第 4 周 架构手段

秦金卫/KimmKing

资深架构师

KK架构训练营讲师

讲师介绍

- Apache Dubbo/ShardingSphere PMC
- 资深开源参与者/前阿里巴巴/京东架构师
- 《高可用可伸缩微服务架构》作者
- 《JVM 核心技术32讲》专栏作者
- 《面向性能的系统架构设计》作者
- 阿里云MVP/腾讯云TVP/TGO会员

目录

1. 架构重点细节：技术对比选型，解决方案的制定，现状梳理与差距分析，遗留系统的分布式架构改造
2. 架构的实施落地过程与组织方式：如何推动解决各类技术与非技术问题，架构三板斧

1. 架构重点细节

① 技术对比选型

② 解决方案的制定

③ 现状梳理与差异分析

④ 遗留系统的分布式架构改造

1. 架构重点细节
1.1 技术选型

技术选型：本质是技术决策

「结果（收益）评估」：要回答“要不要做”，希望拿到什么结果，你要从哪几个维度去衡量结果，从哪几个技术/业务指标去验收成果；

「可行性成本评估」： 成本评估包括： 人财物等资源的投入成本

技术成本，主要是技术选型带来的成本，包括：

- 技术实现成本
- 技术升级成本
- 问题排查成本
- 代码维护成本

机会成本：当你把人力、时间花在这件事上，同时就等于放弃了另外一件事，而没有做另外这件事将带来什么样的影响是什么协作成本：多人协作所增加的时间精力开销。一个方案的协作方越多，需要沟通协调的成本也就越高，可控度越低

「风险评估」：技术风险评估，也叫技术风险判断力。即有哪些技术风险需要未雨绸缪，比如该技术选型如果不开源，未来遇到线上问题无法深入代码排查导致问题无法解决影响业务的风险、技术选型对业务侵入性高导致影响既有业务的风险

技术选型：三个关注点

技术：包括语言、框架、工具、设计模式、开发模式等。取其长避其短；关注技术的发展前景。适用于当前项目并且没有太明显短板。技术的生命周期必须显著长于项目的生命周期。

业务：业务初期，灵活、快速。业务发展期，可靠、健壮。业务成熟期，稳定、妥协。

人：独裁式决策，还是民主式决策。团队的基本能力问题。

技术选型过程中最终影响决策的还是人本身

技术选型：几个思考项

1、技术开放性（开源or闭源）：开源意味着，如果有一天你遇到了一个Bug，你至少还有机会通过修改源代码来迅速修复或规避这个Bug，解决你的系统火烧眉毛的问题，而不是束手无策地等待开发者不一定什么时候发布的下一个版本来解决

2、开源社区是否活跃：大部分遇到的 Bug，其他人可能早就遇到并且修复了，你在使用过程中遇到的一些问题，也比较容易在社区或者通过搜索找到类似的问题和解决方案 「学习和维护成本」 即使开源，如果实现的编程语言是相对非常冷门的语言，或者跟自身团队的主要技术栈不match，那么学习和维护成本也会比较高

3、与周边生态的集成和兼容：因为系统是由不同的部分组成的，不同部分需要相互交互和协同，因此不同部分的技术选型需要考虑互相的兼容性，比如Kafka和Flink 就有比较好的兼容性，Flink内置了Kafka的Data Source，使用Kafka就很容易作为Flink的数据源开发流计算应用

4、对原有系统代码的侵入性：如果需要对原有系统代码做较大的改动，则可能会引入风险，进而会对业务造成影响

技术选型：几个思考项

5、技术的边界：了解到足够的知识和经验，我们就知道很多边界在哪里，在边界内思考问题。

6、最佳实践：积累了很多实践以后，我们选型会轻松很多。

7、贴近最核心诉求，并且最好能有机会去验证核心场景。

== 识别出来，哪些是核心的场景。

撮合，清算。

技术选型：流程与步骤


确定选型目标
确认选型范围
确认选型原则
选择关键指标
对比分析
选择核心场景
搭建验证demo
测试验证结果
确认分析结果
对技术项评分
选择最高分数
整理选型文档

技术选型：示例

1、JDK的选择，，，JDK11，

2、数据库的选择

3、Spring的选择

4、Hibernate还是MyBatis

5、缓存的选择

技术选型：示例 - 参考文档

MQ选型：消息队列是分布式系统中重要的中间件，在高性能、高可用、低耦合等系统架构中扮演着重要作用。分布式系统可以借助消息队列的能力，轻松实现多种技术能力。

	RocketMQ	Kafaka	Pulsar	RabbitMQ
开发语言	Java	Scala	Java	Erlang
高可用	主从架构	分布式架构	分布式架构	主从架构
事务消息	支持	支持	支持	支持
集群扩容	增加节点	增加节点，通过复制数据均衡	增加节点，通过新增分片均衡	增加节点
单机吞吐量	高（十万级）	高（十万级）	高（十万级）	低（万级）
消息延迟性	毫秒	毫秒	毫秒	微秒
持久化	支持	支持	支持	支持，性能差
消息回溯	支持	支持	支持	不支持
延迟队列	支持	不支持	支持	支持
死信队列	支持	不支持	支持	支持
优先级队列	不支持	不支持	不支持	支持
消费推拉模式	长轮询 pull	长轮询 pull	push	push
消息顺序性	通过加锁保证消息发送到一个队列，单个消费者线程消费来保证有序	分区有序	独占订阅模式保证消息有序	单线程发送单线程消费来保证
支持topic数量	万级	千级	万级	百级
元数据保存	基于轻量级namesever（AP）	基于zk(cp)/新版本不需要zk	基于zk(cp)	本地保存
技术成熟度	成熟	成熟	接近成熟	较为成熟
社区活跃度	活跃	活跃	活跃	不活跃
银行案例	邮储银行，平安银行，民生银行，网商银行等	各银行的大数据平台消息传输场景	中国银行小范围使用了腾讯TDMQ-pulsar	

技术选型：示例 - 参考文档

数据库选型

1、银行数据库选型要点

- 技术选型的场景区分
  - 技术选型：通用场景
  - 技术选型：分析场景
  - 技术选型：特定场景
- 选型要点与难点
- 选型原则与要求
- 数据库架构模式
  - 架构1：分布式中间件+单机数据库
  - 架构2：原生分布式数据库
  - 架构3：业务自研+(开源)单机数据库
- 数据库建设模式

2、银行常用数据库介绍

- 国内数据库行业发展概况
- 常用数据库介绍
- 常用数据库介绍
  - TiDB数据库
  - OpenGauss/GaussDB
  - OceanBase数据库
  - TDSQL数据库
  - GoldenDB
  - EsgynDB
  - PolarDB
  - AnalyticDB
  - Greenplum
  - CockroachDB
  - YugabyteDB
  - 常用数据库特性对比总结

3、同业数据库选型案例

- 国有大行与股份制案例
  - 工商银行–基于开源自研
  - 农业银行–基于开源自研
  - 中国银行–基于开源+中间件
  - 邮储银行/民生银行–基于开源+中间件
  - 交通银行/中信银行–基于开源自研
  - 交通银行信用卡–分布式数据库
  - 招商银行–分布式数据库
  - 光大银行–双线方案
- 城商行等中小行案例
  - 张家港银行
  - 贵阳银行
  - 北京银行
  - 西安银行
- 互联网银行案例
  - 微众银行
  - 网商银行

4、[redacted]数据库选型建议

- [redacted]数据库现状
- 数据库选型产品对比分析
- 数据库选型相关成本分析
- 金融行业落地实施方法参考
  - 太平洋保险数据库选型落地经验
  - 工商银行数据库选型落地经验
- 数据库选型建议
- 数据库落地建议
- 关注核心技术难点
- [redacted]选型建议总结

1. 架构重点细节
1.2 解决方案

一般意义的解决方案

单纯问题	目标和路径都很清晰，但是可能很难 学好六门功课，就能上清华北大
两难问题	选择和取舍，放弃一部分可能的机会 有两个姑娘想要嫁给你
棘手问题	目标和路径都不明确，错觉它是简单问题 公司很多现状很low

一般意义的解决方案

这里的解决方案，指的是为了某一个具体目标，分析出一系列问题，进而制定的一系列可行性分析，解决问题的思路和实现方案，整体设计，落地方式等一整套的方案。

此类方案很灵活宽泛，但是核心套路都是一样的，不同角色的表述重点。

比如：作为一个技术负责人，根据公司发展和领导建议，考虑一个成立数据中心的解决方案。

比如：作为一个合作方SA，考虑给甲方做一个新核心项目群的整体解决方案。

比如：作为一个传统公司的架构师，考虑做一个是否建议整体IT上云的解决方案。

不同角度的方案出发点和整体思路不同！

解决方案的制定 - 整体思路

此类方案的题目一般较大，问题相对模糊。所以非常考验方案制定人员的问题分析拆解，以及解决更大范围的问题域的能力。

第一步：对齐核心目标 ==》听领导的话

第二步：分析现状数据 ==》我去调研了

第三步：发现关键问题 ==》我找到关键了

第四步：解决关键问题 ==》我解决问题了

第五步：整体方案分析 ==》我考虑全面了

第六步：实施落地步骤 ==》方案是可行的，并且按我说的做就成

第七步：组织协调资源 ==》 找领导要人要钱要地盘要权力

解决方案的制定 - 以第一类举例

背景为一个电商创业公司，老板提出要建设一个数据中心。

你作为技术负责人，如何思考这个问题。

1、老板为什么有这个想法？

2、目标到底是什么？

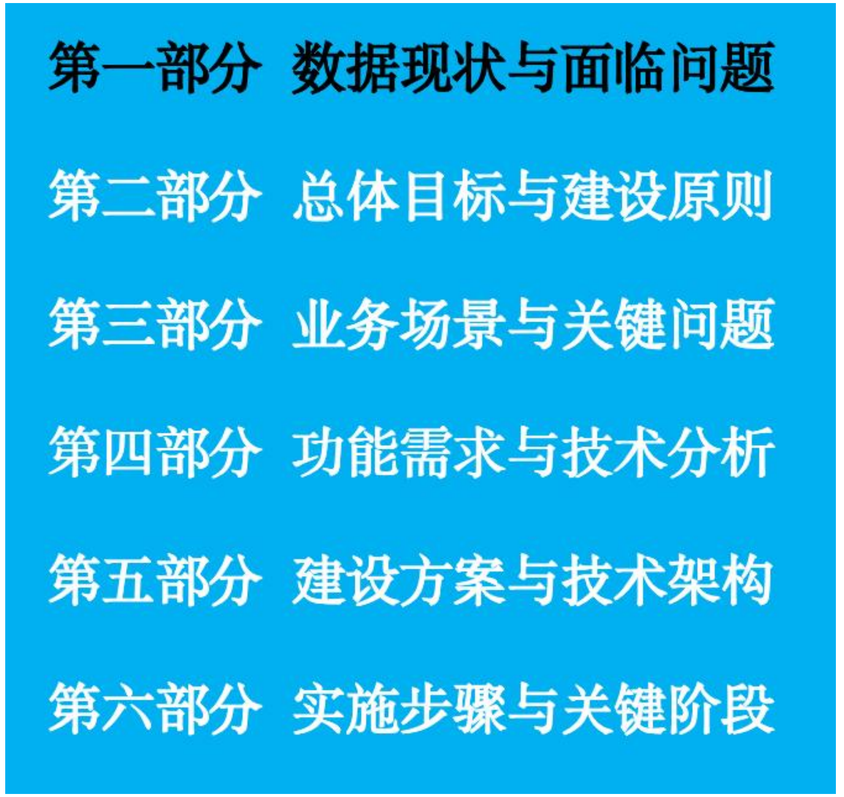
3、你需要帮老板做什么？/为什么是你？

4、怎么来做？

5、涉及到哪些利益相关者？

6、如何快速推动这个事儿？

7、如何让这个事儿达到预期效果？



解决方案的制定 - 以第二类举例

背景为乙方公司的SA，现在甲方某个银行客户想要做一个新核心项目群建设的整体解决方案。

你作为此时售前负责人，如何思考这个问题。

1、客户到底想要什么？

2、我们到底有什么？资源，产品，成功案例

3、客户想要的距离现在有多大差距？

4、这些差距具体转化成了哪些痛点问题？

5、如何解决问题，用我们的方式？

6、为什么是我们，而不是他们？

7、步骤如何操作？

核心是，我们懂客户，并且成功过，所以可以帮助客户成功。

解决方案的制定 - 以第三类举例

传统企业的IT整体上云，留给大家思考，怎么来设计。

要不要上云？当领导说让你做一个上云的方案，目的可能是什么？

现状怎么整理，上云的问题在哪里？如何解决？云上与云下的优缺点？

如何落地：成本，步骤，资源，时间。

解决方案的总结

解决方案，围绕核心目标去设计一套完整的逻辑，通过各个角度，分析拆解，应对核心问题。

体现谁的价值。甲方、乙方。。。==》自己的位置

达成什么目的。参考“向上管理第一项：路径 P 背后的目标 B”：

https://xie.infoq.cn/article/dbb55965401974810a8ba3fc1

详细的推演是为了什么？应对挑战。

1. 架构重点细节
1.3 差距分析

差距分析-差异

方案1：包括A，B，C，D

方案2：包括A，BB，CC，E

存在差异，对比优劣。

业务：提出业务变更。

技术：提出技术变更。

差距分析的重要性

现状1：单体，基础监控，

目标N：微服务，服务治理，可观测性，

跨度太大怎么办？ ==》 制定多个基线。

差距分析的重要性

现状1：单体，基础监控，===》作为基线BaseLine1

方案1：单体，改进监控，自动化运维，===》BaseLine2

方案2：微服务，，，，，===》BaseLine3

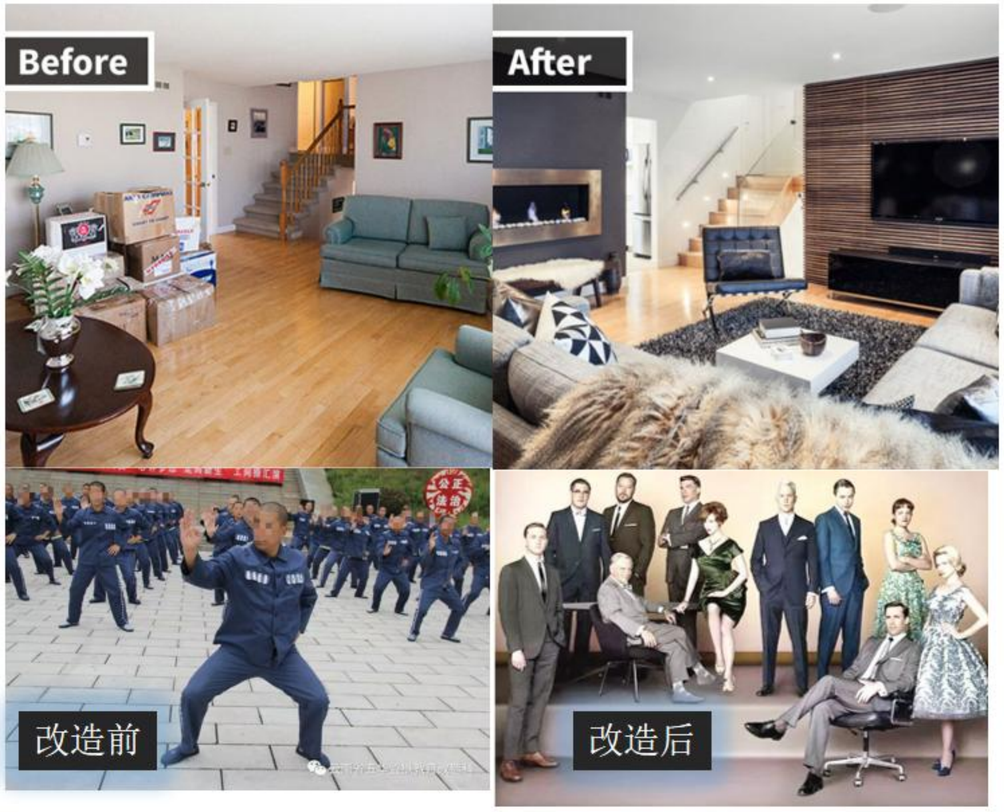
.....

目标N：微服务，服务治理，可观测性，

1. 架构重点细节
1.4 遗留系统

微服务架构的实施案例

最佳实践之一：遗留系统的微服务改造



① 功能剥离、数据解耦

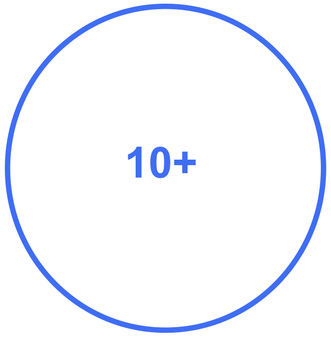
② 自然演进、逐步拆分

③ 小步快跑、快速迭代

④ 灰度发布、谨慎试错

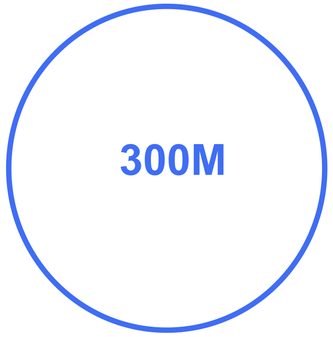
⑤ 提质量线、还技术债

实施案例--某证券金融系统背景介绍



10多条业务线都是基于此核心系统设计开发，造成核心系统功能复杂，业务处理能力弱，对中前台的业务快速响应能力支撑不足。

100万以上的活跃交易用户，用户数据量大，对新用户体验和数据处理能力，有较高要求。当时的核心系统无法满足这些要求。



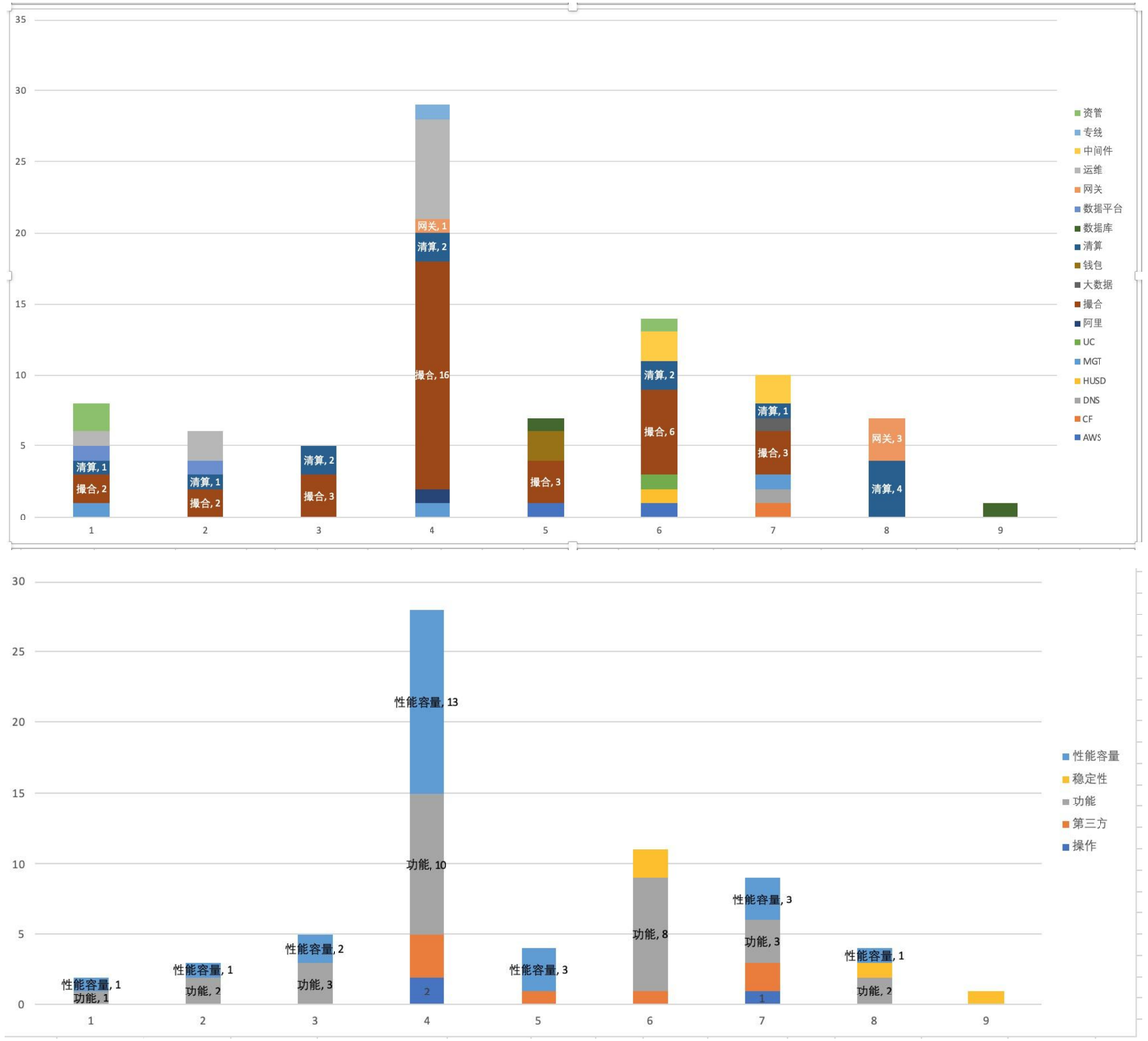
每天交易订单大概在2-3亿左右，要求核心系统有非常强的订单交易处理能力、账户和资金的清算能力，当时系统无法做到分布式。

每天的交易总金额超过100亿美金，对系统的稳定性和数据的一致性也有强烈的要求。当时的系统无法满足可用性和一致性要求。

架构改造--提效增速，中台化建设

重新梳理和定义系统架构和业务架构，整合各项能力中心，作为微服务架构改造和实施的基础指导。

架构改造--混沌工程，稳定性建设



-99%

找到哪些系统是瓶颈

找到经常出问题的系统，时间，人员，进行写分析和复盘。对过去三年的COE进行分析，找到瓶颈因素并分类应对。

找到哪些因素是瓶颈

引入全链路监控，波测等技术手段，以业务指标+技术指标为指导。增强CICD自动化能力，引入灰度、蓝绿、金丝雀等多种发布机制。

从机制上保证稳定性

减少99%的稳定性问题

架构改造--夯实底层，基础性建设

引入新技术解决问题

- 拆分数据库，提升容量
- 消息中间件，缓冲处理
- 内存化处理，降低延迟
- 引入容器化，弹性伸缩
- 数据分层化，分级治理

架构改造—最终效果

前两期改造完成后

吞吐量 50x

同时延迟降低到原来的1/20

每个系统维护成本降低

业务满意度提升5倍以上

系统稳定性大大增强

可用性从两个9到99.95%

系统性能有了质的飞跃

性能造成的客诉等问题基本消失

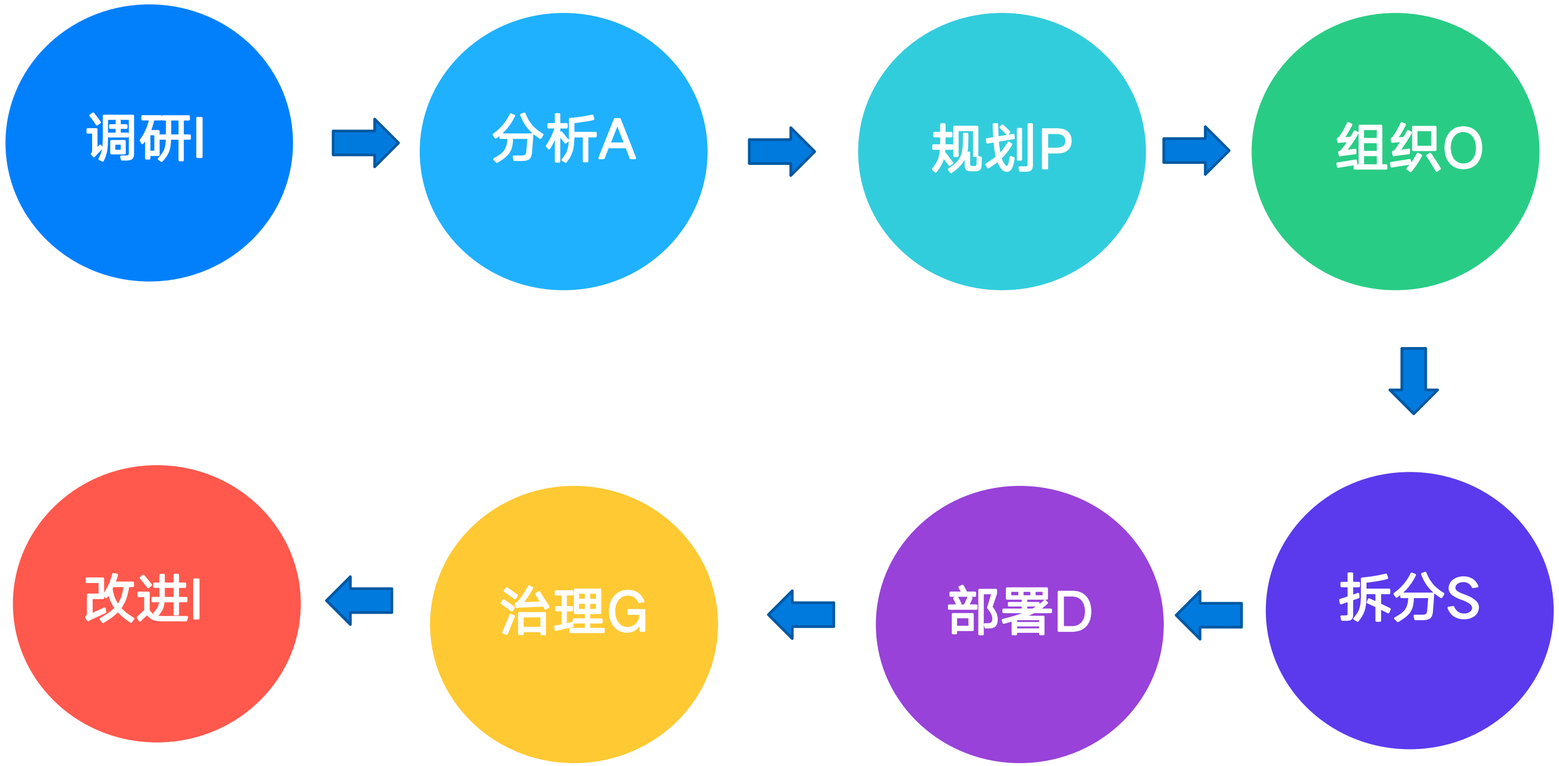
2. 架构的实施落地过程与组织方式

① 如何推动解决各类技术与非技术问题

② 架构三板斧

2. 架构的实施落地过程与组织方式
2.1 推动问题

架构实施落地的通用过程方法--I6I



架构实施落地的通用过程方法--I6I—准备阶段

- 调研Investigation
了解系统现状和各方诉求

- 分析Analysis
明确核心问题和改造目标

- 规划Plan
制定架构改造阶段和计划

- 组织Organization
协调系统相关团队和资源

- 拆分Splitting
根据规划进行服务拆分

- 部署Deployment
引入自动化容器化部署

- 治理Governance
实现微服务治理和管控

- 改进Improvement
基于以上步骤迭代优化

合理的选择，就是最优选择

团队的技术栈

技术发展趋势

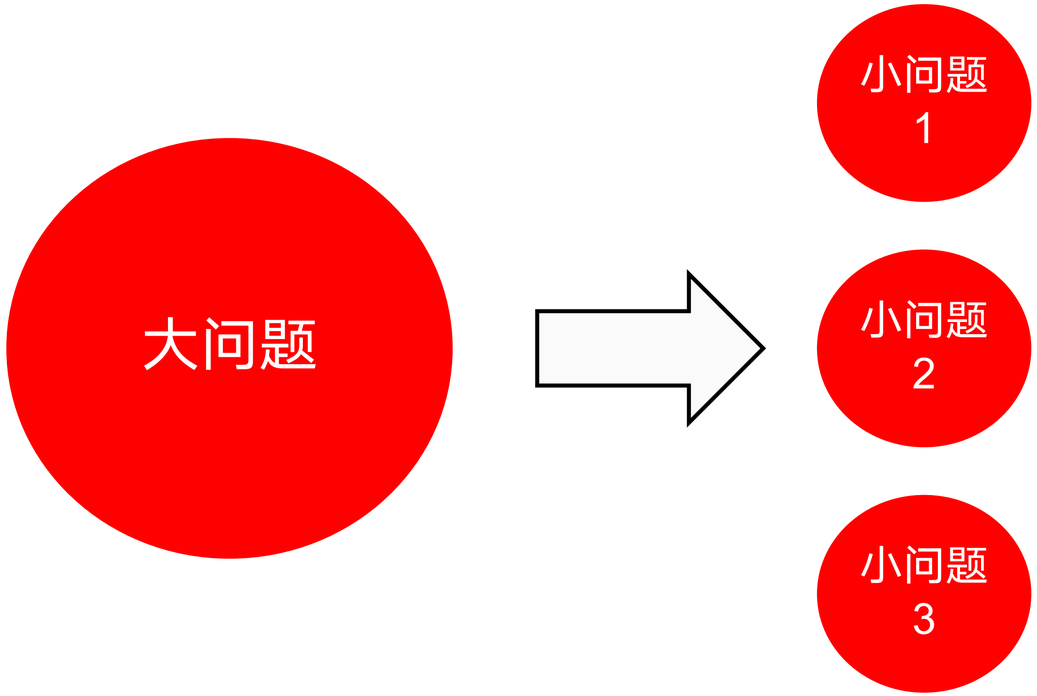
开源与国产化

2. 架构的实施落地过程与组织方式
2.2 架构三板斧

架构三板斧

1、拆分问题

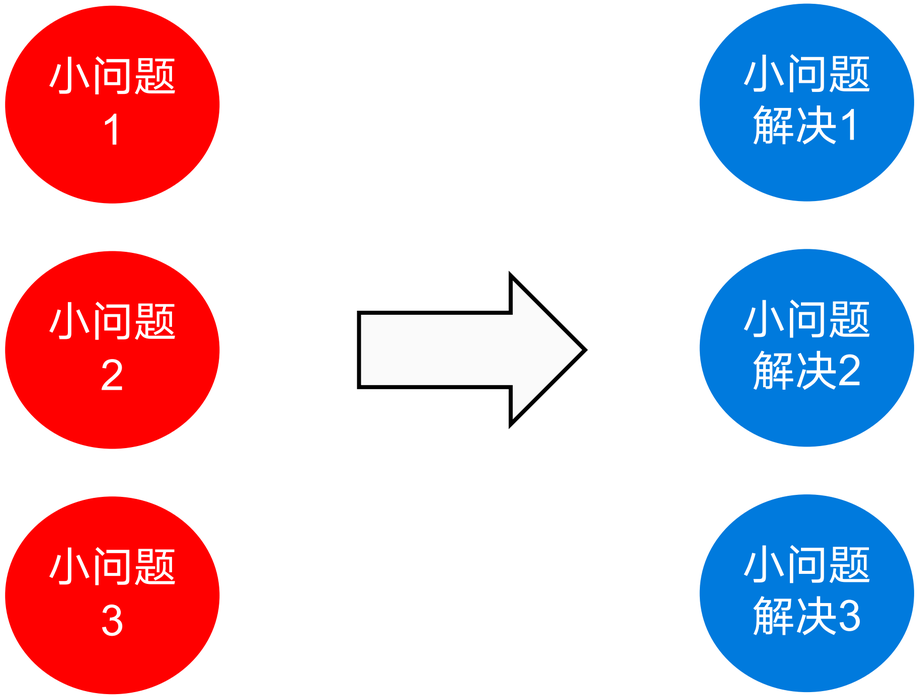
问题的拆分能力，是架构师的第一能力



架构三板斧

2、解决问题

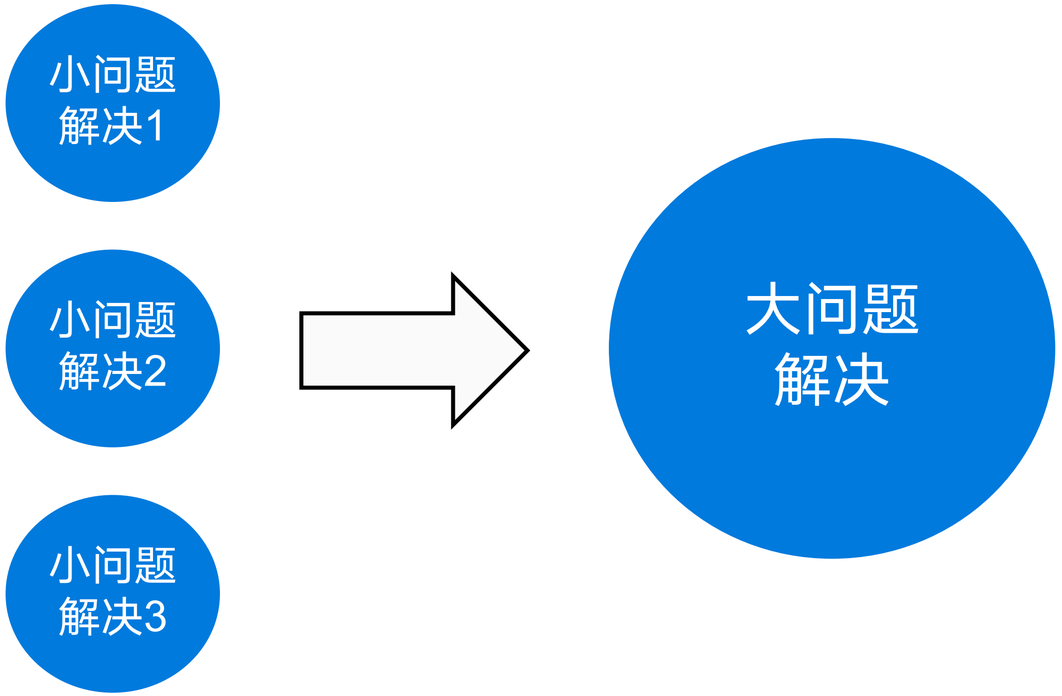
问题谁解决的不重要，关键是所有的问题，都找到解决办法



架构三板斧

3、组装问题

把所有的问题和解决办法，再拼装回去，变成完整的架构设计



第四周-作业实践

1、写一份自己工作场景下的某个技术选型报告

Q&A

谢谢各位

kimmking@163.com