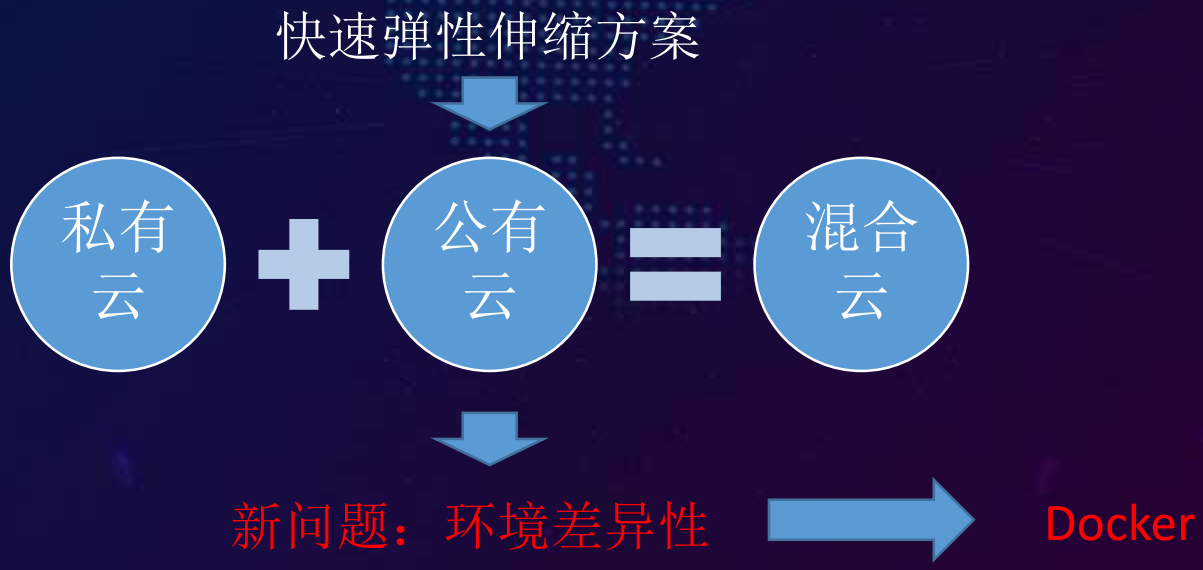
03 峰值应对 - 传统方案

痛点：

1. 扩容流程繁琐
2. 设备申请周期长
3. 人工干预多
4. 机器负载饱和度低



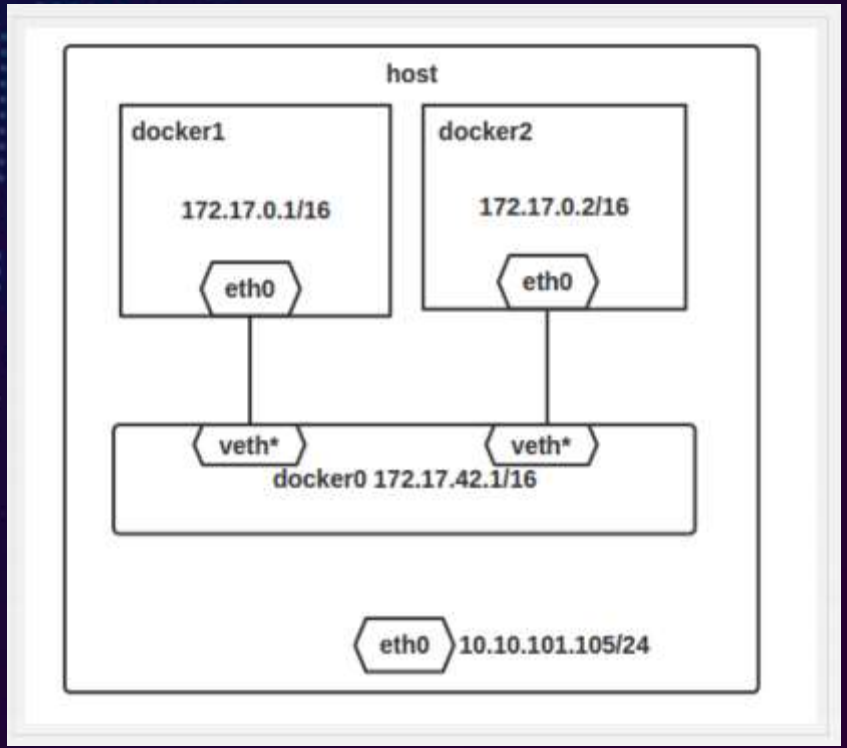
03 峰值应对 - 新方案



03 Docker – 网络模型选择

Bridge模式：

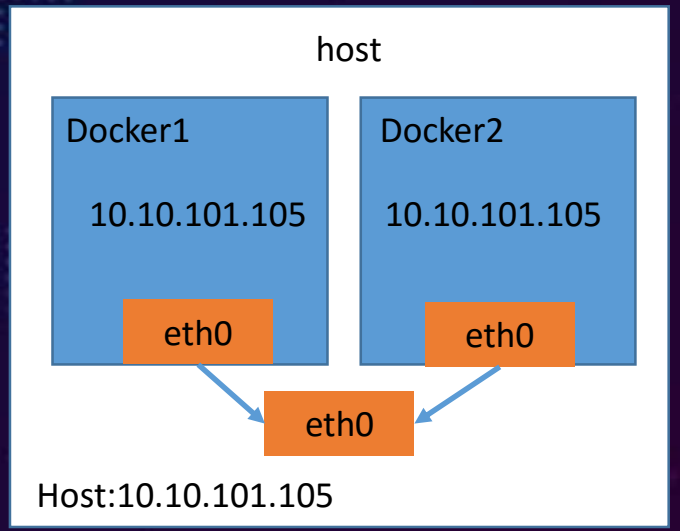
- Docker的默认设置
- 虚拟以太网桥docker0
- veth pair技术
- 问题：RPC服务注册后，Client难以访问



03 Docker – 网络模型选择

解决方案 - Host模式：

- 共享宿主机的网络命名空间
- 比Bridge模式更快，没有路由开销



03 Docker

Docker优点：

- 更快速交付和部署
- 更高效的虚拟化
- 更轻松的迁移和扩展
- 更简单的管理

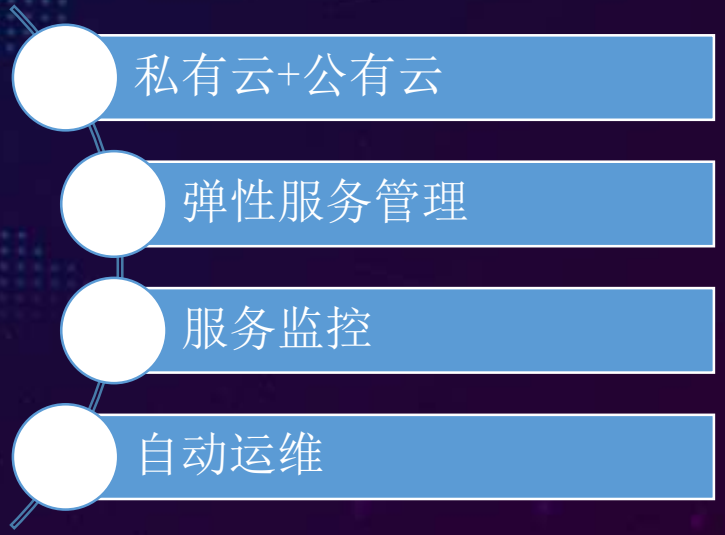
公有云

基于Docker的混合云架构 - DCP

03 DCP混合云平台简介

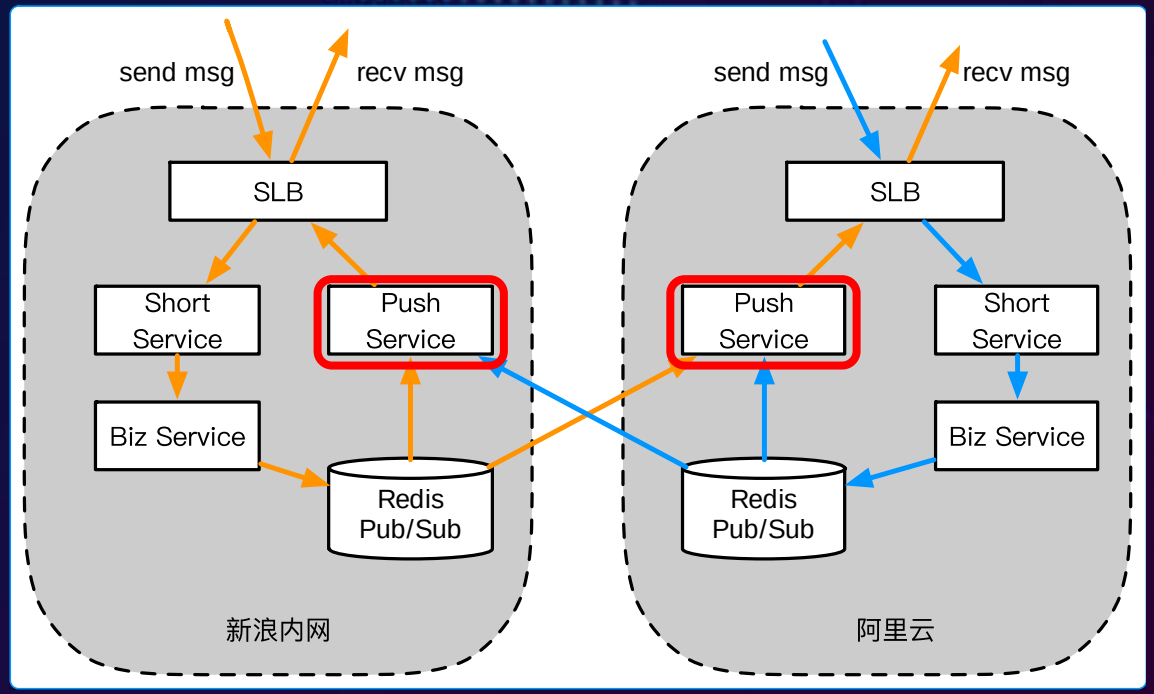
DCP(Docker Container Platform)

- 10分钟扩容1000+节点
- 每天600亿次的API调用
- 每天万亿次的RPC调用



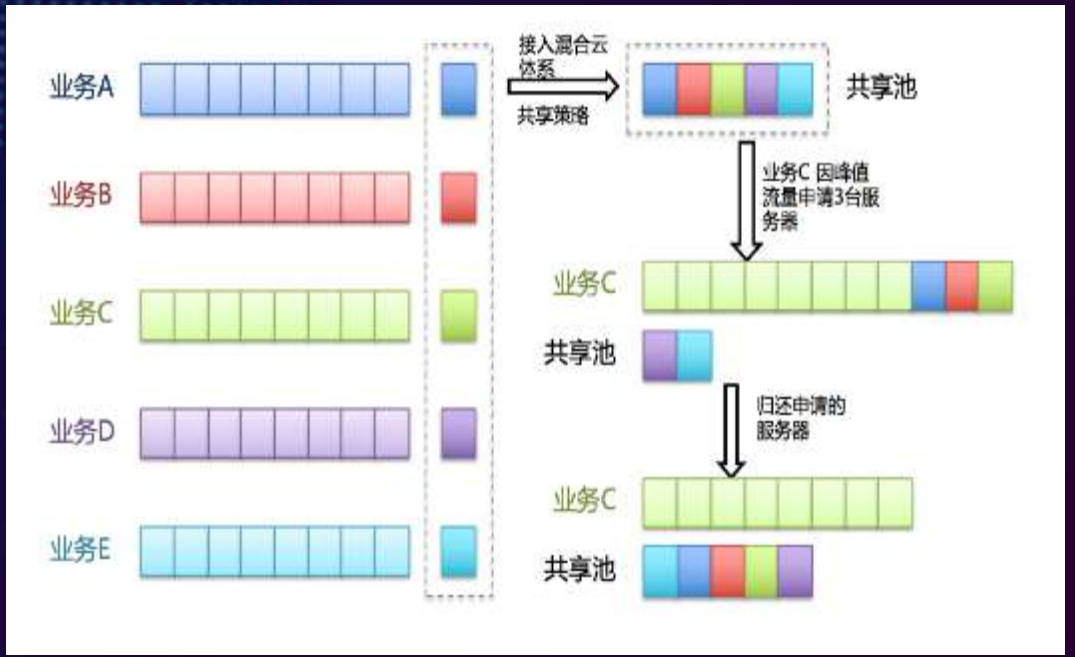
03 直播互动 + DCP

直播互动核心服务混合云部署架构



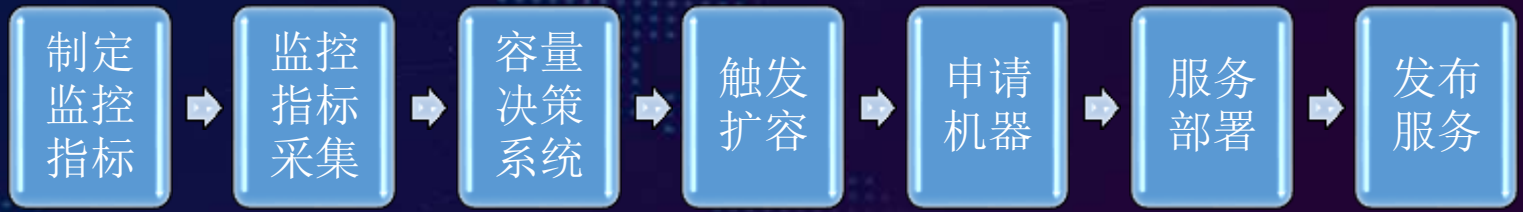
03 DCP弹性伸缩 - 化零为整

设备从哪来？

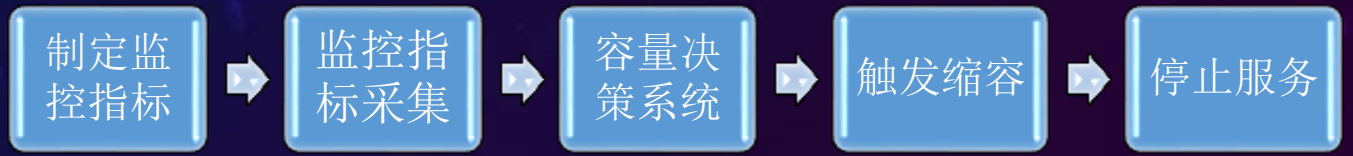


03 直播互动自动化扩缩容流程

- 扩容流程



- 缩容流程



03 监控指标

监控指标分类

1. 业务性能指标
2. 机器性能指标

制定监控指标流程

03 指标监控 - Push Service

业务性能指标

- 用户长连数
- 消息推送量

机器性能指标

- CPU
- 内存
- PPS
- 带宽
- IOPS

03 监控指标采集

03 弹性扩缩容总结

1. 通过Docker化解决环境差异问题，快速扩展
2. 通过混合云架构DCP解决资源弹性伸缩问题
3. 通过自动化扩缩容实现直播互动无人看守

Part.4
典型案例分享

04 未实现自动化扩缩容

背景：

- 神州飞船回收
- 时间：凌晨三点多
- 微博全量push

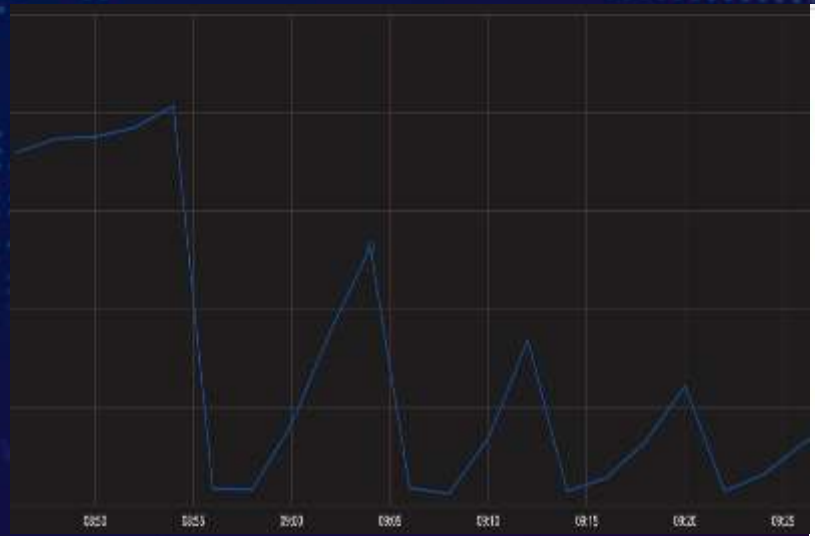
问题：

- 服务快到瓶颈
- 人工值班可能存在遗漏
- 监控不够完善

自动化扩缩容

04 实现自动化扩缩容之后

背景：十九大会议讲话直播



Thank you!