QCon
全球软件开发大会

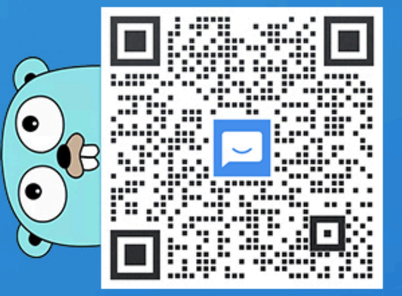
美团Serverless平台的探索与核心技术优化实践

殷琦

美团技术专家

抢占先机，成为未来 3 年抢手的后端开发人才

【Go 进阶训练营】带你成为字节跳动 2-2 级别 Go 工程师



扫码获取详细大纲
并咨询课程详情

- 12 大模块教学
- 3 大企业级实战案例
- 13 周视频课程教学
- 大厂助教 1v1 答疑
- 班主任全程贴心督学
- 简历直推一线互联网公司

自我介绍

殷琦，美团技术专家，基础架构应用中间件组负责人。

2015年加入美团，一直致力于高性能、高并发、高可用中间件研发实践。先后负责消息系统、微服务框架、任务调度、API网关、Serverless等中间件产品。

目录

背景-美团为什么要做Serverless？

尤其是前端开发，我太难了

资源利用率低

- ✓ 大部分业务流量存在明显波峰、波谷现象
- ✓ 大量低频业务

为了保证业务SLA、容灾要求需要冗余部署，导致资源利用率低

研发成本高

- ✓ 流程繁琐（如申请机器、构建、发布配置、申请域名等）
- ✓ 组件平台不统一、接入配置复杂（如日志中心、网关等）
- ✓ 各个组件开发模型不统一，学习使用成本高
- ✓ 发布、回滚慢，影响迭代速度

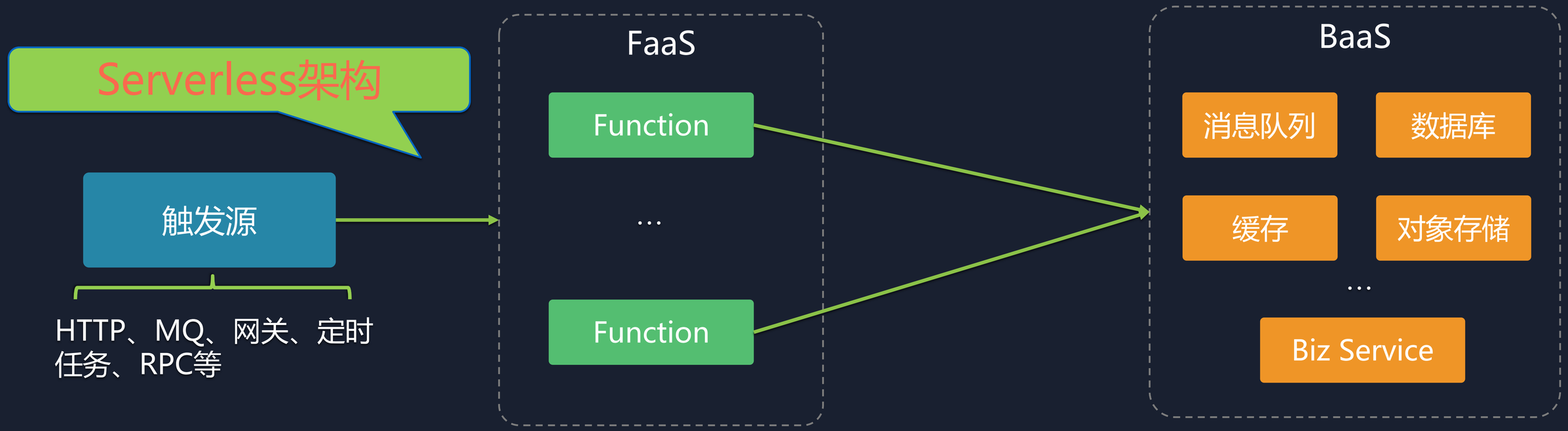
运维成本高

- ✓ 节点不可用
- ✓ 机器宕机
- ✓ 高低峰集群扩缩容
- ✓ 机房容灾

需要人工处理，存在大量的运维和运营成本

背景-美团为什么要做Serverless？

Serverless（无服务器架构）：FaaS、BaaS和面向应用的Serverless服务



Serverless特点：弹性伸缩、按需付费、事件驱动、无需运维

聚焦业务逻辑实现

背景-美团Serverless里程碑

Nest

2019.03

产品调研、立项

2019.08

MVP版本上线，上线试点业务

2019.12

支持高可用改造，冷启动优化(1)，丰富触发源

2020.06

支持资源池部署、轻量级容器，冷启动优化(2)

2020.12

支持合并部署，冷启动优化(3)，支持CLI、WebIDE研发工具

2021.05

支持IDE插件、场景化改造

1 快速落地，上线试点

2 优化技术，提升稳定性

3 完善生态，提升效能

目录

- 背景
- 选型与实现原理
- 核心技术优化
- 落地场景与收益
- 未来规划

选型与实现原理-技术选型

FaaS or BaaS or 面向应用的Serverless服务？

先FaaS再面向应用的Serverless服务

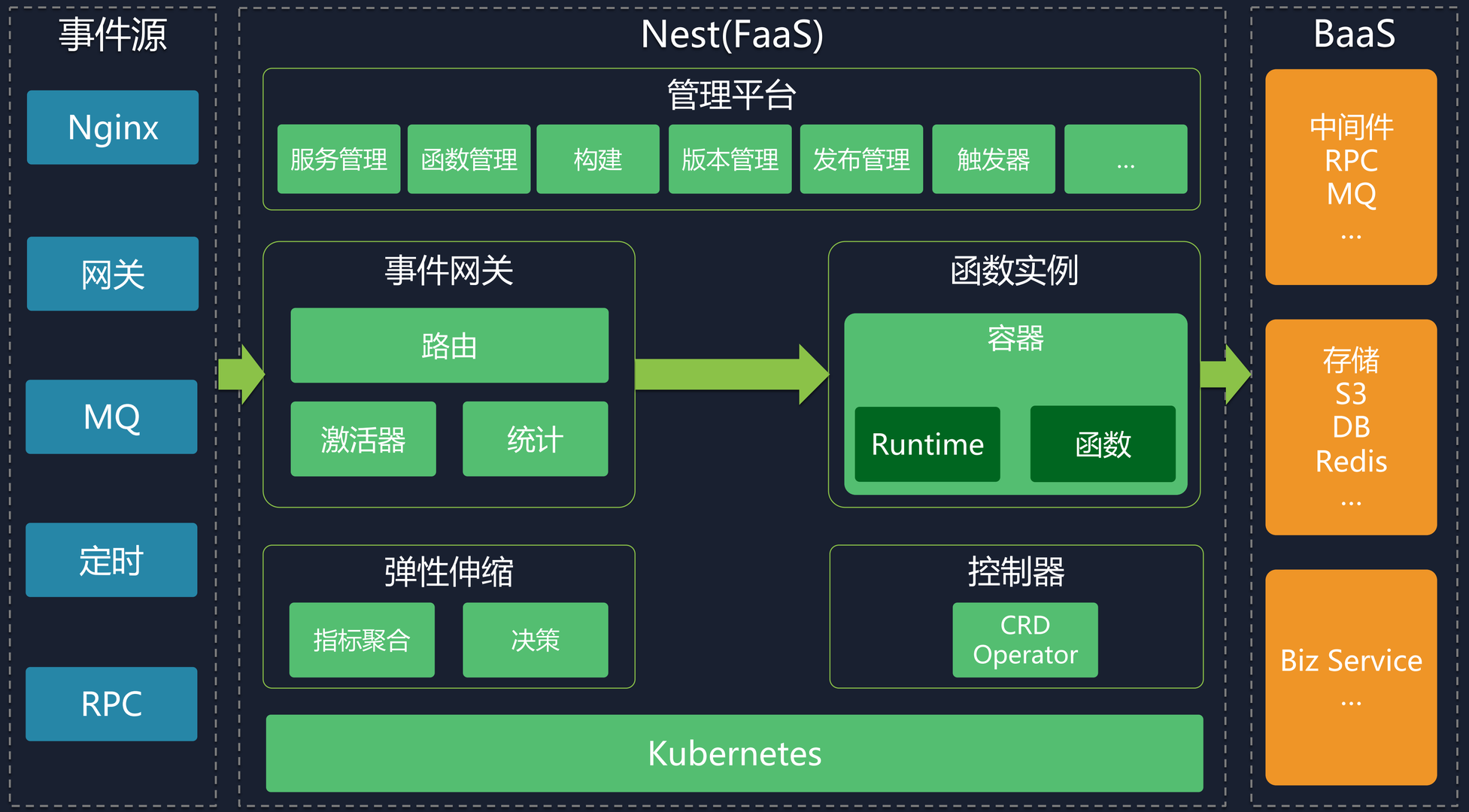
Hulk or 原生Kubernetes？

原生Kubernetes

Go or Java？

Java语言

选型与实现原理-架构

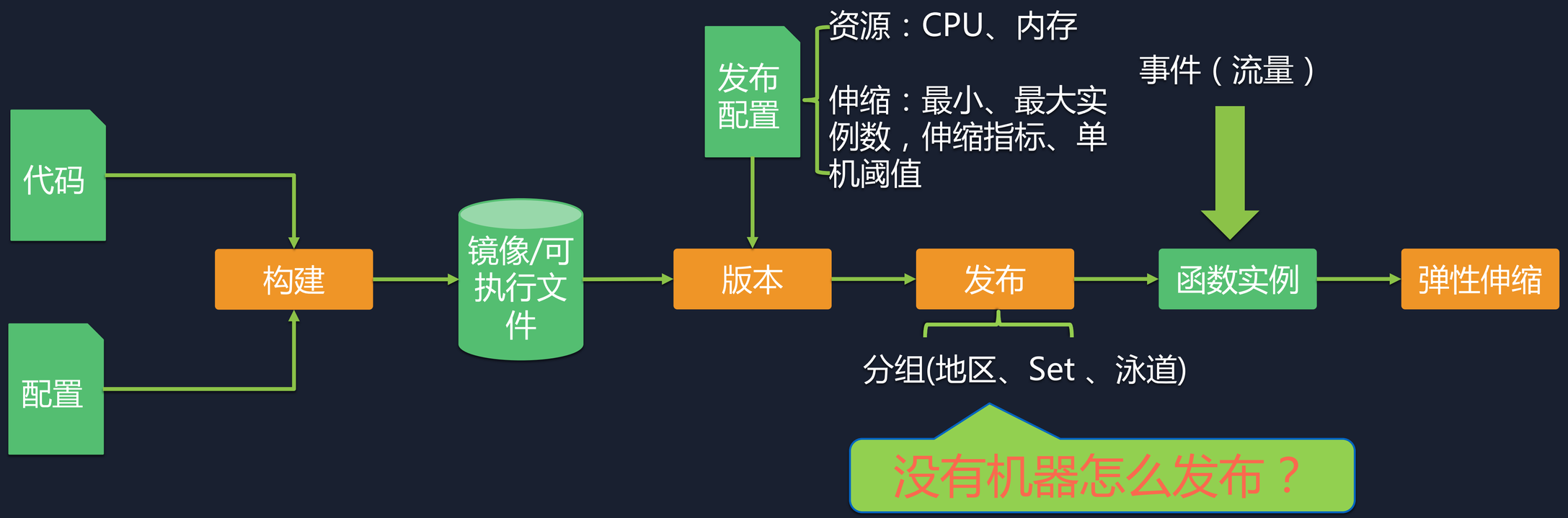


- 事件源
- 事件网关
- 弹性伸缩
- 控制器
- 函数实例

选型与实现原理-架构之弹性伸缩

- 何时伸缩：根据流量统计数据实时计算函数期望实例数
- 伸缩多少：指标/单实例阈值=期望实例数
- 伸缩快慢：取决于冷启动耗时

选型与实现原理-函数如何发布？



函数生命周期

目录

- 背景
- 选型与实现原理
- 核心技术优化
- 落地场景与收益
- 未来规划

核心技术优化-弹性伸缩

全托管，要求极高

问题1：伸缩频繁，服务不稳定

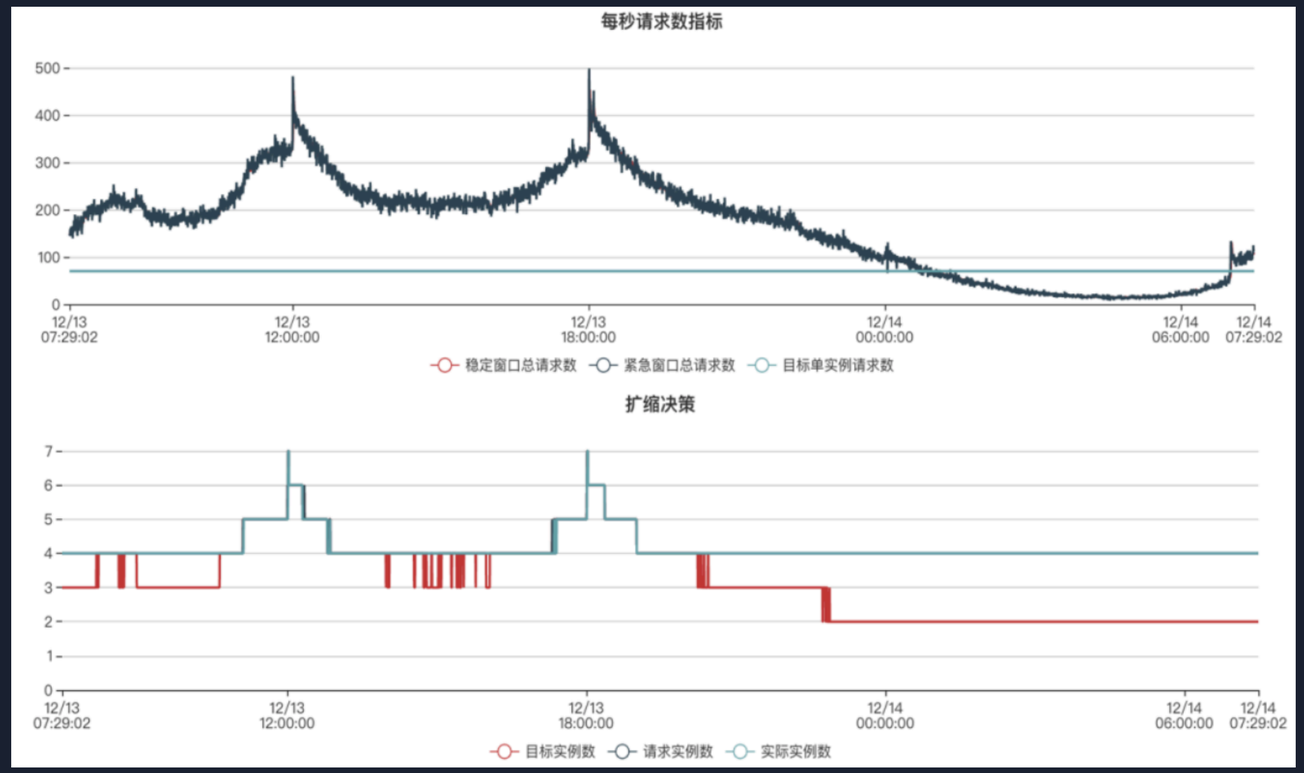
- 滑动窗口，计算均值
- 延时缩，实时扩
- 增加QPS指标

问题2：扩容来不及，服务不稳定

- 提前扩容，如达到阈值0.7倍就扩容

问题3：不同场景，需求不一

- 定时伸缩
- 混合指标伸缩（CPU、Memory、并发度、QPS）
- 支持预留实例



函数实例伸缩

最小实例数为4，单实例阈值100，阈值使用率0.7

核心技术优化-冷启动

冷启动是指在函数调用链路中包含了资源调度、镜像/代码下载、启动容器、运行时初始化、用户代码初始化等环节。

启动快 → 扩容快 → 高效稳定

核心技术优化-冷启动

多阶段持续优化

阶段1：镜像启动优化（容器IO限速、一些特殊Agent启动耗时、启动盘与数据盘数据拷贝）

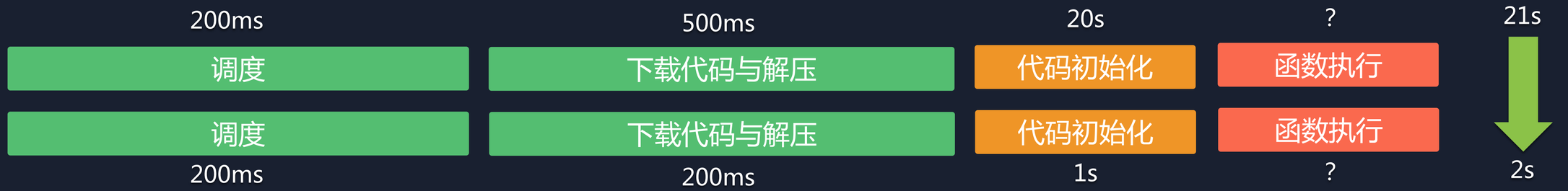
阶段2：资源池优化（空间换时间）

核心技术优化-冷启动

精益求精

阶段3：核心路径优化（高性能的压缩解压算法（LZ4与Zstd）以及并行下载和解压）

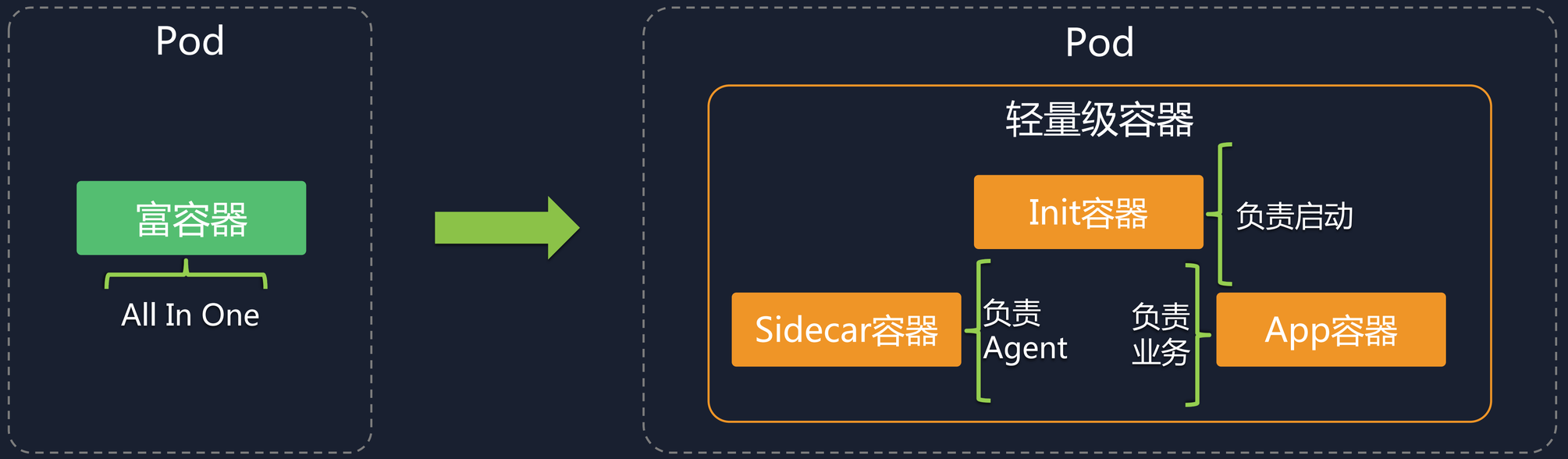
阶段4：基础逻辑下沉（减小包大小，预启动中间件等基础逻辑）



核心技术优化-容器稳定性

问题：刚扩容的实例不稳定，负载高，导致请求超时

原因：Agent升级导致；函数实例资源配置较小。容器内资源竞争



富容器 VS 轻量级容器

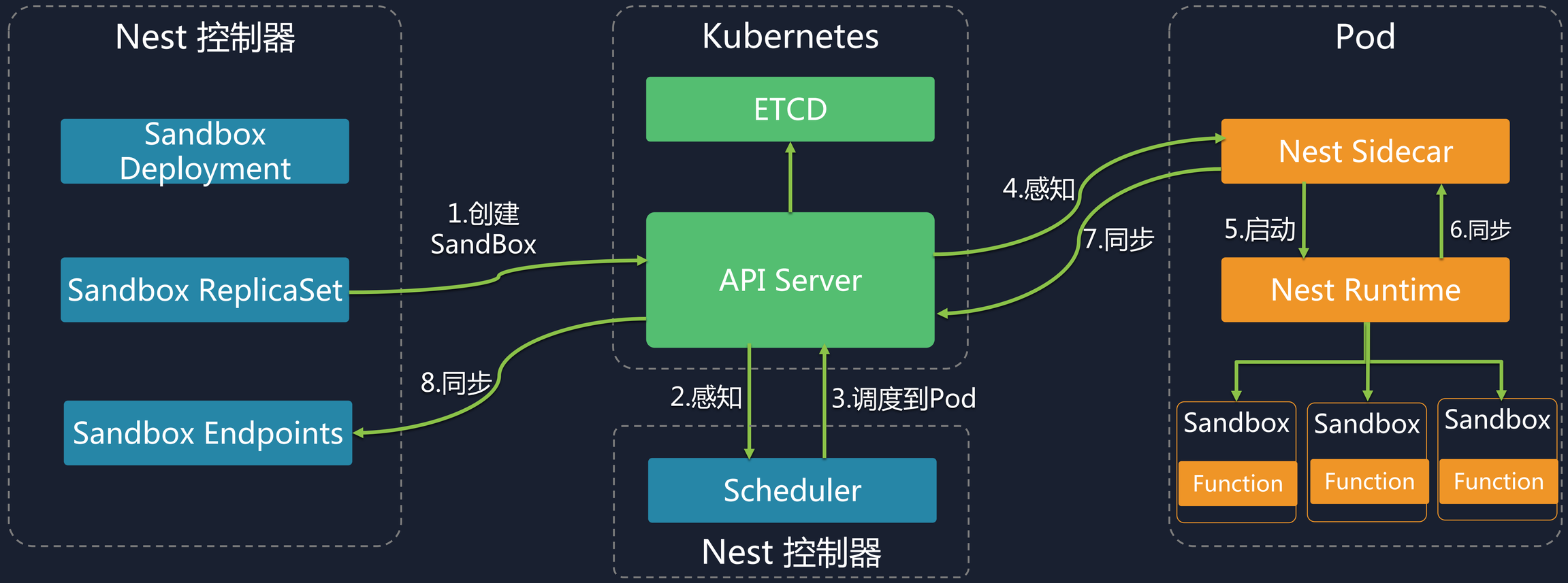
核心技术优化-合并部署

问题：资源利用率低

原因：低频业务不能缩容到0，预留实例；容器重，系统开销大

函数合并部署

核心技术优化-合并部署



参考Kubernetes设计，自定义CRD；Pod类比成Kubernetes Node，

Sandbox类比成Kubernetes Pod，Nest Sidecar类比成kubelet

核心技术优化-合并部署

问题：不同函数之间的如何隔离？ → 方案：不同语言不同方案

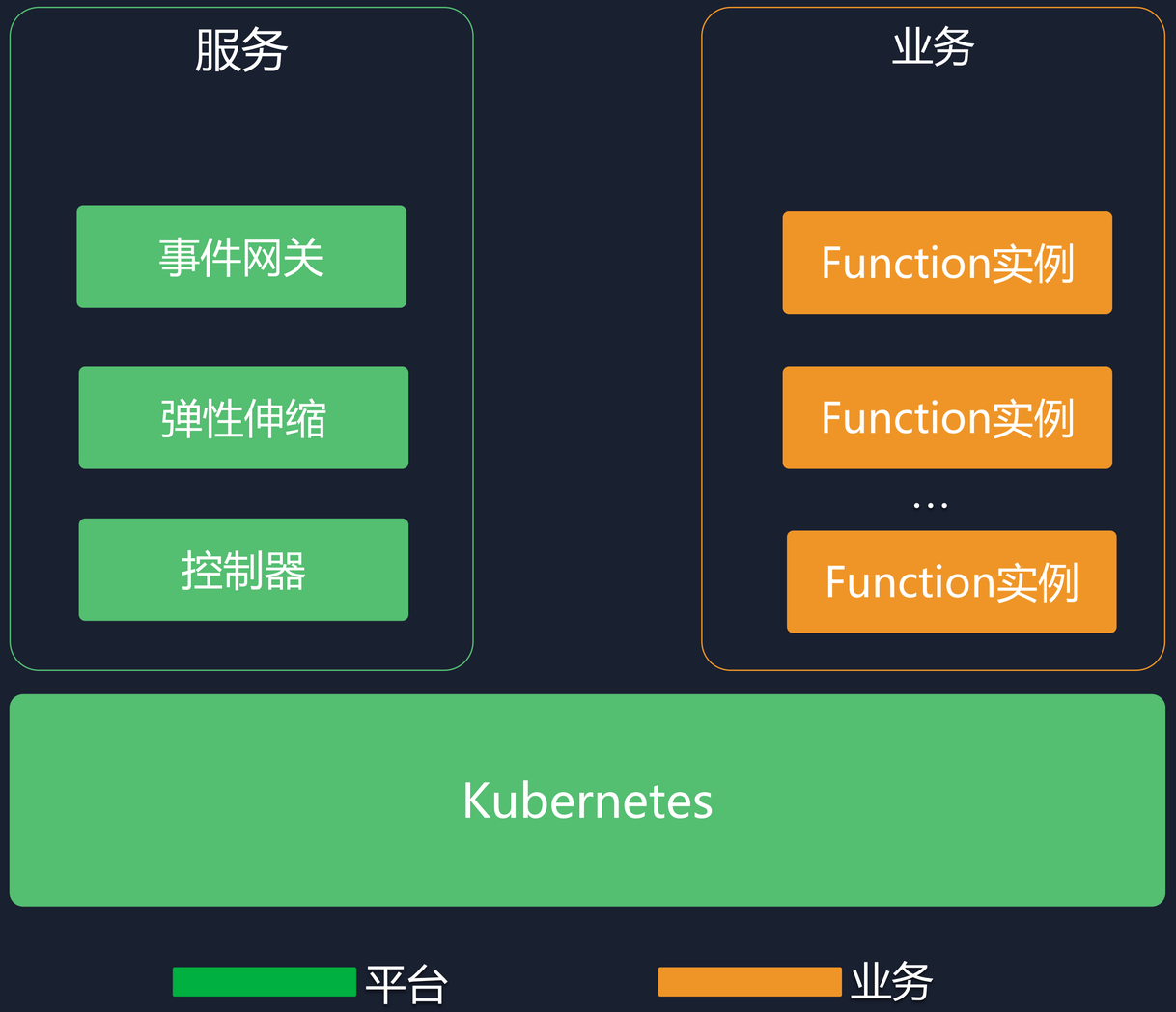
不同的隔离策略

核心技术优化-研发生态

问题：研发流程割裂 → 方案：集成与被集成

核心技术优化-高可用

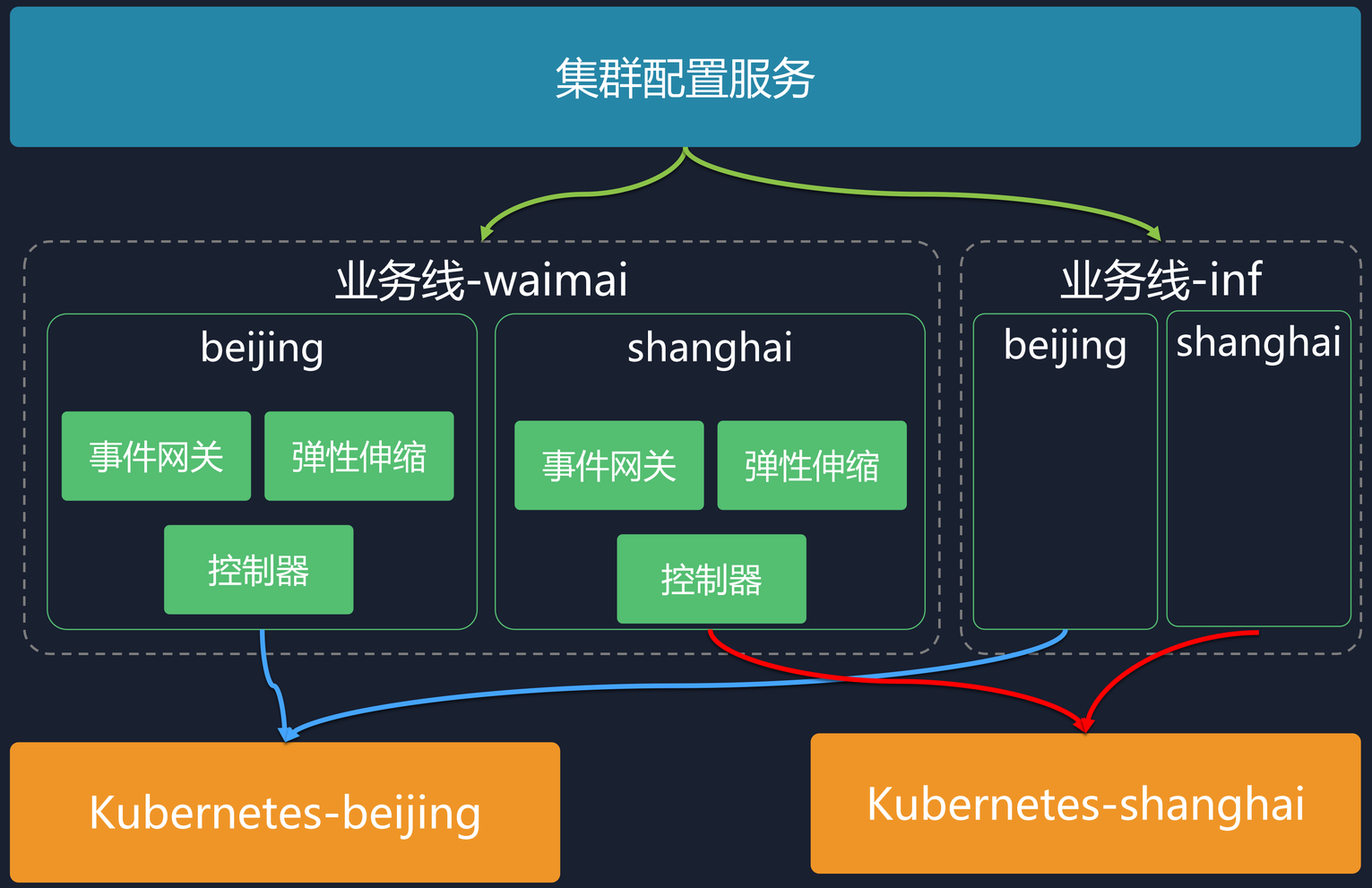
双管齐下：平台与业务



核心技术优化-高可用

平台高可用

- 架构层：地域隔离、业务线隔离
- 服务层：事件网关采用异步化和限流
- 监控运营层：监控告警、核心链路治理、故障演练、梳理SOP
- 业务视角层：7*24不间断实时巡检



核心技术优化-高可用

业务高可用

- 服务层：降级、限流
- 平台层：实例保活、机房打散，多层级容灾、监控告警

函数监控报表

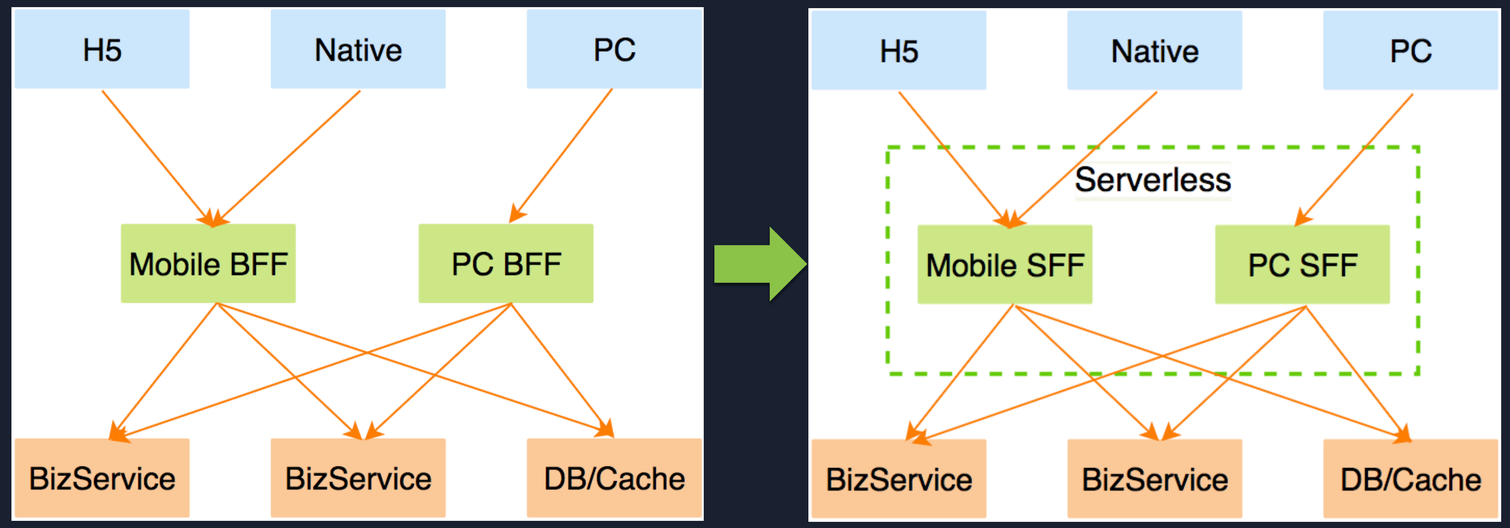
目录

落地场景与收益

前端Node.js技术栈规模化落地，几乎涵盖了所有业务线，如外卖、优选、酒旅等

后端Java技术栈在个别业务线做了一些落地

- BFF(Backend for Frontend)场景
- SSR(Server Side Render)场景
- 后台管理端场景
- 定时任务场景
- 数据处理场景
- 集成平台场景



BFF vs SFF

落地场景与收益

1 资源利用率提升到40%+

2 研发运维效率提升30%-40%

提升资源利用率

- ✓ 高频业务，弹性伸缩，提升到40%以上
- ✓ 低频业务，合并部署，成倍提升

提升资源利用率

提升研发成本

- ✓ 无需申请机器
- ✓ 提供CLI、WebIDE工具，一键生成代码模版
- ✓ 统一开发模型，降低学习成本
- ✓ 一键接入日志中心、自带监控
- ✓ 秒级别发布、回滚

关注业务逻辑实现

提升运维成本

- ✓ 自动替换异常实例
- ✓ 高低峰集群自动扩缩容
- ✓ 自动各机房打散机器，机房故障，自动容灾

全托管，无需运维

目录

- 背景
- 选型与实现原理
- 核心技术优化
- 落地场景与收益
- 未来规划

未来规划

场景化改造

支持Serverless工作流

完善研发生态

传统服务Serverless化


THANKS

美团_Serverless_平台的探索与核心技术优化实践

扫描二维码 提交议题反馈

2021 InfoQ 技术大会近期会议推荐

—— 盘点一线大厂创新技术实践

北京站

GMTC
全球大前端技术大会

时间：2021年07月04-05日
地点：北京 · 国际会议中心

扫码查看完整日程

深圳站

ArchSummit 全球架构师峰会 th

时间：2021年07月23-24日
地点：深圳 · 大中华喜来登酒店

扫码查看大会专题