CODE A BETTER LIFE
一 行 代 码　亿 万 生 活

直连低可信系统
实现高可用服务实践

郑旭

美团

郑旭

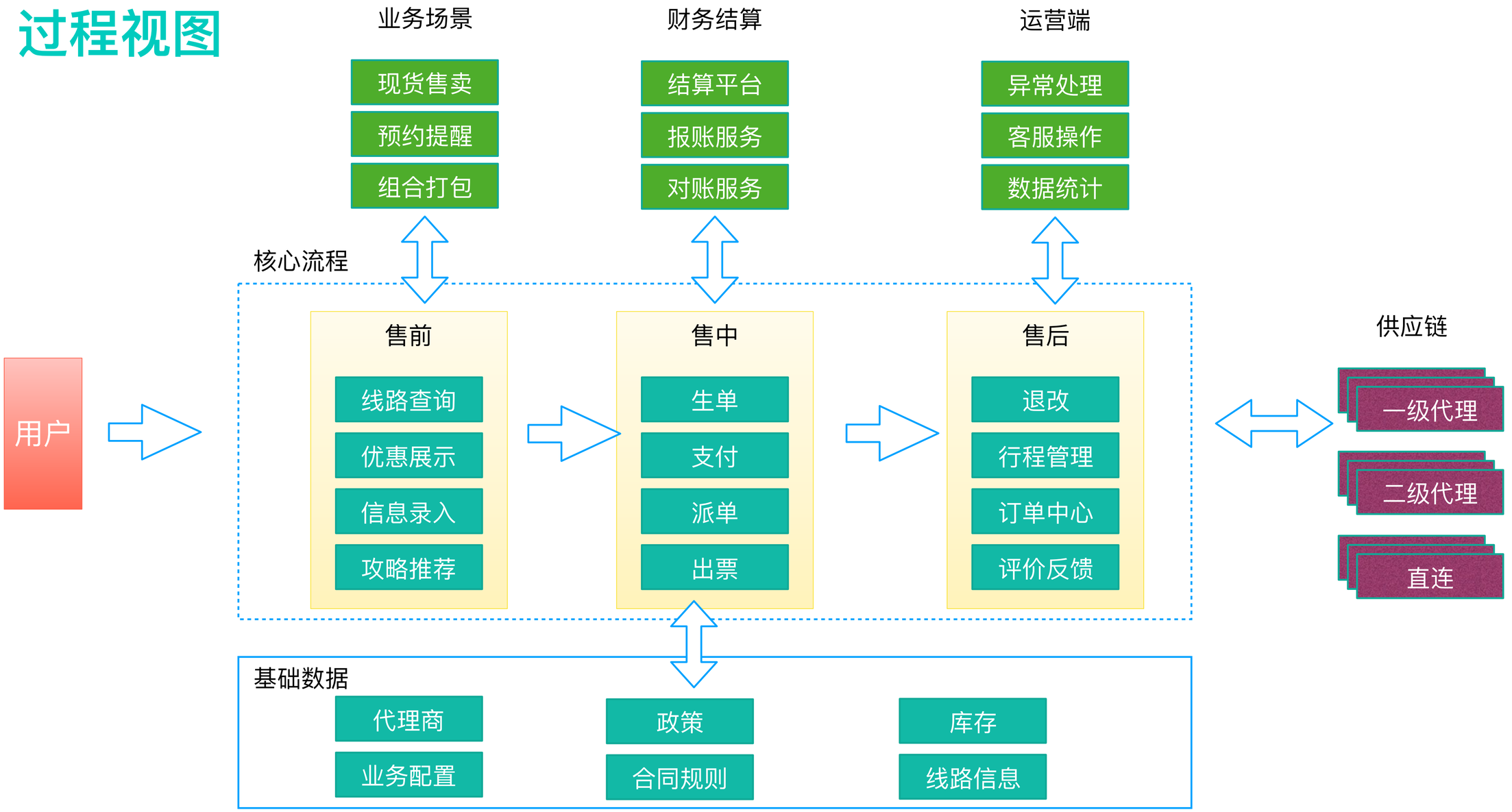
高级技术专家

大交通业务系统架构师

目录

业务架构

过程视图


业务场景
财务结算
运营端
现货售卖
预约提醒
组合打包
结算平台
报账服务
对账服务
异常处理
客服操作
数据统计
核心流程
用户
售前
线路查询
优惠展示
信息录入
攻略推荐
售中
生单
支付
派单
出票
售后
退改
行程管理
订单中心
评价反馈
供应链
一级代理
二级代理
直连
基础数据
代理商
业务配置
政策
合同规则
库存
线路信息

业务架构

逻辑视图

数据来源

- 代理商录入
- 代理商API
- 直连

价格规则

- 票面价
- 服务费
- 儿童学生价
- 优惠组合

退改规则

- 合同定制
- 通用方式
- 混合方式

结算规则

- 销售消费
- 未消费退款
- 已消费退款
- 佣金搭售

售卖规则

- 销售时间起
- 销售时间止

搜索规则

- 缓存策略
- 数据来源
- 索引规则
- 筛选策略

派单规则

- 黑白名单
- 指标模型
- 调度方式

出票方式

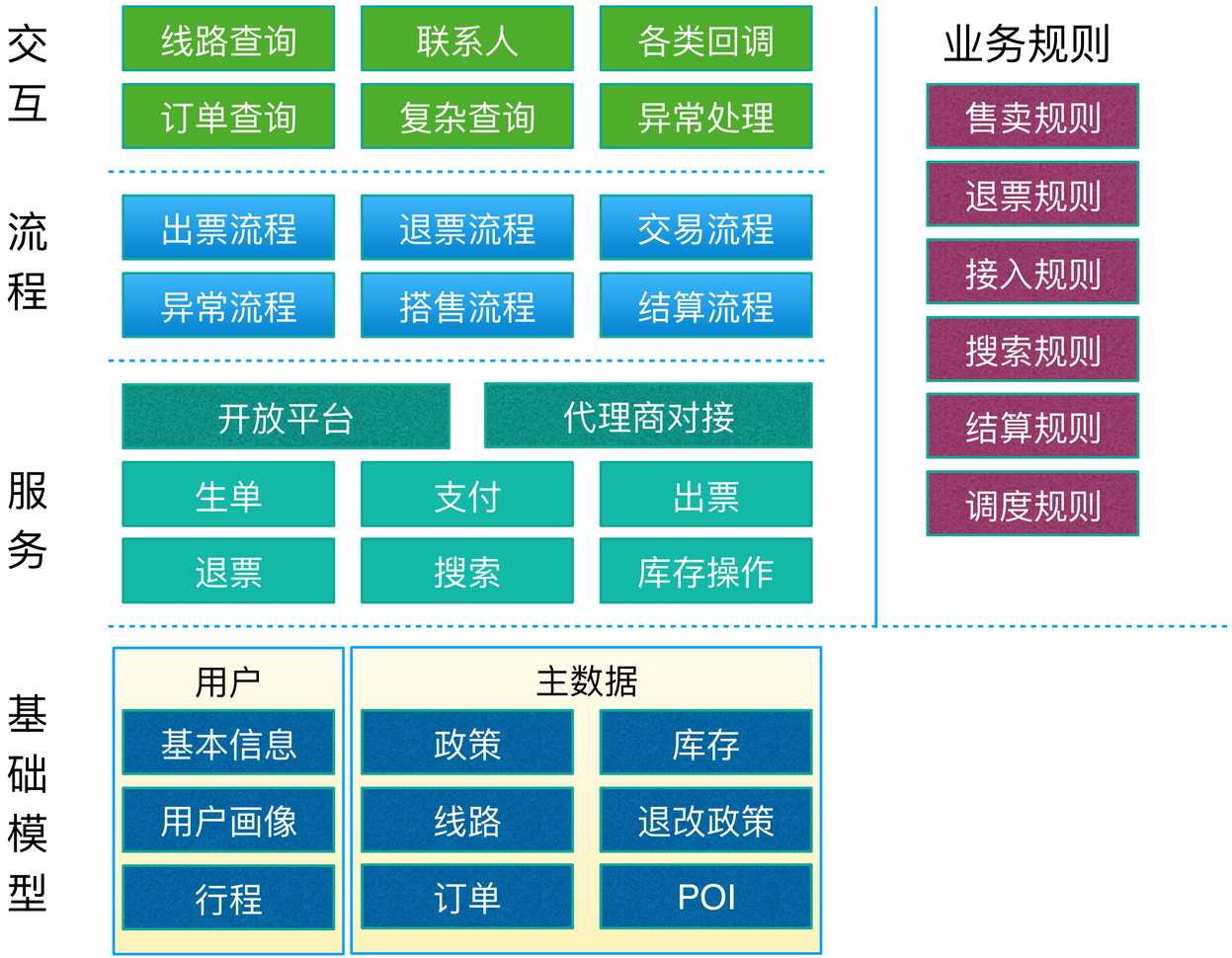
- 纯线上API
- 运营界面
- 纯线下

交互方式

- 同步接口
- 异步回调
- 轮询

业务架构

架构分层



接口：统一处理C端交互、第三方回调

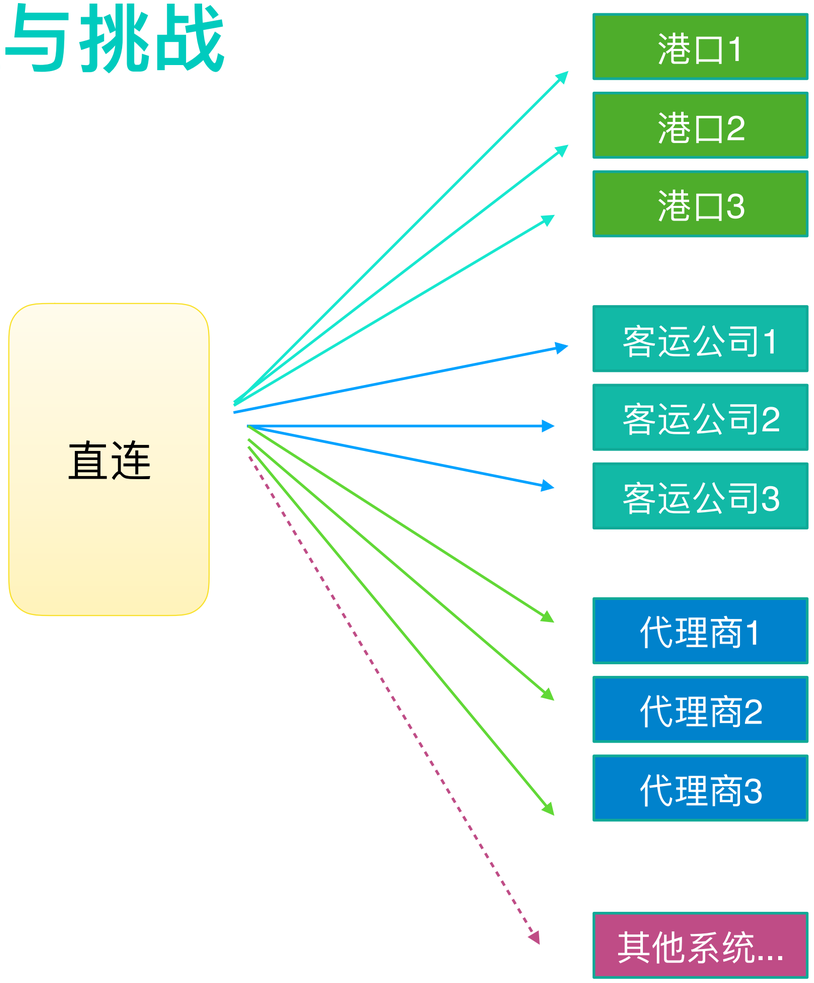
流程：以交易、出票、退票、为核心流程，异常流程提供容错保障

规则抽象、统一处理：
统一数据接入，统一处理各路渠道的数据

目录

业务架构

问题与挑战



常见问题

- 这么多第三方系统都单独接一遍，太痛苦
- 下次改动还得再来一遍
- 第三方经常挂，接入的多了，无时不刻在运维
- 第三方经常出不了票，用户、产品经理、老板都在抱怨
- 被各路人diss：别人家能出票，为啥我们不能

业务架构

问题与挑战

特征

- 供应商很多，长尾明显，接入和运营效率低
- 协议和语言不统一
- 流程复杂、不统一
- 峰值流量冲击供应链

核心问题

- 系统封闭
- 接口低可用
- 接口多变
- 限流不明确
- 共享库存

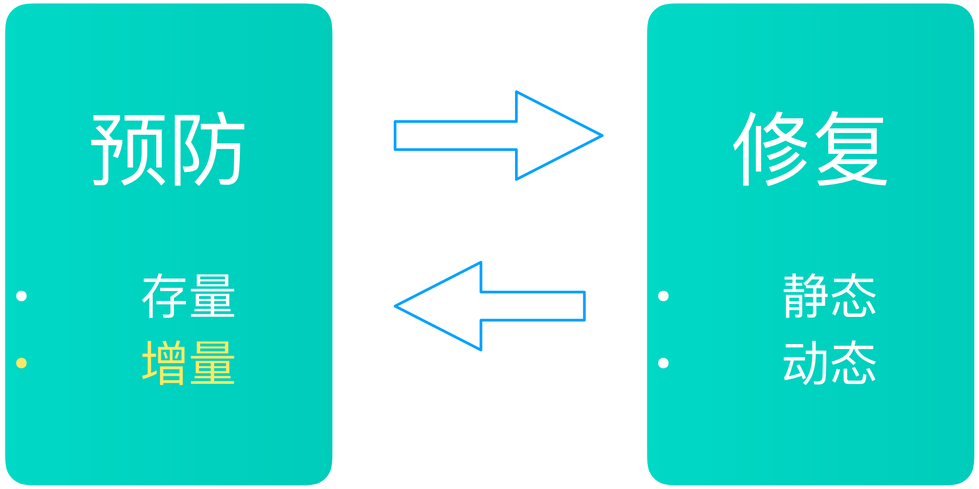
难点

- 多因素相互干扰怎么办
- 峰值能力如何最大化

目录

解决思路与方法

可用性解决思路——接口变更



预防

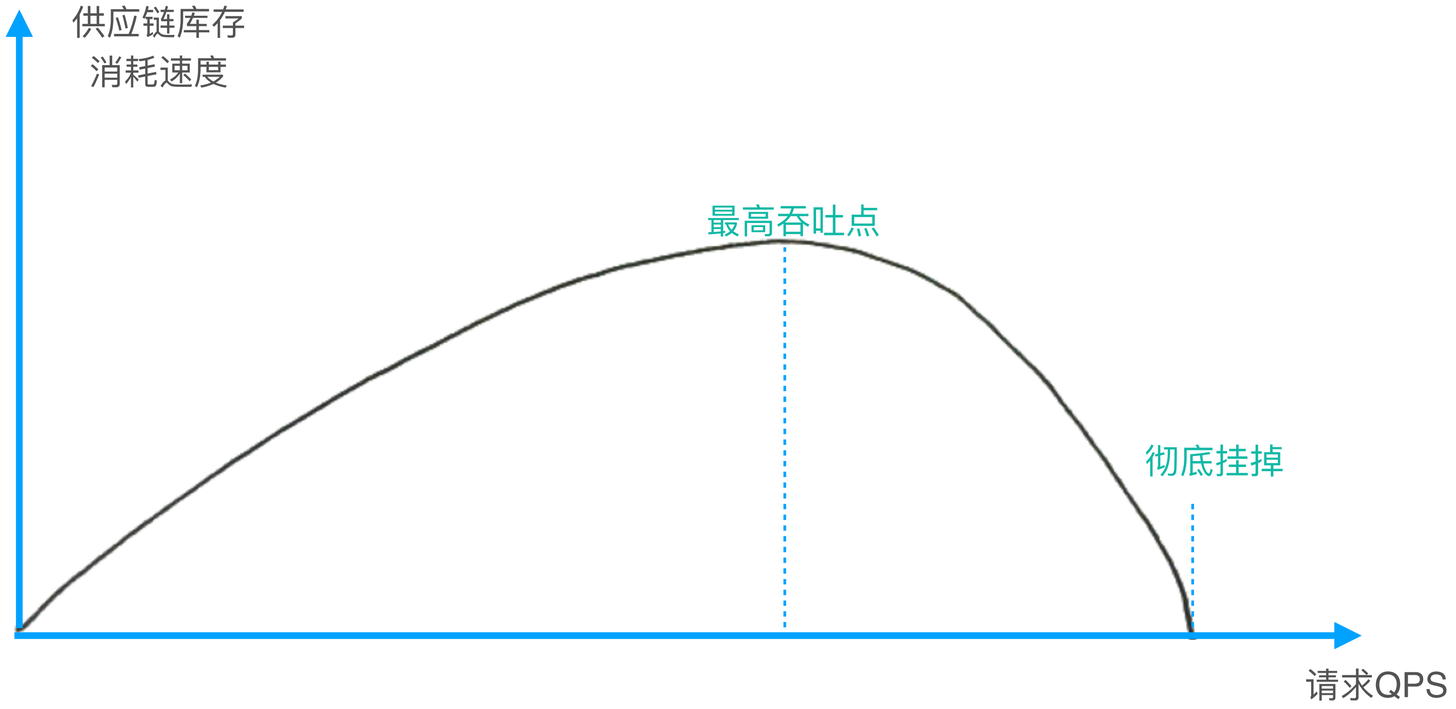
- 存量接口变更易监控
- 增量接口发布难监控
- 监控告警、降级、压测、演练

修复

- 静态修复较慢，但稳妥
- 动态修复较快，易埋坑

解决思路与方法

可用性解决思路——峰值场景



近弹道曲线图（电梯球弧线）

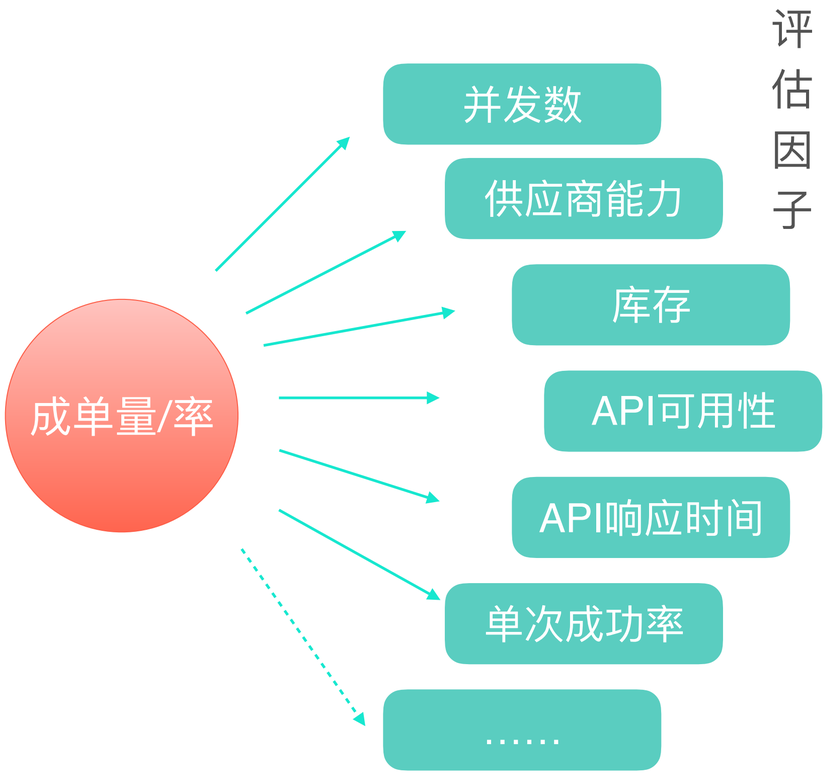
目标分解

目标

限流阻碍	→	最小化被限流
流量尖峰	→	自适应扩容
共享库存	→	最高速消耗库存

解决思路与方法

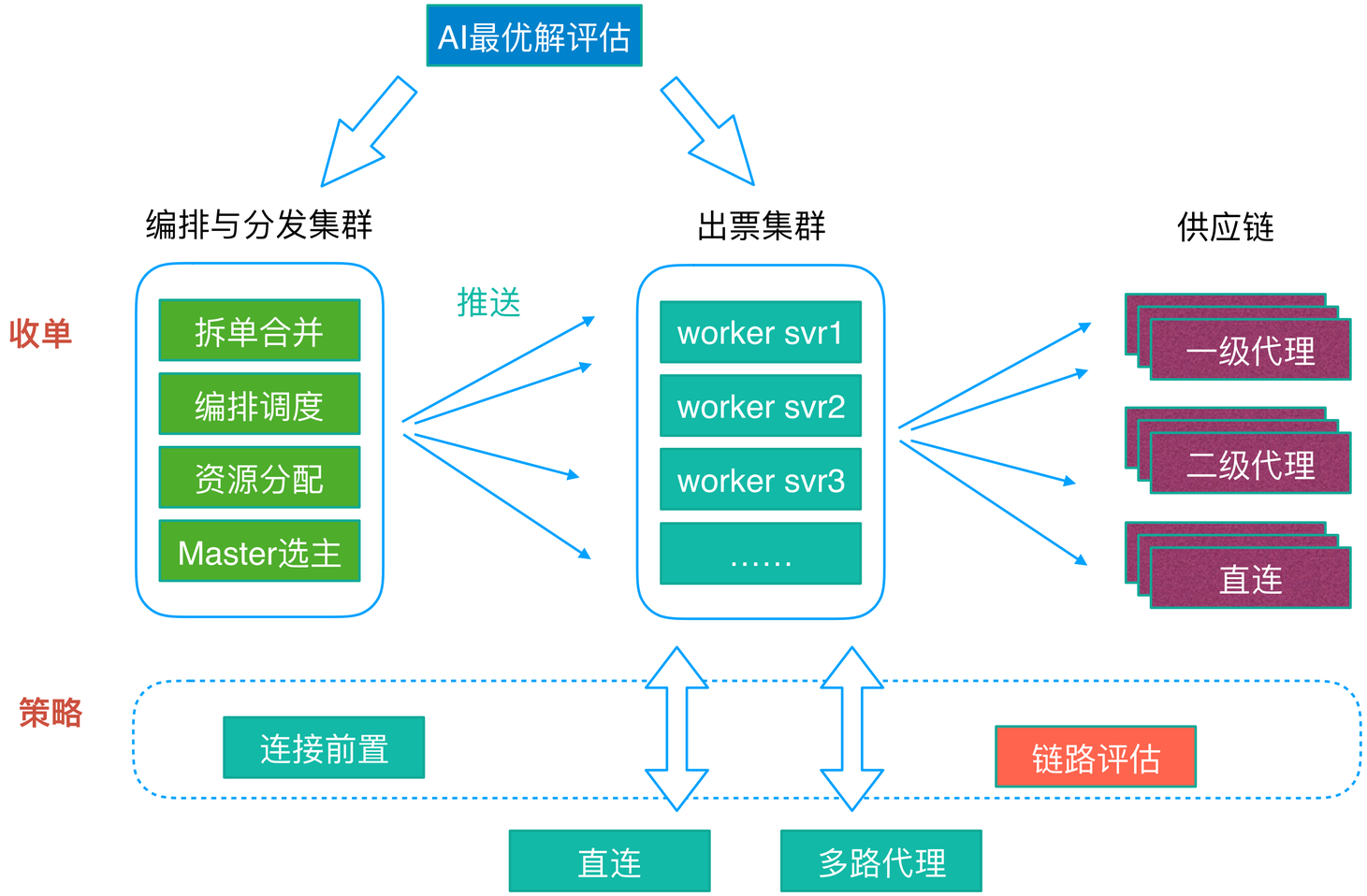
峰值场景-最优解评估



- 确定评估指标：大指标，各小指标，优先满足大指标
- 模型训练：收集评估因子数据，利用机器学习，训练模型，找出相关性
- 指导反馈：多维度指标联动，不断反馈和预测，动态调优模型

解决思路与方法

架构方案



大数据处理分布式模型

- 类MapReduce架构
- Master负责任务编排和分发
- Worker专注于执行

链路评估

- 全链路异步trace
- 自定义业务级traceID
- function级粒度
- 流量回放
- 数据+逻辑diff
- ABtest

解决思路与方法

研发流程辅助

A 开发

需求评审

设计review

代码review

B 测试

功能测试

回归测试

全链路压测

Sonar静态扫描

C 发布

分支合并

发布计划

灰度与回滚

低峰发布

D 运维

链路监控

值班管理

常见问题FQA

熔断降级

解决思路与方法

服务运维能力完善

系统能力完善

- 隔离：超时、重试
- 冗余：无单点、多机房、N+1
- 降级：保核心、分级
- 过载保护：熔断、限流
- 服务治理：服务化、链路监控

日常巡检

- 容量规划：20，5，3
- 链路压测：环境构建、极限测试
- 故障演练：SOP
- 定期梳理：关键系统、核心指标

故障检查处理

- 事前：监控告警、容量规划、预案
- 事中：问题定位、回滚、降级、扩容、过载保护
- 事后：评估影响、解决方案、总结提高

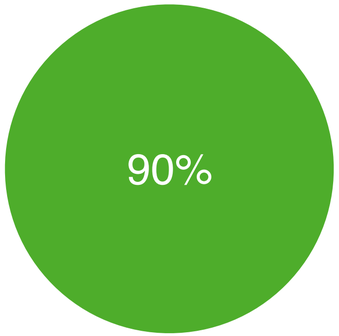
解决思路与方法

实践中一些问题

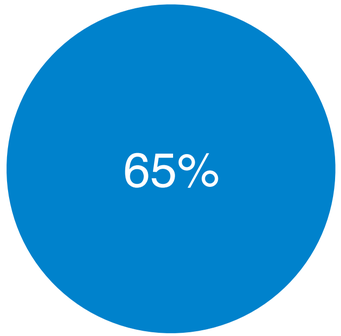
序号	分类	问题	优化方法
1	redis缓存	大Key	本地缓存+定时刷新
2		热Key	把一个Key拆分成多个Key，流量打散
3		写QPS过高	将分布式锁操作转换为读操作+写操作
4		读QPS过高	本地缓存+持久化
5	MySQL	写并发过高	删除不必要的状态； 多线程并发更新优化为单线程异步更新
6	JVM	FullGC频繁	连接池内存泄漏，通过补救措施解决问题
7	日志	日志写入量过大	异步化

数据成果

系统吞吐量提升10倍



无效请求下降90%



成单时间缩短65%

故障恢复时间 小时级 → 分钟级

小结

围绕业务取最优解

01 业务可解决的不要留给技术

商务谈判

产品折中，交互异步化

02 技术针对性优化

避免“通用设计”陷阱

03 量化最优解，机器学习

限流模型优化

04 明确某些不可避免的问题：限流、共享库存

Q&A

CODE A BETTER LIFE

一 行 代 码　亿 万 生 活

更多技术干货
欢迎关注“美团技术团队”

招聘：java后台研发工程师岗位
邮箱：zhengxu@meituan.com

美团