探索微服务下一站
Koupleless 模块化研发框架与运维调度系统

盛知 蚂蚁集团

Content

目录

Part 01

微服务现状

协作与资源成本

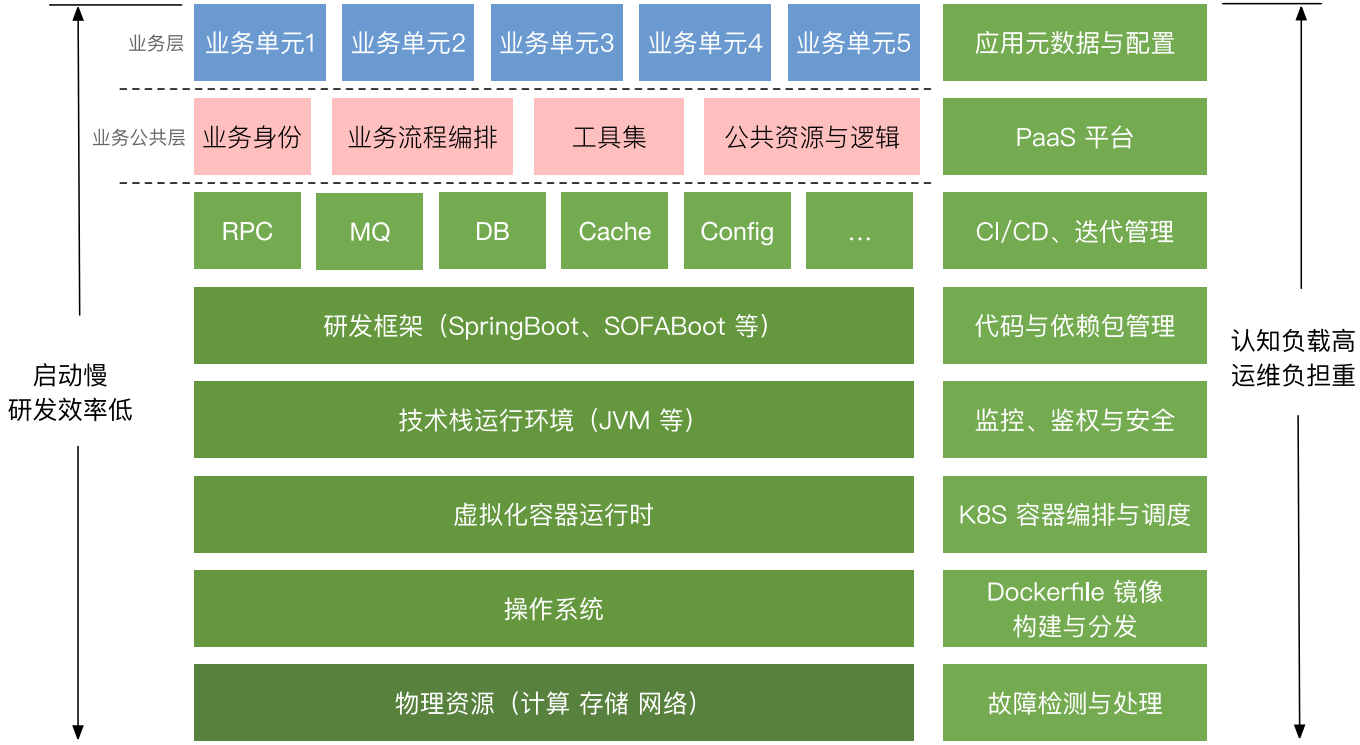
小应用过多
– 资源成本
– 长期维护成本高

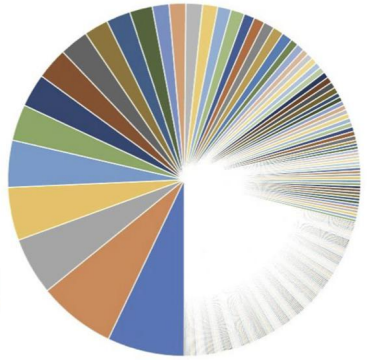
大应用过大
– 多人协作阻塞
– 变更影响面大风险高

研发效率问题

效率低：认知负荷高，运维负担重

- 业务开发者需要感知复杂基础设施，异常多
- 框架与中间件升级维护成本高、周期长
- 部署上线慢，特别是大应用（构建启动慢、机器多）





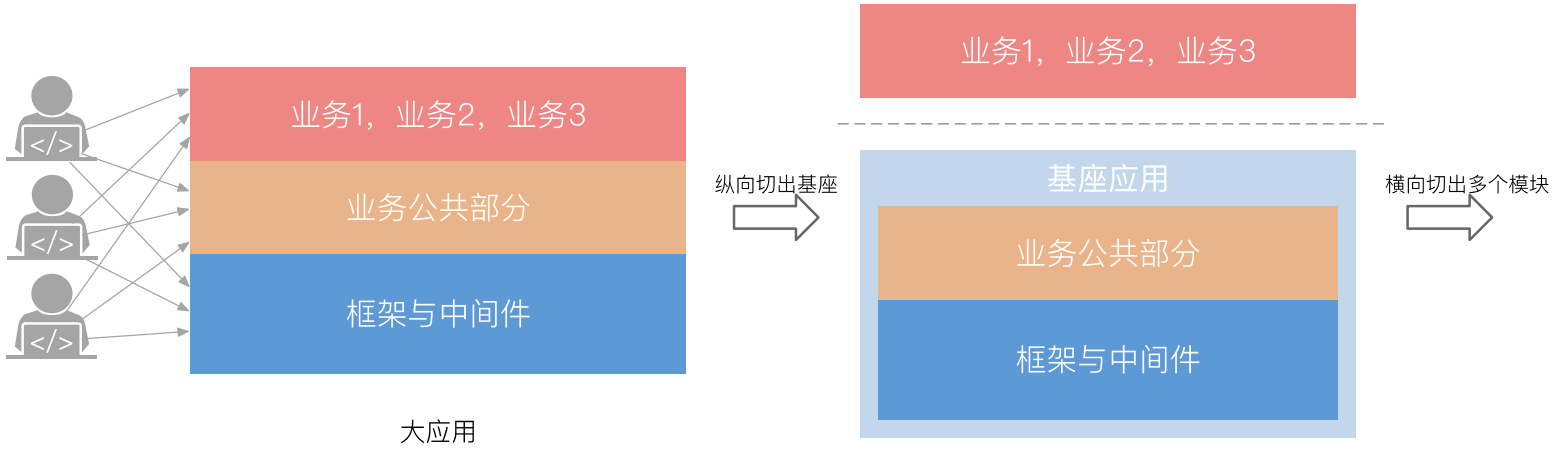
异常多

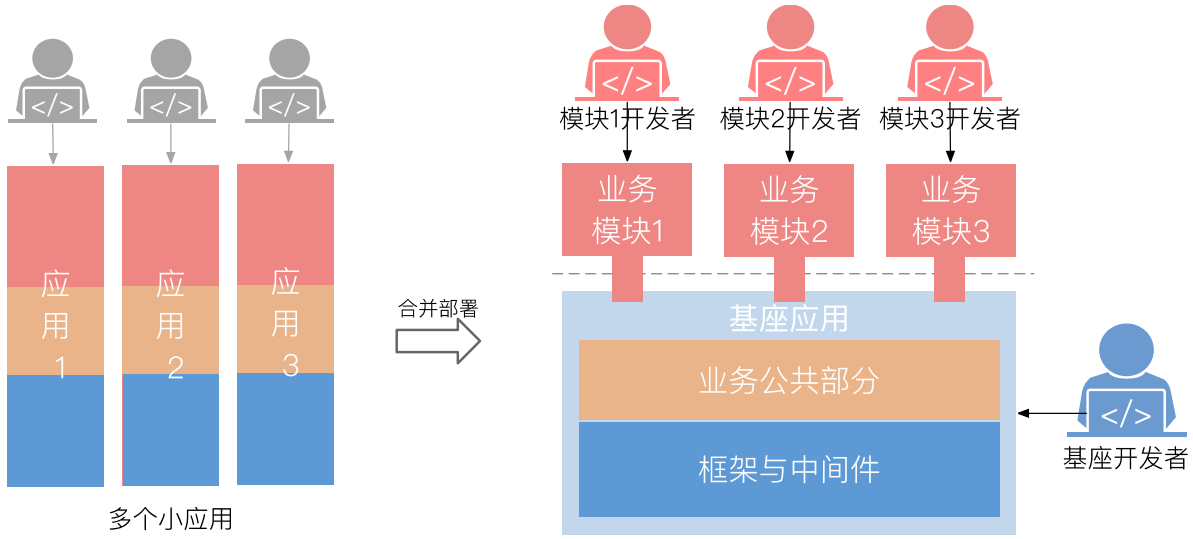
Part 02

Koupleless 的解法与效果

Koupleless 模块化研发框架

大应用拆分成多个模块，小应用合并部署成一个进程





模块：

- 不需关心资源、容量
- 专注业务开发
- 秒级启动
- 30秒级并行发布

基座：

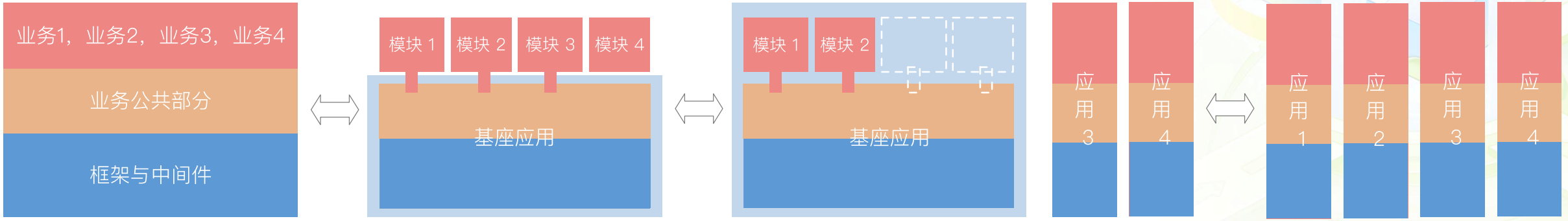
- 研运与传统应用完全一致
- 沉淀通用依赖和逻辑
- 为模块提供运行资源与环境

Koupleless 模块化研发框架

SOFASTACK

降低微服拆分成本

- ~~资源成本~~
- 单体架构与微服务架构之间增加模块架构
- 提供半自动化拆分工具，单体应用低成本拆分成模块应用
- 模块应用可低成本演进成微服务，也可回退回单体应



单体架构　模块架构　模块架构　微服务架构　微服务架构

Koupleless 模块化研发框架

模块是什么

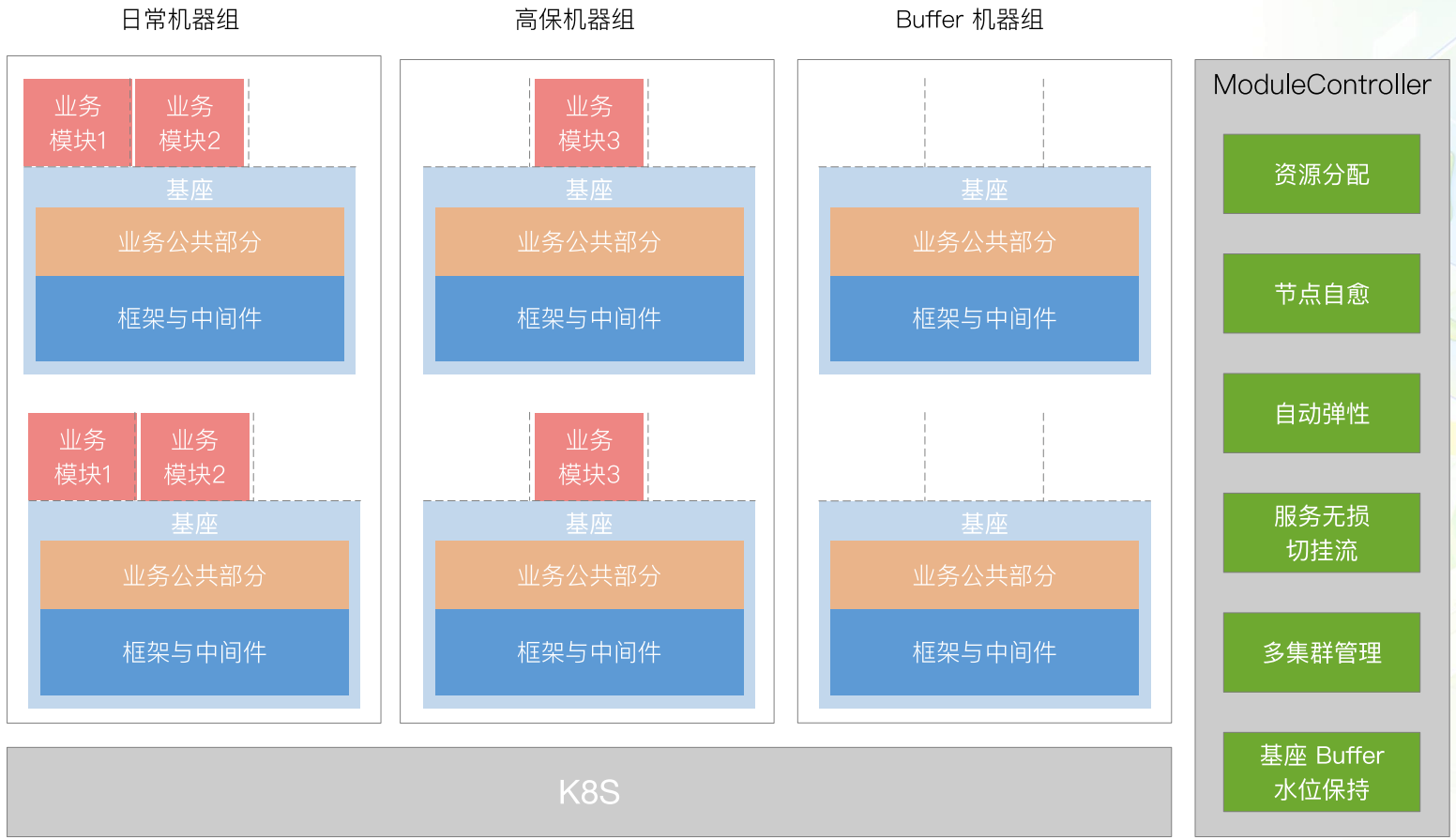
- 一个普通的 SpringBoot, 构建产物为 jar 包而非镜像
- 每个模块一个独立的 ClassLoader + SpringContext
- 热部署（不重启机器）


```xml
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter</artifactId>
    <scope>provided</scope>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
    <scope>provided</scope>
</dependency>
```

Koupleless 研发与运维调度平台

SOFASTACK

调度与伸缩



Koupleless 模块化研发框架

可解决的问题

- 应用拆分过度，机器成本和长期维护成本高
- 应用拆分不够，多人协作互相阻塞
- 应用构建、启动与部署耗时久，应用迭代效率不高
- SDK 版本碎片化严重，升级成本高周期长
- 平台、中台搭建成本高，业务资产沉淀与架构约束困难
- 微服务链路过长，调用性能不高
- 微服务拆分、演进成本高

模块化研发效果对比

	传统应用	模块	对比
构建速度	265 s	27 s	1/10
构建产物大小	1385 MB	0.02 MB	1/70000
运行内存消耗	337 MB	17 MB	1/20
部署耗时	141 s	4s	1/35

Koupleless 研发与运维调度平台

SOFASTACK

– 只感知业务本身，低认知负载，秒级启动
– 并行迭代无阻塞

– 模块可独立部署成进程，也可合并部署成模块
– 部署粒度与变更风险面小，只涉及模块代码和对应机器

– 模块不占额外机器，只占极少业务自身的cpu和内存
– 弹性调度粒度小，资源密度高

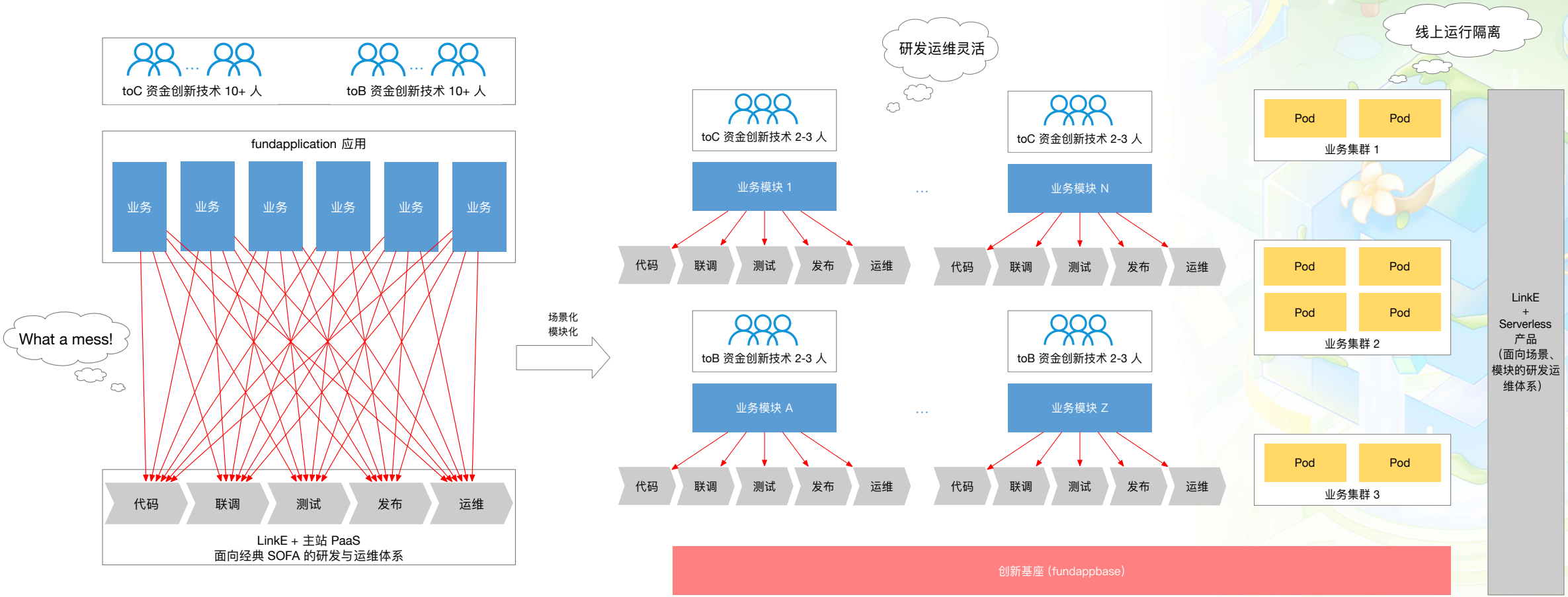
– 存量应用可低成本改造成或拆分成模块
– 模块可低成本演进成微服务也可回退回单体

– Speed as you need
– Pay as you need
– Deploy as you need
– Evolve as you need

Part 03

Koupleless 实践经验与案例效果

Koupleless 案例一：蚂蚁集团热部署 – 提高研发效率



主要矛盾

创新效率低下，发布 10min+，周级别 迭代

多团队共建协作成本高，互相抢占现象严重

流量不隔离，无法支持业务高保，故障风险高

Serverless 架构红利

创新大幅提效，发布 10min => 13s，周级别 –> 1周3次 * x 迭代

多人敏捷迭代，模块独立开发运维互不影响，资源成本下降

隔离流量和资源从而实现故障隔离

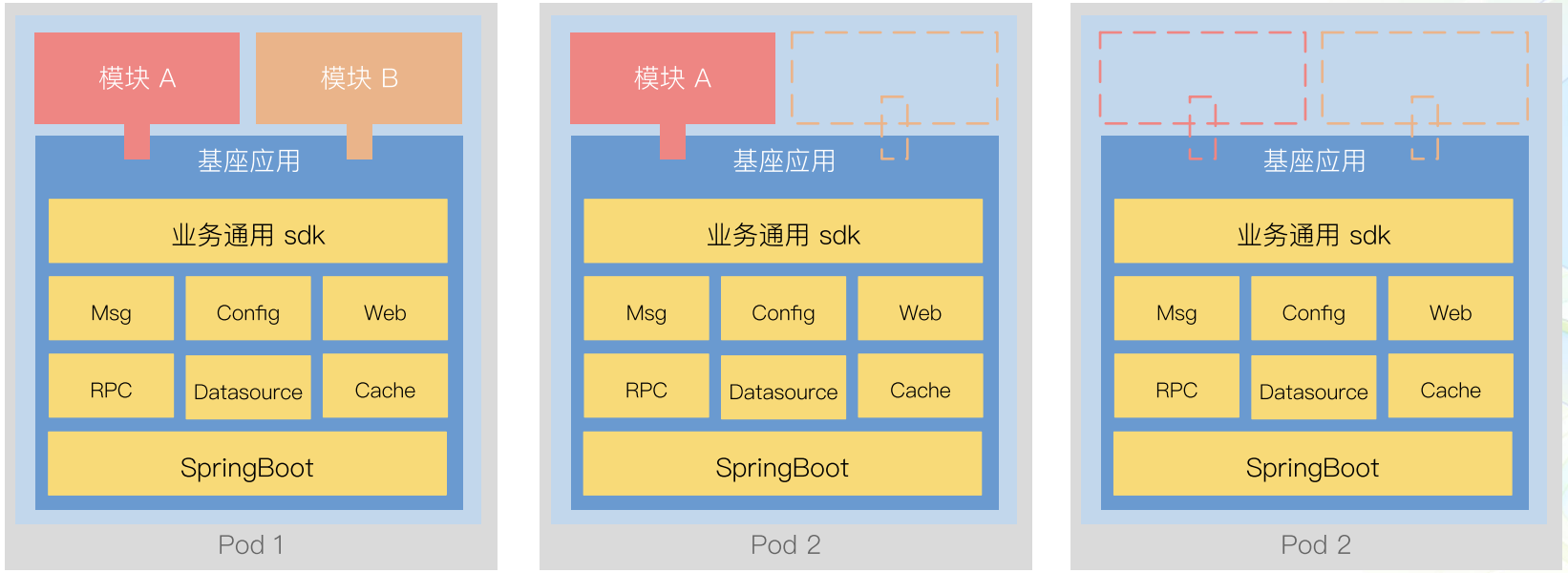
Koupleless 案例二：南京爱福路合并部署 – 省资源

价值： 长尾应用多且每个机房至少要部署 2 台机器，CPU 使用率仅 10%。使用合并部署多个应用可以合并到同一个基座上，基座由各业务域专人负责维护，从而极大降低了开发者的运维成本和资源成本。

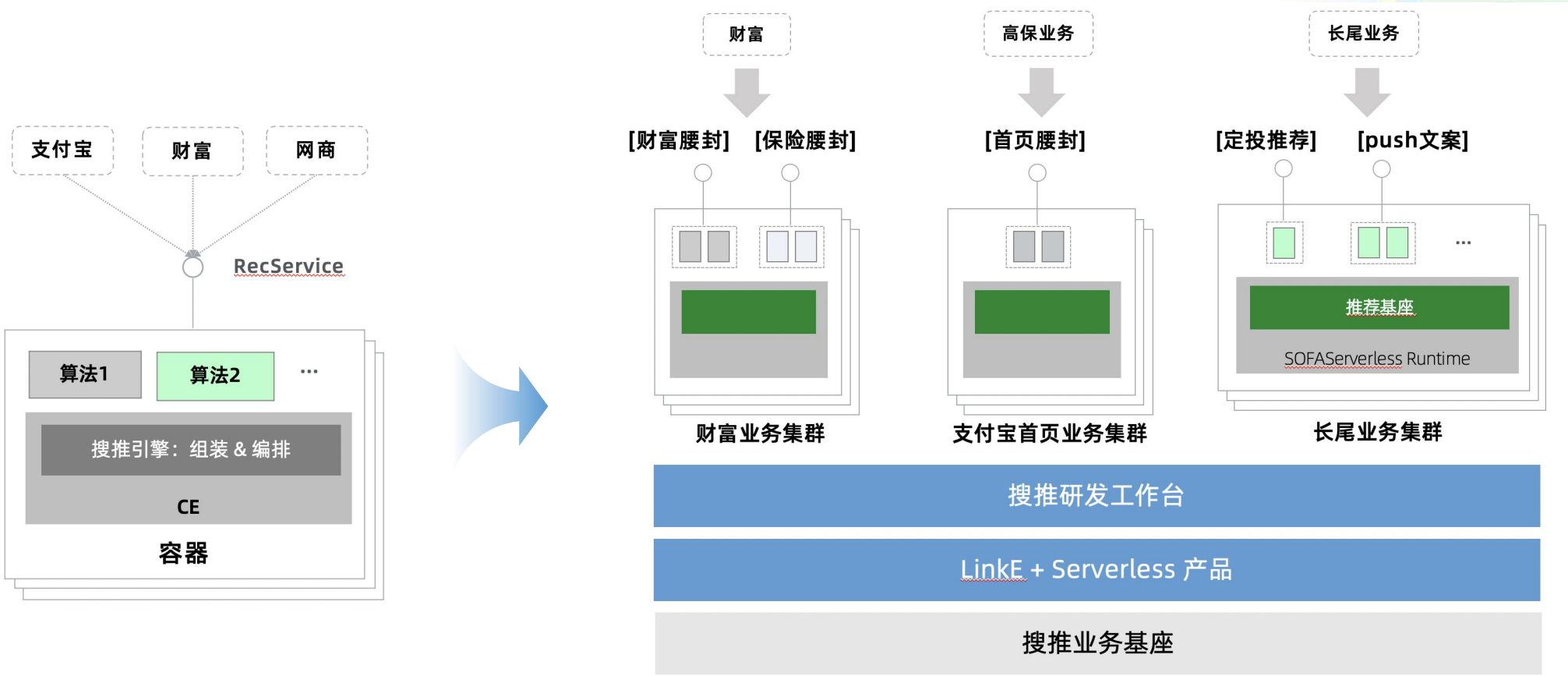
收益：
1. 极致的**资源成本降低**，南京爱福路目前已经将 6 个传统应用合并到 1 个基座上
2. 大幅提升启动速度，启动速度从 114 秒下降到 29 秒

Koupleless 案例三：阿里国际通用基座 – 基础设施下沉

价值： 在阿里国际，各种 SDK 的升级打扰、构建发布慢、申请机器额度管理是痛点问题。借助 Koupleless 通用基座模式，帮助部分应用实现了基础设施低感升级，同时应用的构建与发布速度也从 600 秒减少到了 60 秒。



Koupleless 案例四：中台与代码片段



特点：

- 基座更加复杂，大部分业务接口与编排逻辑定义在基座
- 模块更小：SPI 的实现、代码片段或者 类似 groovy 脚本

Koupleless 开源与规划

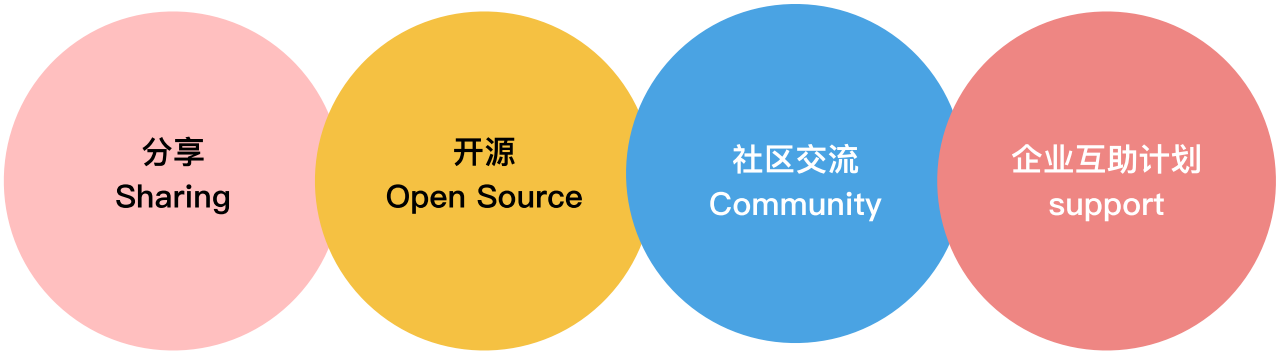
关键时间计划

当前：SOFAArk 模块化组件已开源 6 年，ModuleController、Arklet、Runtime 1.0 版本已正式发布！可统计到的有 20+ 外部企业投产

Q2：模块低成本接入、多应用适配框架完成上线

Q3：ModuleController 1.5 版本发布，模块代码扫描与准入

Q4：ModuleController 调度与伸缩，模块与基座管理产品发布



落地场景
客户案例
踩坑经验

源码能力
经验方法
治理标准

社区共建
长期演进

新增 50+ 企业实现降本增效

微信交流群

钉钉交流群

Thanks.