

搜狗搜索广告检索系统

弹性架构演进之路

Geekbang>

极客邦科技

全球领先的技术人学习和交流平台

扫我，码上开启新世界



Geekbang>

InfoQ | EGO NETWORKS | StuQ

InfoQ

专注中高端技术人员
的社区媒体

EGO NETWORKS

EXTRA GEEKS' ORGANIZATION
高端技术人员
学习型社交网络

StuQ

实践驱动的IT职业
学习和服务平台



促进软件开发领域知识与创新的传播



实践第一 案例为主

时间：2015年12月18-19日 / 地点：北京·国际会议中心

欢迎您参加ArchSummit北京2015, 技术因你而不同



ArchSummit北京二维码



[北京站]

2016年04月21日-23日



关注InfoQ官方信息
及时获取QCon演讲视频信息

目录

1

背景介绍

2

系统架构的演进历程

3

展望未来

搜索广告：效果即人心



网民眼中的搜索广告

传统搜索广告

 [酒店预订](#) [携程2折起](#) [携程保证最低价](#)

酒店预订上携程网,享受携程有房保证!预订确认后若到店无房,获赠首晚房价,在携程订酒店保证最低价,买贵了赔您3倍差价,住酒店每间夜送100元酒店消费券,...

搜索广告-新形态

 [酒店](#), [免费预订酒店](#), [艺龙官网](#)

热门城市: [北京酒店](#) | [上海酒店](#) | [广州酒店](#) | [天津酒店](#) | [更多](#)

海景城市: [青岛酒店](#) | [三亚酒店](#) | [大连酒店](#) | [厦门酒店](#) | [更多](#)

限时抢购: [北京抢购](#) | [上海抢购](#) | [广州抢购](#) | [天津抢购](#) | [更多](#)

酒店: [短租公寓](#) | [机票预订](#) | [旅游指南](#) | [酒店团购](#) | [更多](#)

图谱搜索广告-新产品

猜您关注 - 推广



连锁酒店



快捷酒店



广州香格里拉大酒店



汉庭



六星级酒店



希尔顿酒店



如家快捷酒店



酒店式公寓

新形态

- 广告展示形态多样
- 广告物料极大丰富

新逻辑

- 搜索广告: 关键词定位, 精准展示
- 图谱广告: 通过实体关系, 推荐展示

攻城狮面临的挑战？

产品策略
迭代频繁

新产品
变现

新变现系统 OR 迭代老系统？

广告物料
指数增长

流量
快速增长

系统运维
成本越来越高

机器成本
增长快，
预算不足

...



变

目录

1

背景介绍

2

系统架构的演进历程

—服务架构演进

—数据架构演进

—运维架构演进

—心得体会

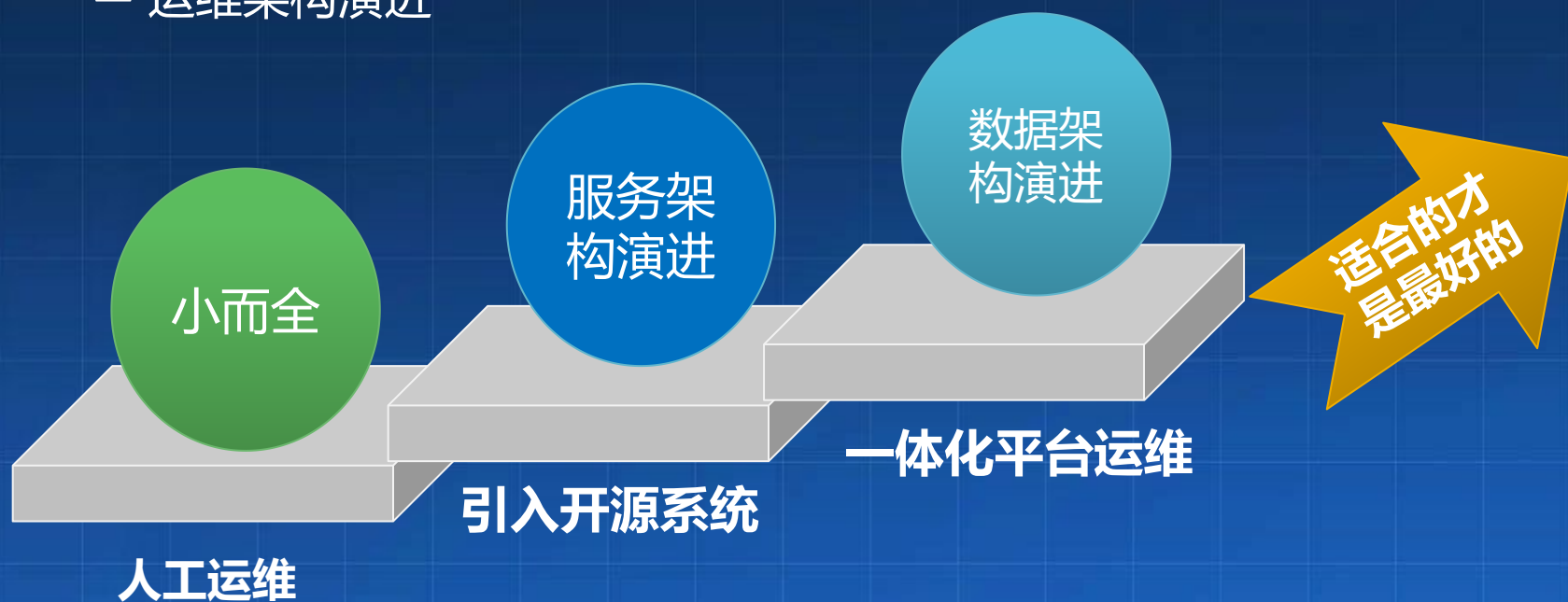
3

展望未来

系统架构的演进历程

- 三个维度

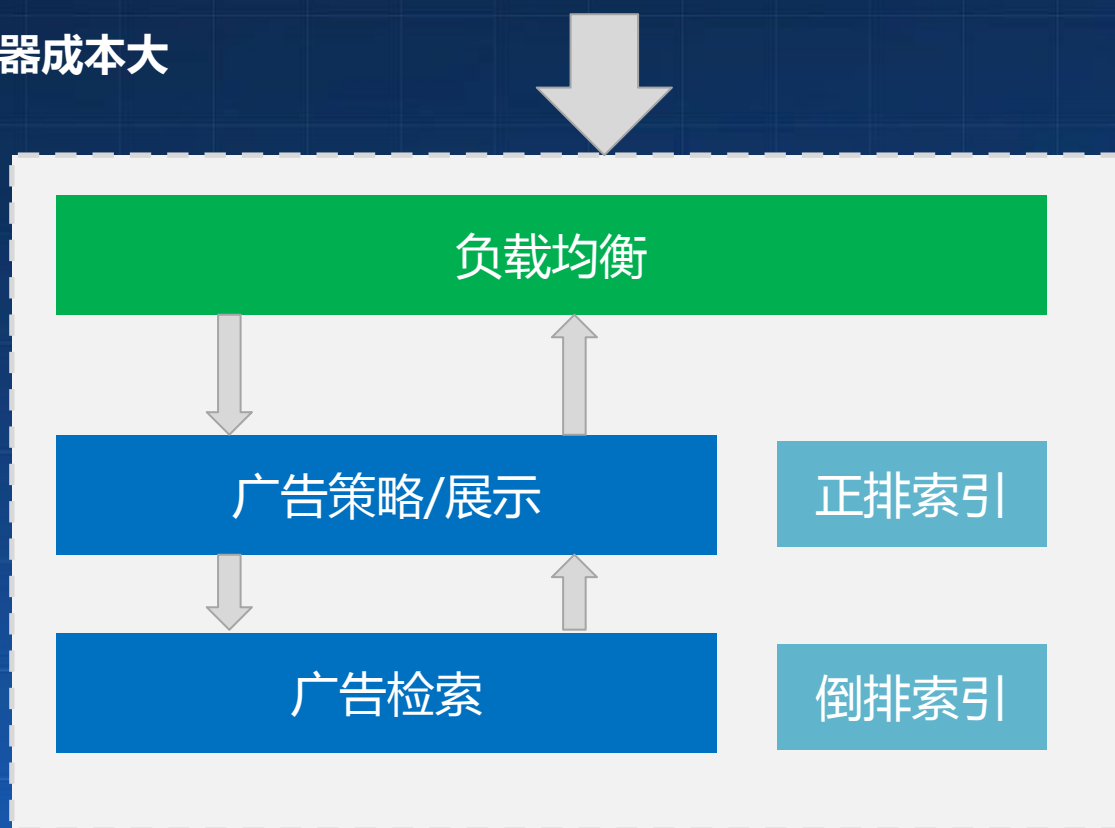
- 服务架构演进
- 数据架构演进
- 运维架构演进



初始架构-小而全

特点

- 结构简单，机器占用少，运维成本低
- 灵活性不够，不便于扩展
- 支持垂直扩展，机器成本大



初始架构-物理部署结构

PC搜索广告请求



PC搜索广告
集群



问问广告请求



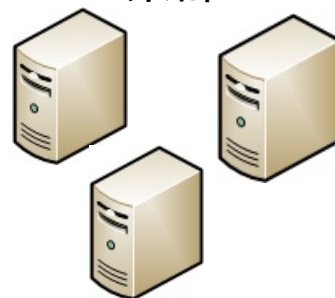
问问搜索广告
集群



图谱广告请求



图谱广告
集群



1 + 1 = 2

服务架构-演进策略

抽取
主干
逻辑

- 所有变现产品均需要，**不可或缺**

+

公共
逻辑
服务化

- 与业务逻辑**关联较少**
- 可独立于主干**自我发展**
- 可依据产品形式进行**裁剪**

+

业务
逻辑
配置化

- **大而全**，基本实现所有产品策略逻辑
- **掩码配置**完成策略逻辑加载
- 逻辑之间关联较少，可配**自由度高**

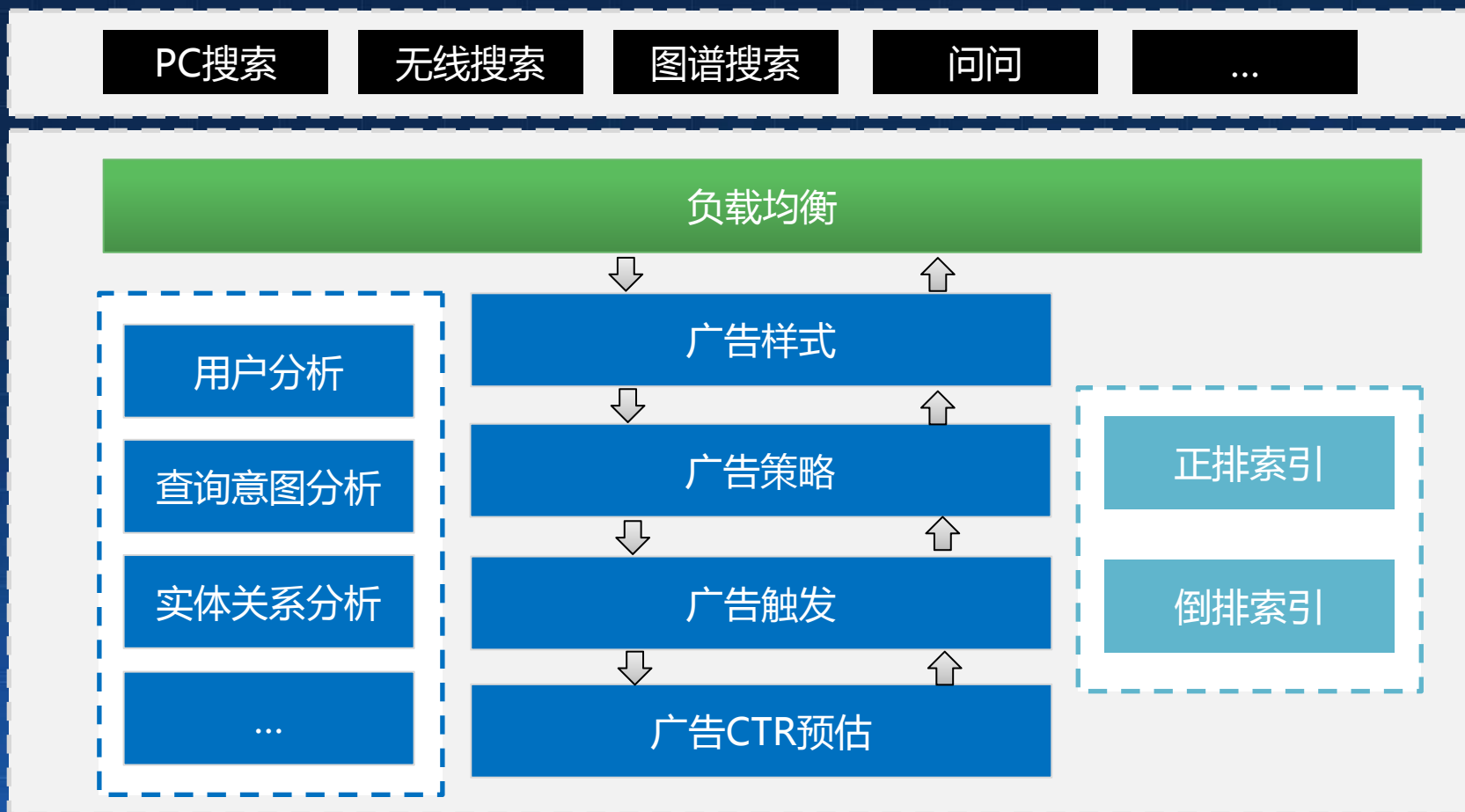
||

服务
架构

- 系统**可大可小**，业务**可复杂可简单**
- 支持业务**快速迭代**
- 依赖关系复杂，**运维成本**增加



服务架构-整体架构

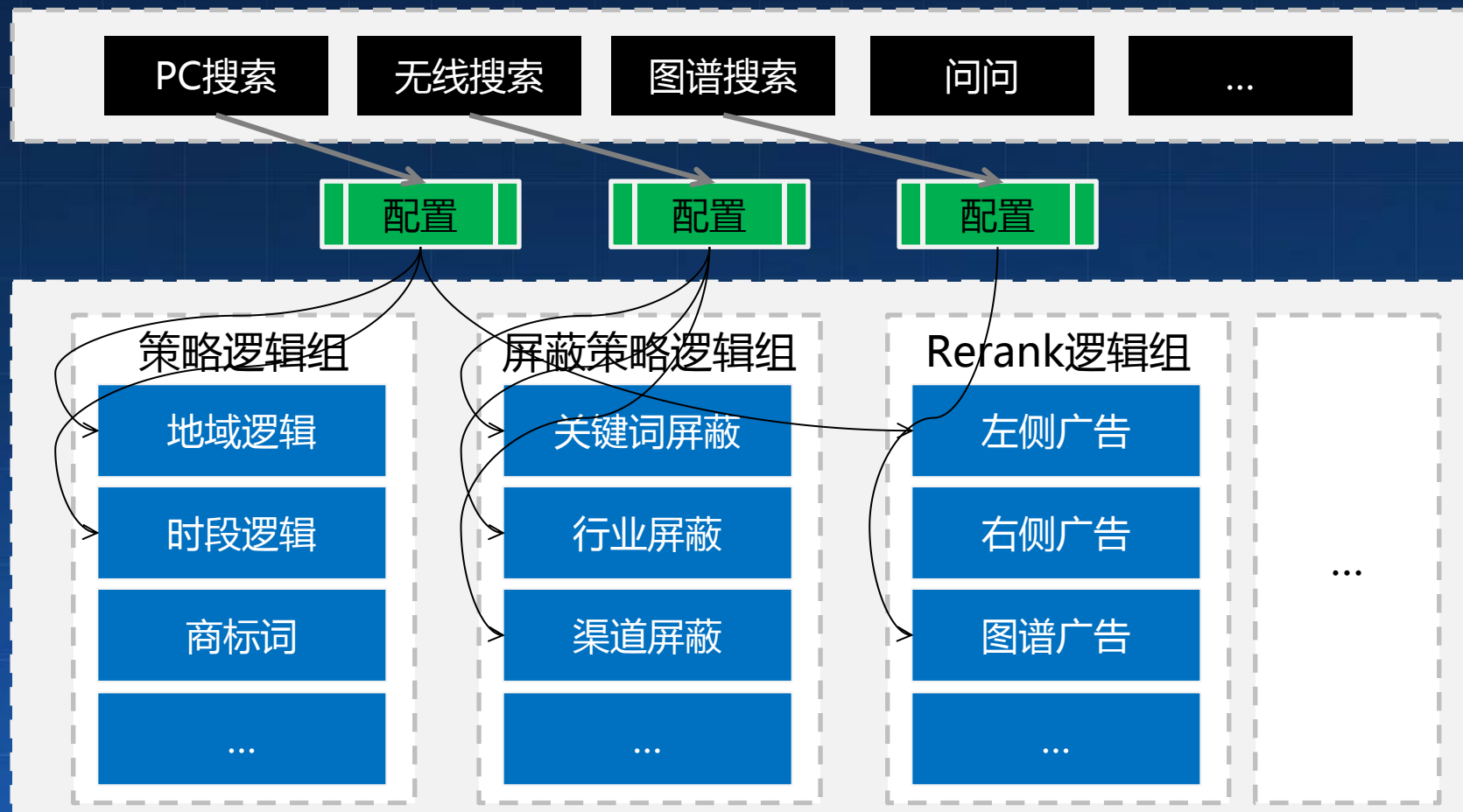


服务架构-公共逻辑服务化

- 如何最有效的拆解出公共分支逻辑？
 - ✓ **对外松耦合，对内逻辑内聚**，一次PV只请求一次
- 如何保证系统整体性能？
 - ✓ **异步总线+任务状态机**



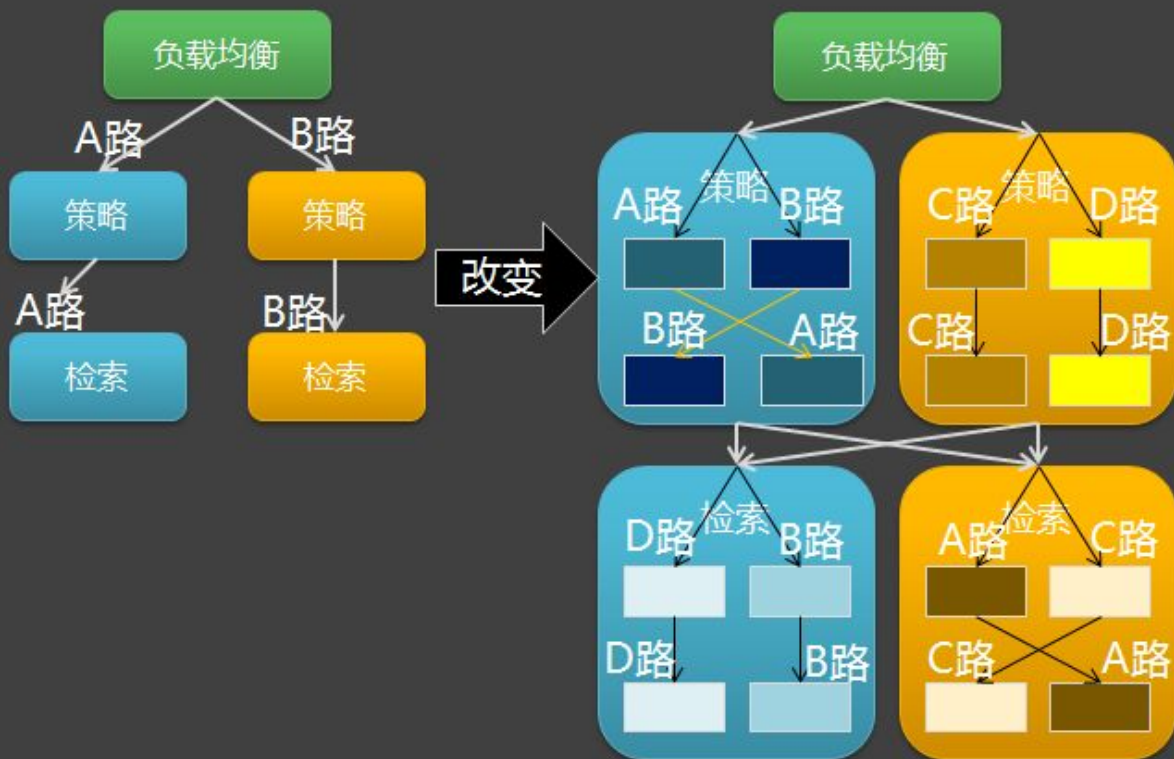
服务架构-业务逻辑配置化



服务架构-流量标签

特点

- 物理切分转变为**逻辑切分**
- 按逻辑切分，实现流量**最大复用**，节省机器
- 定制化、灵活性、多维度、可统计



服务架构-其它关键设计

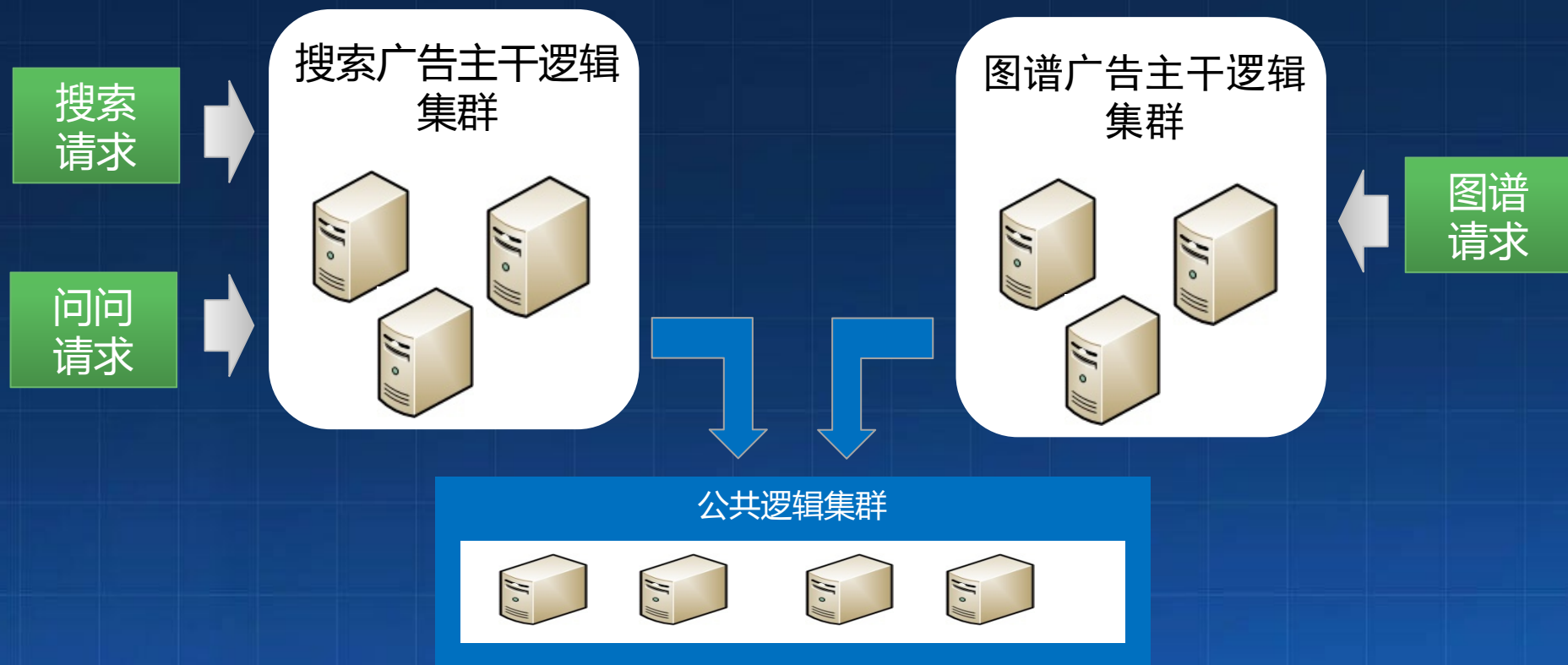
分布式系统故障发现？

- **内外结合**，外部心跳监控+内部阈值判断
- **对上负责**，不仅对下发现故障，还需对上**故障传导**
- 故障时，流量逐层**自动切换**

分布式系统逻辑问题调查？

- 打桩：服务打入**Debug桩**
- 输入：完全**实时模拟输入**
- 输出：自动抓取Debug信息，进行**规则分析**，转化为产品**可读懂规则**

服务架构 - 物理部署结构



1 + 1 < 2

目录

1

背景介绍

2

系统架构的演进历程

- 服务架构演进
- 数据架构演进
- 运维架构演进
- 心得体会

3

展望未来

数据架构-演进策略

数据服务化

公共数据按照**业务特性**切分
以**服务形式**提供数据接口
与业务逻辑分离

+

水平扩展

对数据进行策略切分, 尽可能保证每个**分片均匀**
物料服务 正排索引等水平扩展

+

弹性扩展

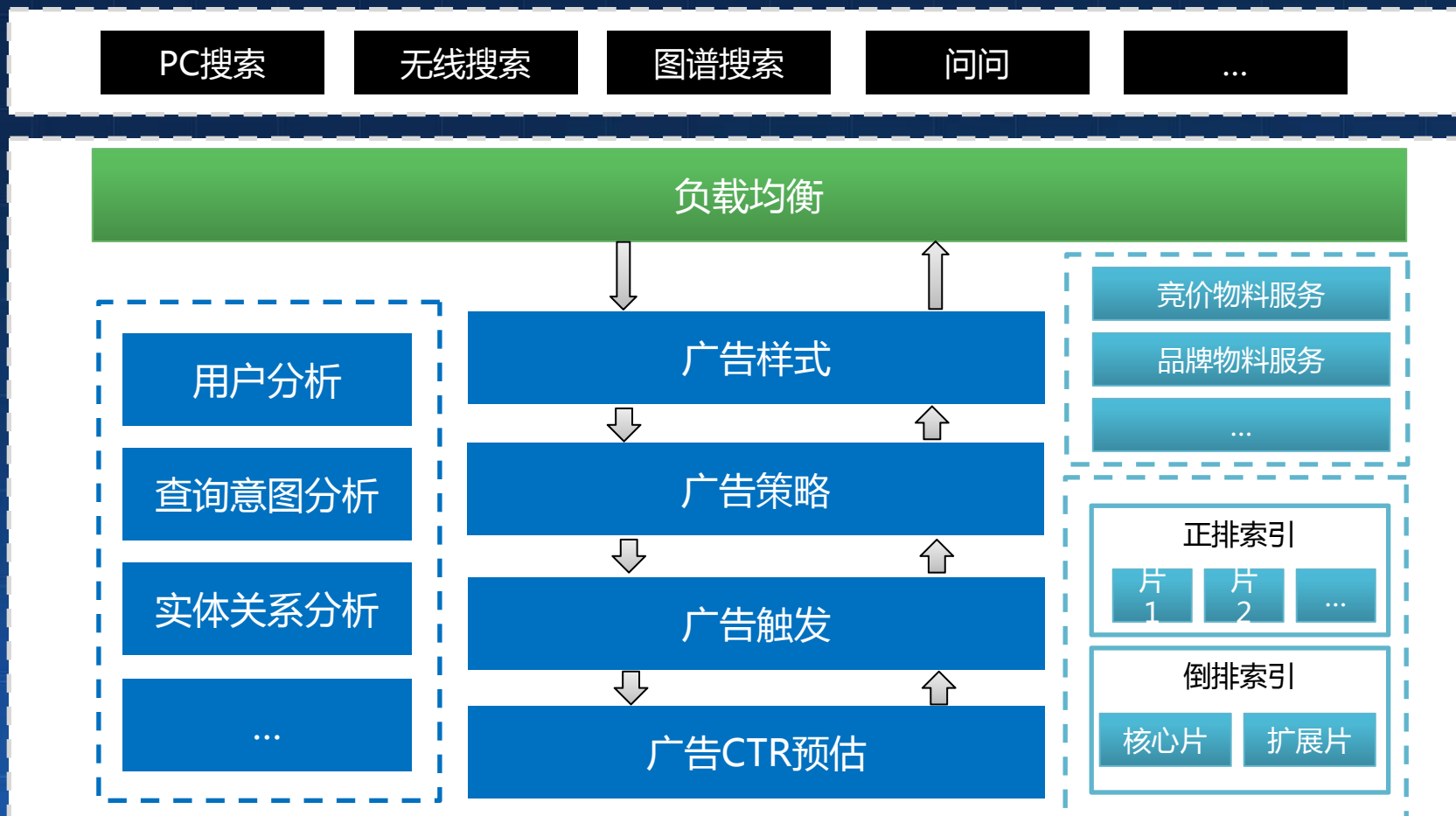
数据分为**核心数据+扩展数据**
核心数据以**最少**的数据量覆盖**最多**业务数据需求
核心数据和扩展数据**实时动态**变化
倒排索引弹性扩展

=

数据架构

数据扩展方式**因势而动**
能够支持海量数据 并能够以**最小的成本**进行扩展

数据架构-整体架构



数据架构-数据服务化

挑战

- 业务服务和数据服务间**通信数据量大**
- 数据服务为所有业务提供数据支持，**请求并发大**
- 系统整体**响应时间增加**

解决方案

- 本地**缓存机制**，命中率85~90%左右
- 请求**过滤机制**，通过对请求的预分类，对不同类型请求分流到不同的数据服务上
- **全异步处理机制**，通过总线和任务状态机架构，异步所有能够并行的请求

数据架构-扩展方式的选择

水平扩展

数据和业务逻辑**松耦合**

机器**内存压力**较大

数据可按指定维度**无限切分**

业务服务和数据服务之间**通信量可控**

弹性扩展

数据和业务逻辑**紧耦合**

机器**计算负载**较高

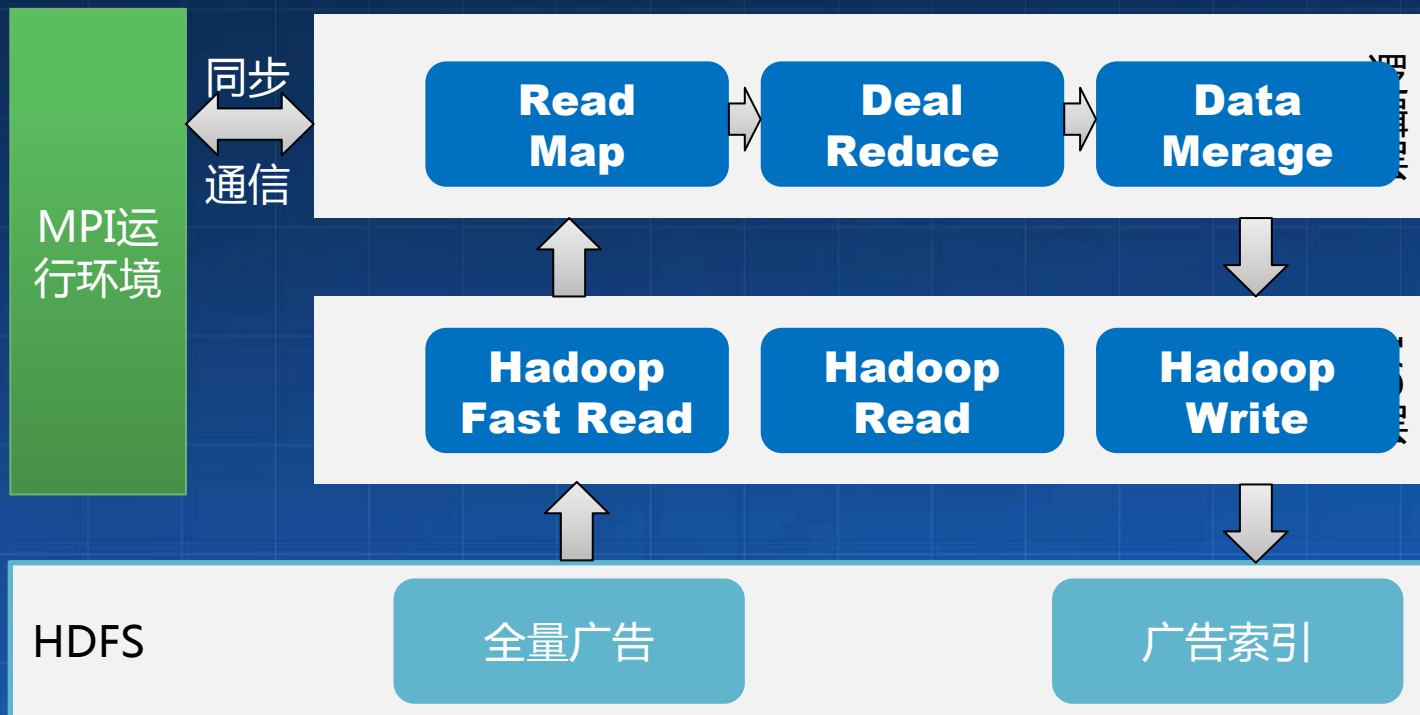
少量数据可覆盖**大量**数据需求

核心数据片可**动态调整**

扩展数据片可支持**较大请求**

数据架构-索引并行制作

- 挑战**
- 数据大，制作耗时长
 - 机器无限扩展，成本大



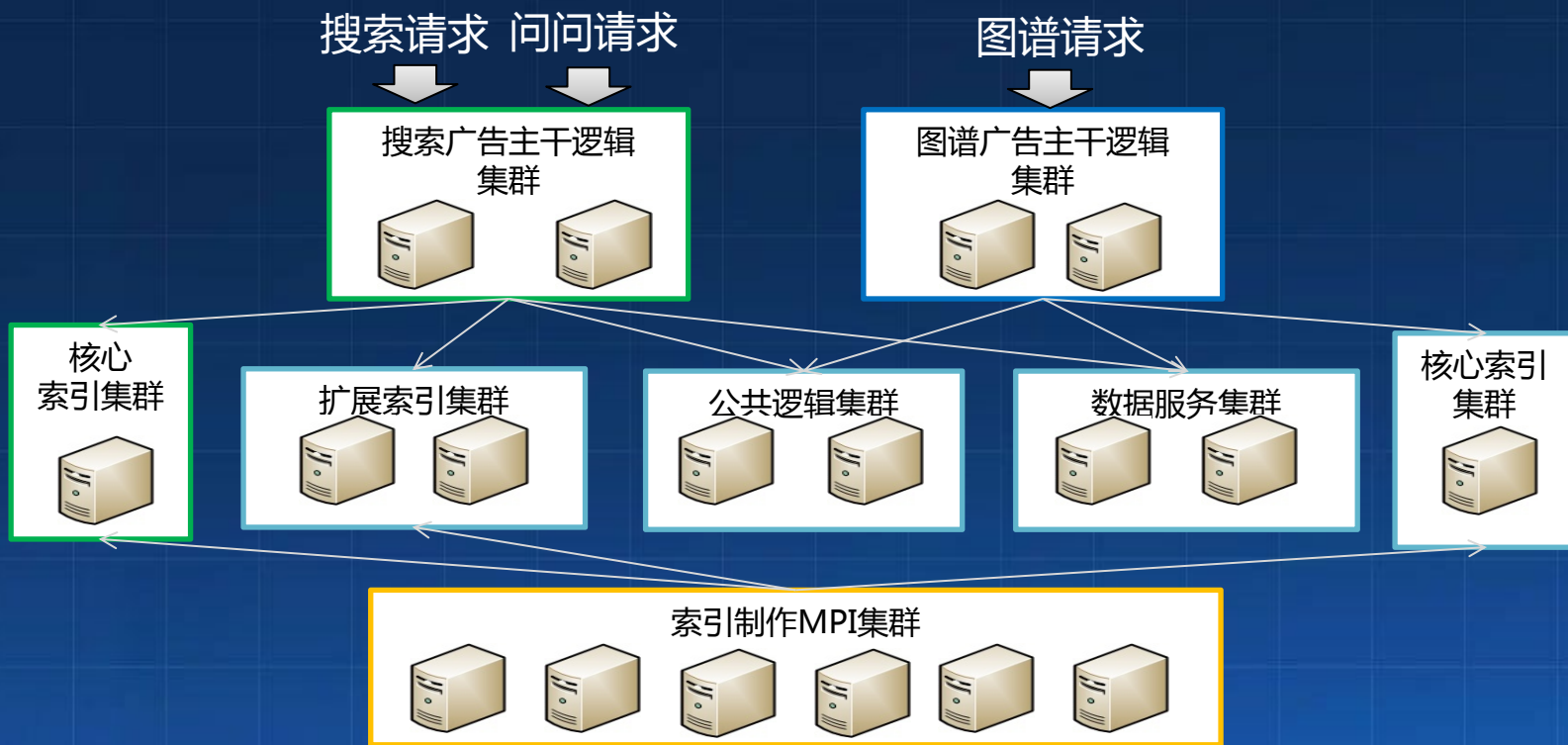
数据架构-其它关键设计

数据本地Cache的选择？

实时数据如何快速更新？

数据如何快速加载？

数据架构-物理部署结构



1 = N

目录

1

背景介绍

2

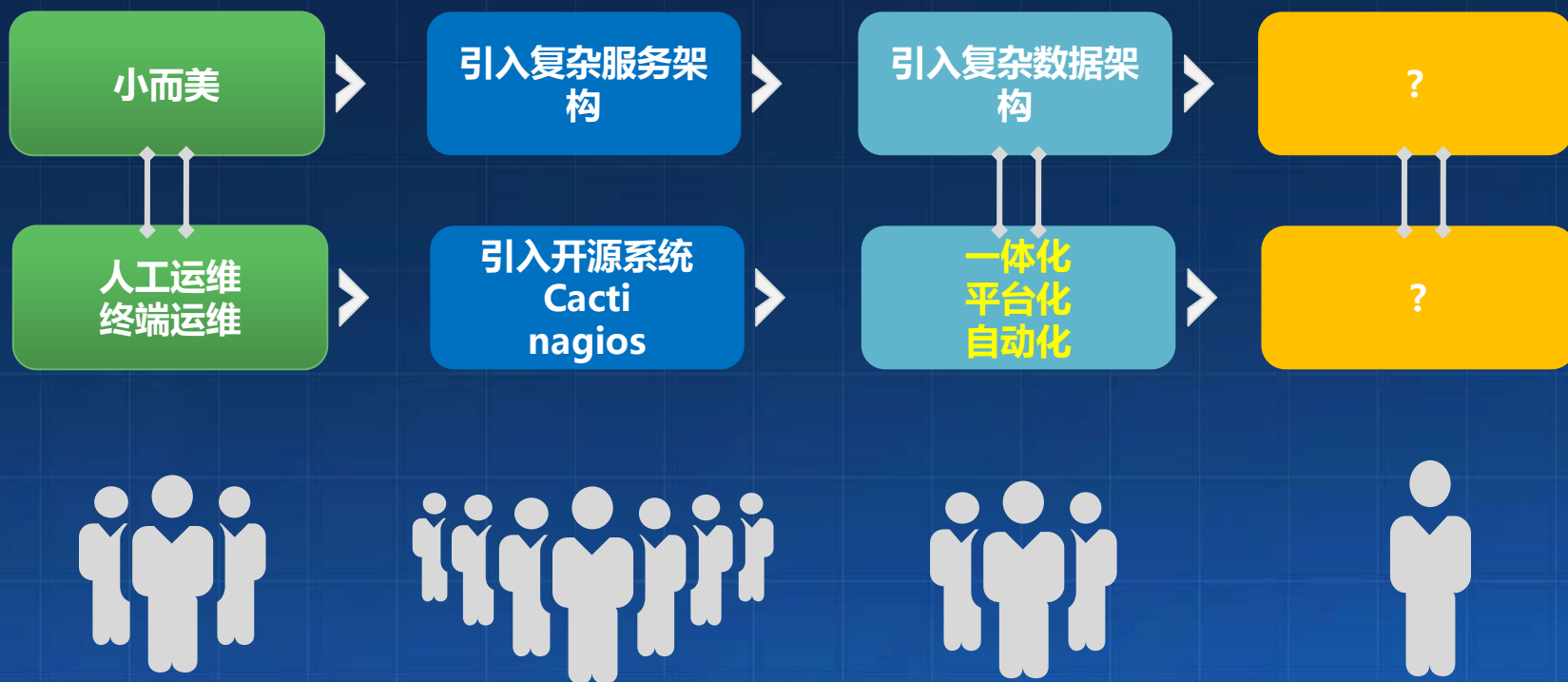
系统架构的演进历程

- 服务架构演进
- 数据架构演进
- 运维架构演进
- 心得体会

3

展望未来

运维架构-进阶演进



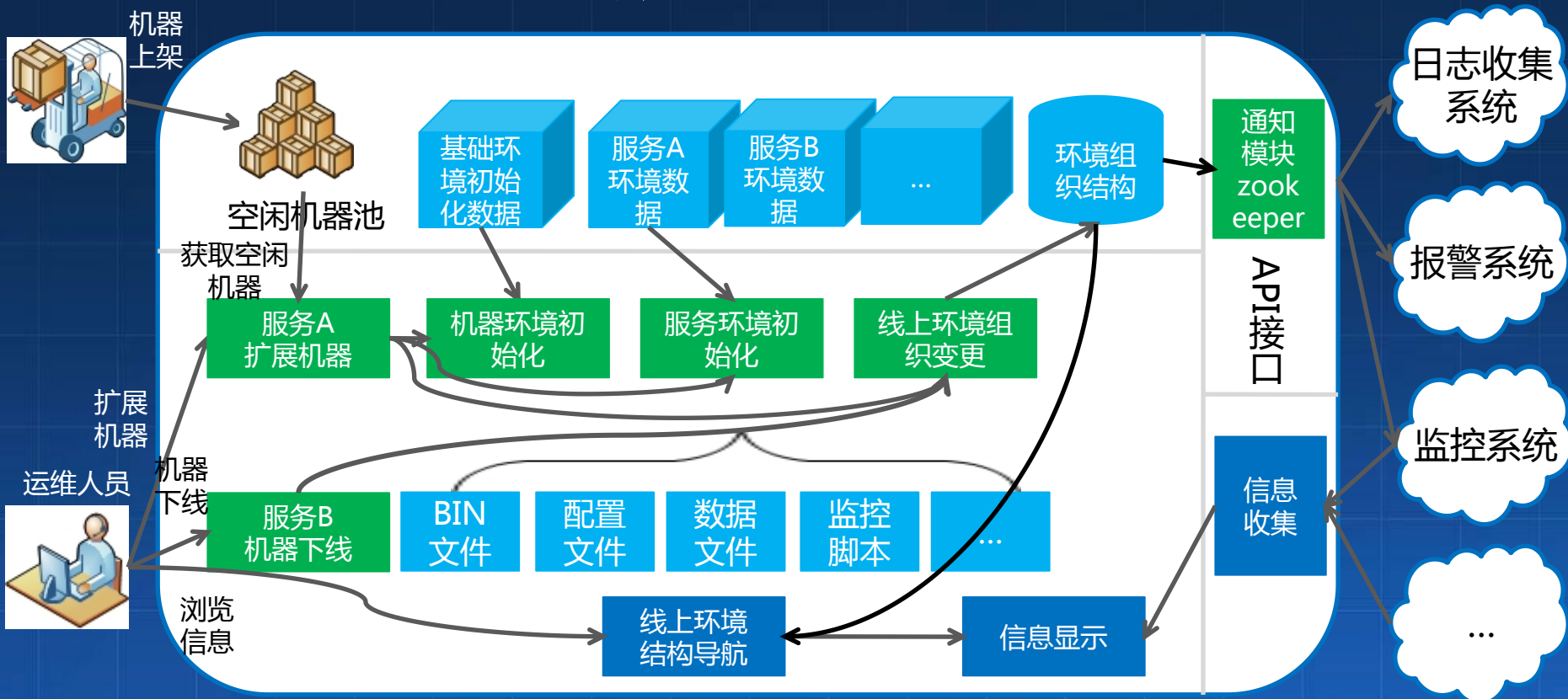
运维架构-一体化平台运维目标

- 基于“**人-平台-对象**”，打破运维系统各自为政、系统间信息独立、目标设备庞大导致管理凌乱无序的局面，实现运维的**统一平台化管理**。



运维架构-以平台为中心的拓扑

凯撒平台



运维架构-凯撒平台上线模式

开发人员按[详细上线单](#)

The screenshot displays the '凯撒平台' (Cesar Platform) interface. The top navigation bar includes '一体化服务管理平台' (Integrated Service Management Platform) and several functional tabs: '机器管理' (Machine Management), '配置管理' (Configuration Management), '数据管理' (Data Management), 'BIN文件管理' (BIN File Management), '权限控制' (Permission Control), '上线管理' (Deployment Management), and '向导式上线' (Guided Deployment). The main content area is divided into two panes. The left pane shows a hierarchical tree view of services under the 'adtech' root, including categories like 'PC_Search', 'main', 'api', 'bing', 'cubic', 'pred', 'cust', 'sh', 'Pr', 'Au', 'sear', 'Mobile_', and 'mair'. The right pane, titled '机器列表' (Machine List), shows a list of machines organized into three rings: 'ring0', 'ring1', and 'ring2'. Each ring contains several machines with IP addresses and status indicators. Above the machine list, there are tabs for 'A', 'B', 'S1', 'S2', 'S3', 'Pre', and 'Cold'. The 'S1' tab is currently selected.

Ring	IP Address	Status
ring0	10.134	...
	10.134	...
	10.134	...
	10.136	...
	10.136	...
	10.139	...
	10.139	...
ring1	10.134	...
	10.134	...
	10.136	...
	10.136	...
	10.136	...
	10.139	...
ring2	10.134	...
	10.134	...
	10.136	...
	10.136	...
	10.139	...
	10.139	...

目录

1

背景介绍

2

系统架构的演进历程

- 服务架构演进
- 数据架构演进
- 运维架构演进
- 心得体会

3

展望未来

心得体会

你的系统真的需要
调整架构吗？

你对整体业务是否
有足够的了解？

你的架构设计是否
具有可延续性？

你的配套运维是否
与时俱进？

目录

1

背景介绍

2

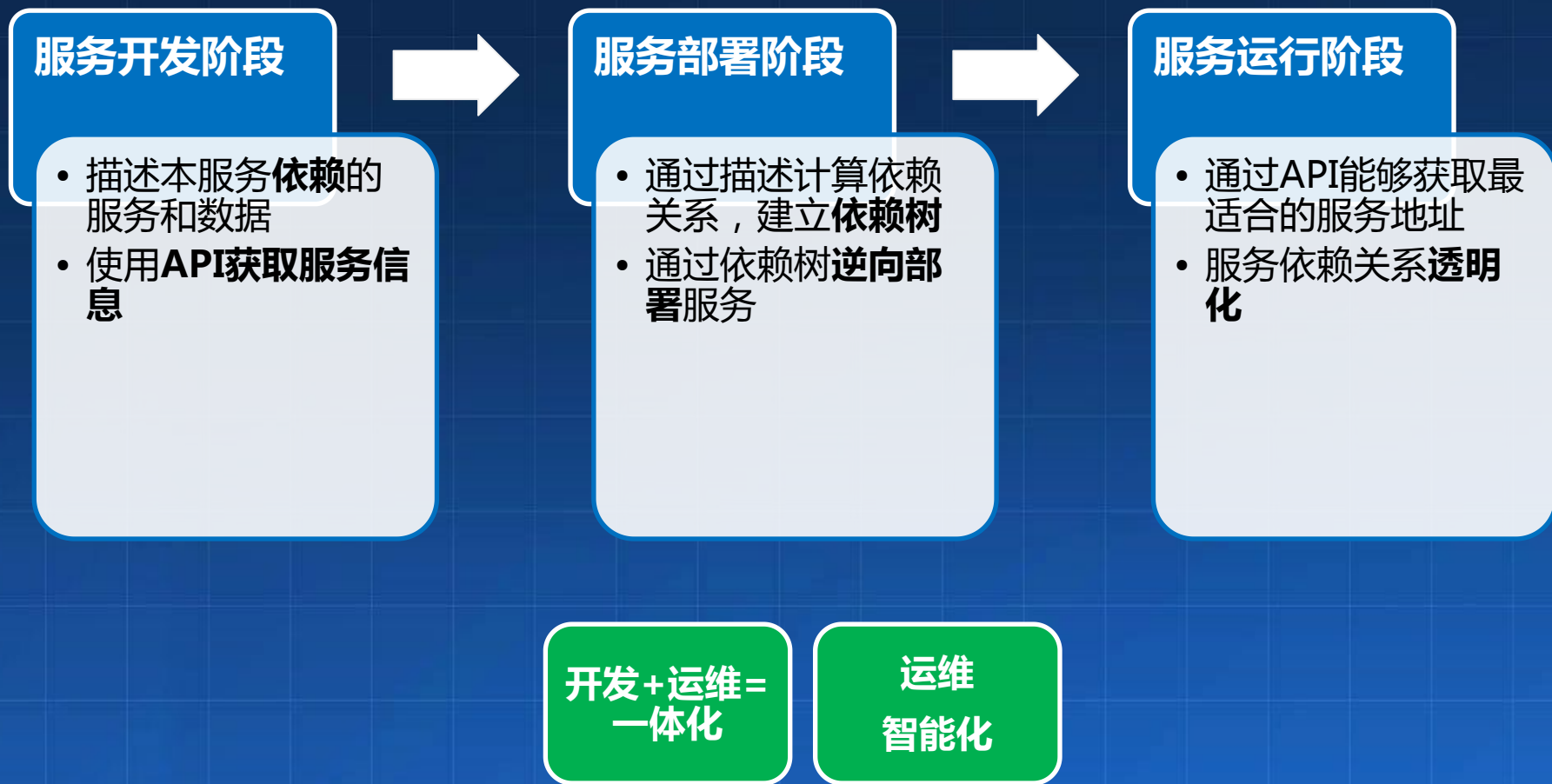
系统架构的演进历程

3

展望未来

展望未来

服务开发与运维进一步结合，达到真正的一体化、智能化



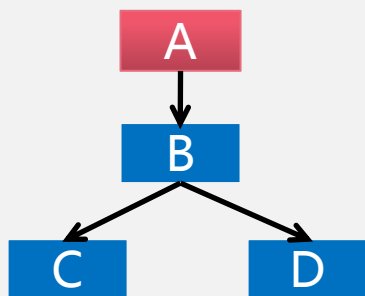
展望未来-场景模拟

开发阶段

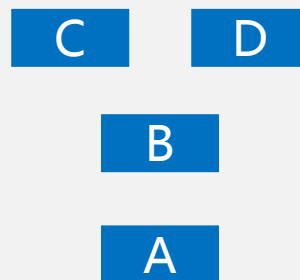
服务A->服务B
服务B->服务C
服务B->服务D
...

部署阶段

依赖树



安装序列层次



运行阶段

服务A->
GetServiceInfo (服务B)
服务B->
GetServiceInfo (服务C)
服务B->
GetServiceInfo (服务D)



THANKS