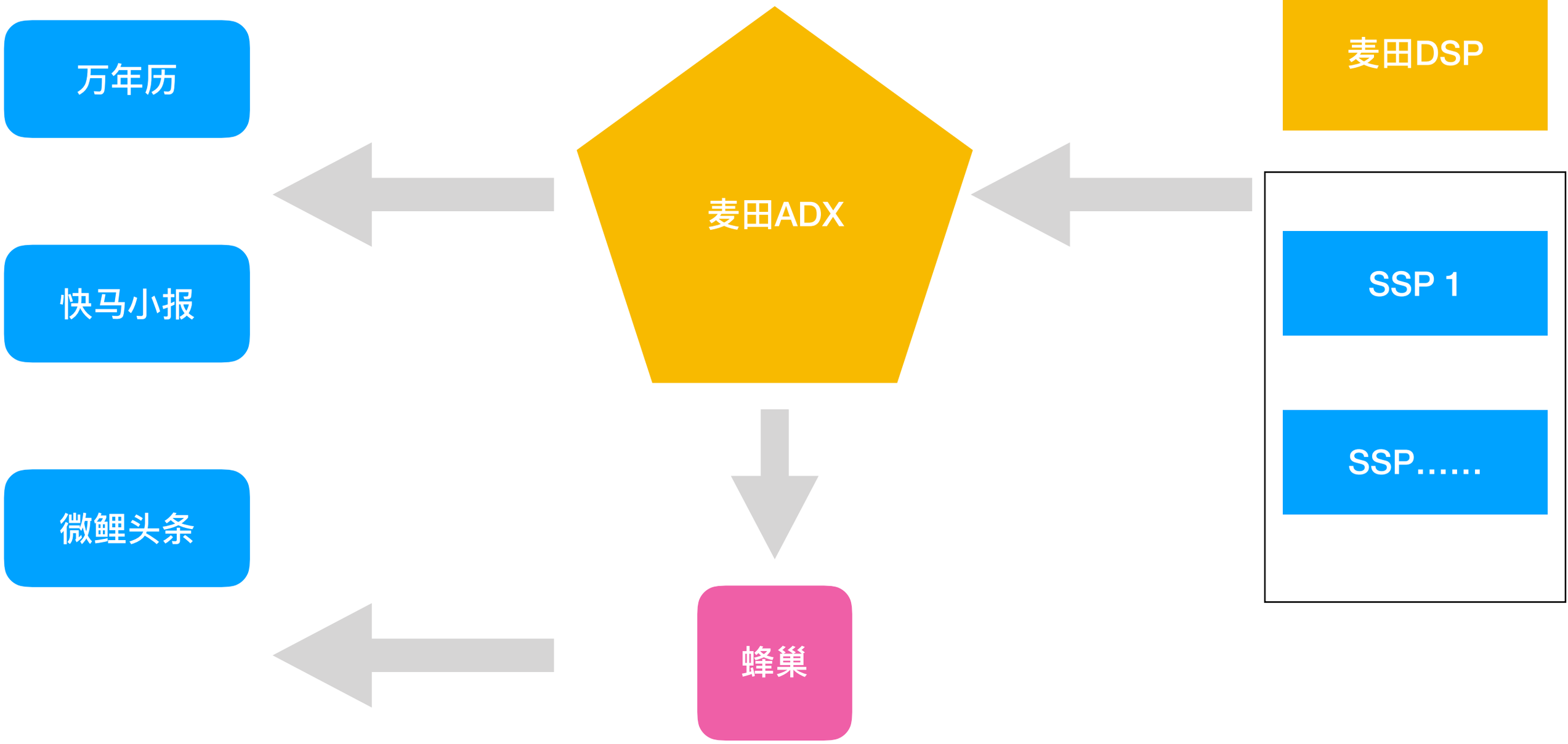
麦田广告系统架构

问题与解决方案

麦田广告

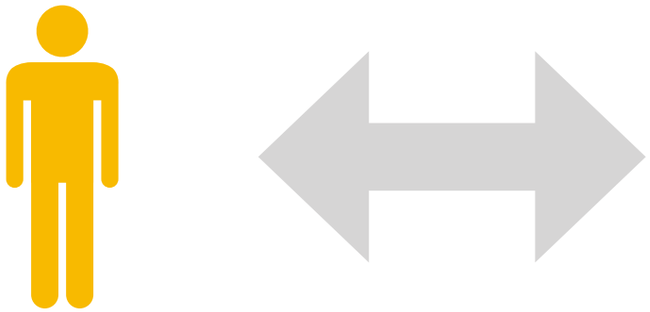
- 麦田ADX
    - 目前是多家SSP、麦田DSP的集合、分流地
- 麦田DSP
    - 目前用于内部定投广告

麦田ADX



麦田ADX—业务模块

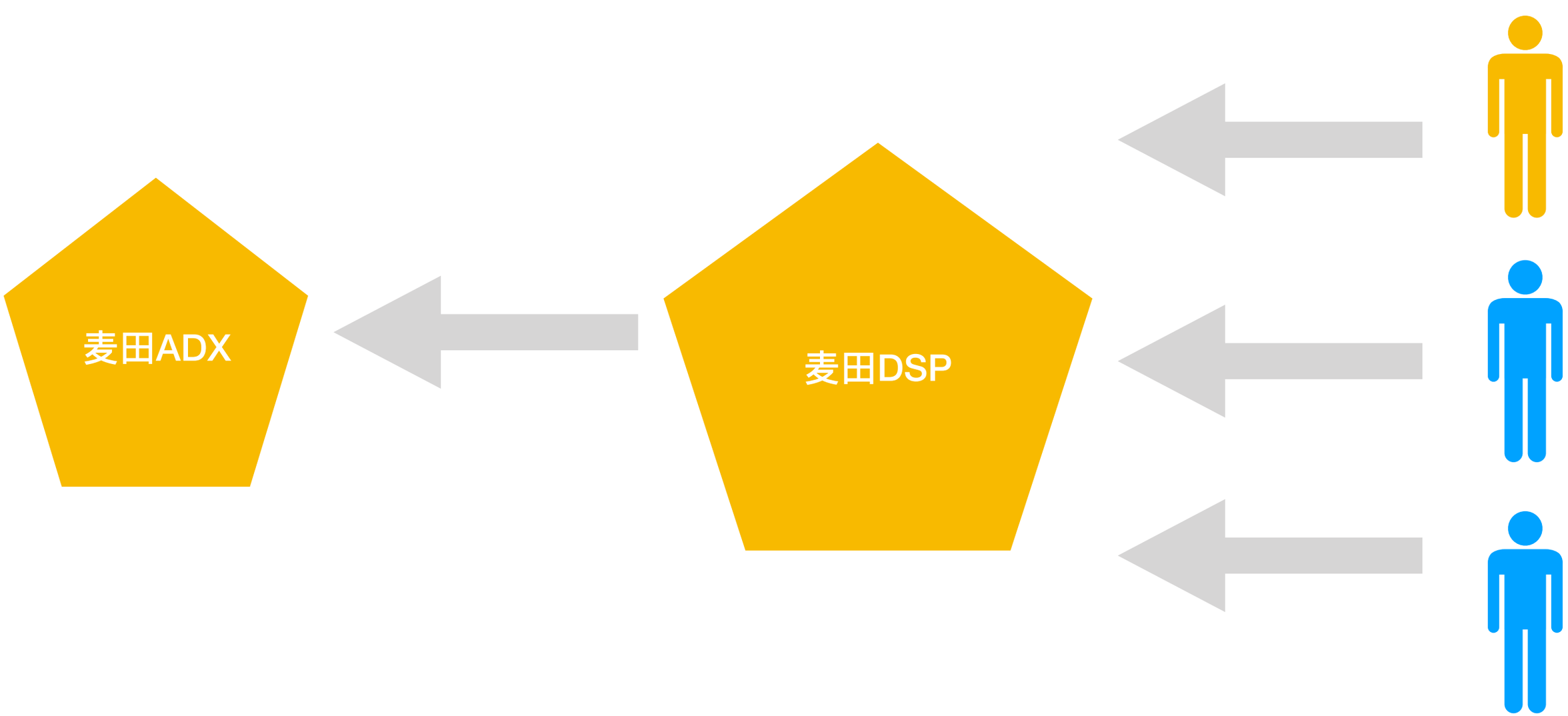
自有媒体资源管理



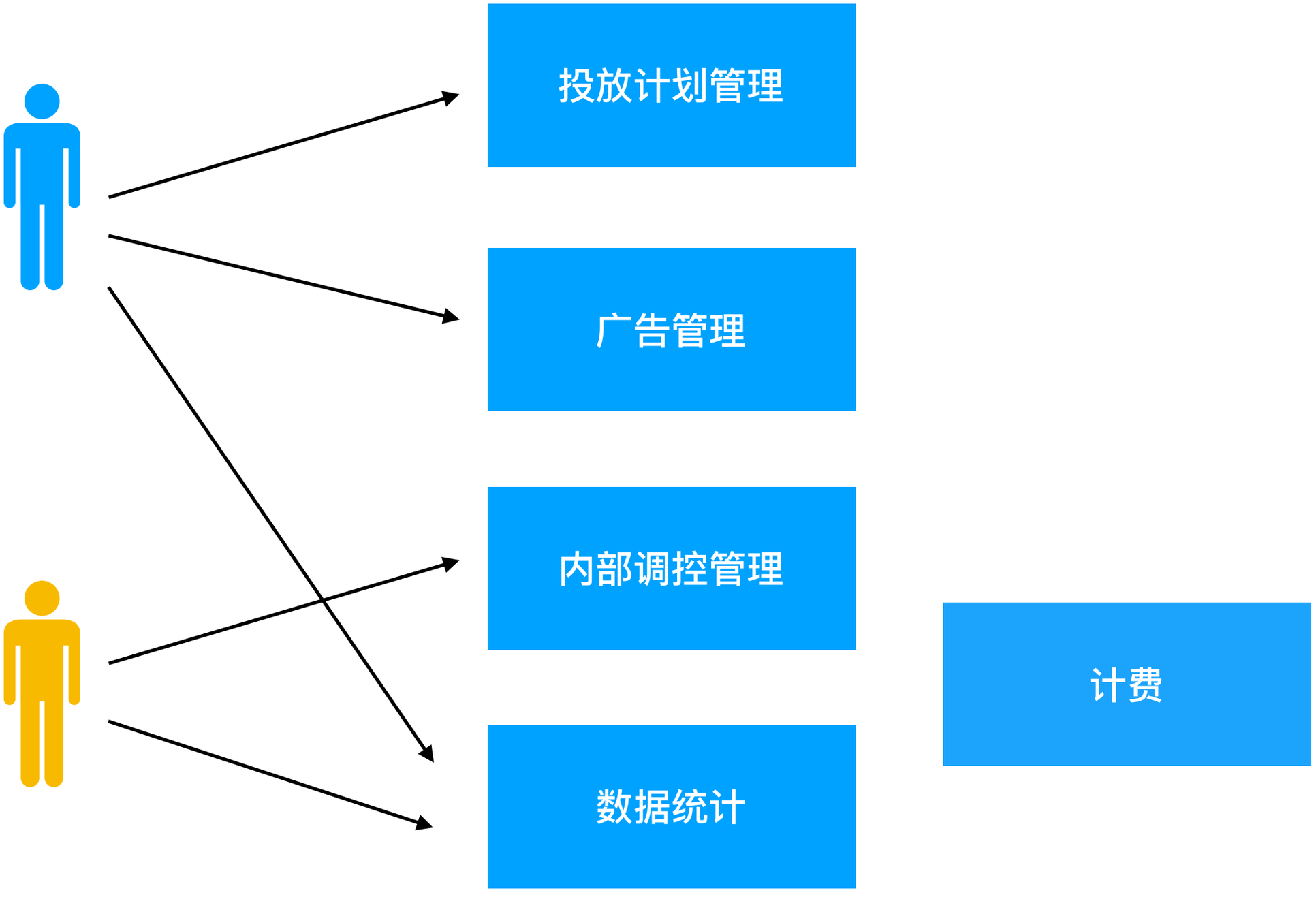
SSP分流管理

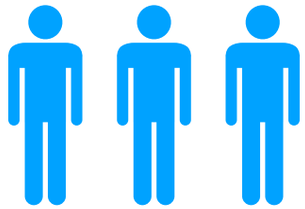
数据统计

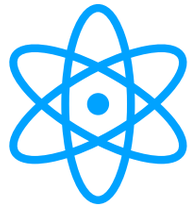
麦田DSP



麦田DSP—业务模块



接入层

REST API

广告主投放

SSP接入

服务器接入服务

广告服务

麦田ADX

麦田DSP

算法

缺失（只有简单的控制）

存储层

广告、计划

账户

实时近似报表

效果跟踪

在线

离线，近实时

近实时多维分析

离线报表

计费报表

监控

麦田广告 —技术栈

Nginx

Dubbo

接入层

Disruptor

RabbitMQ

Storm

中间层

MySQL

Elasticsearch

Redis

Kafka

HBase

Impala

HDFS

Kudu

离线数据层

Open-Falcon

ELK

基础服务

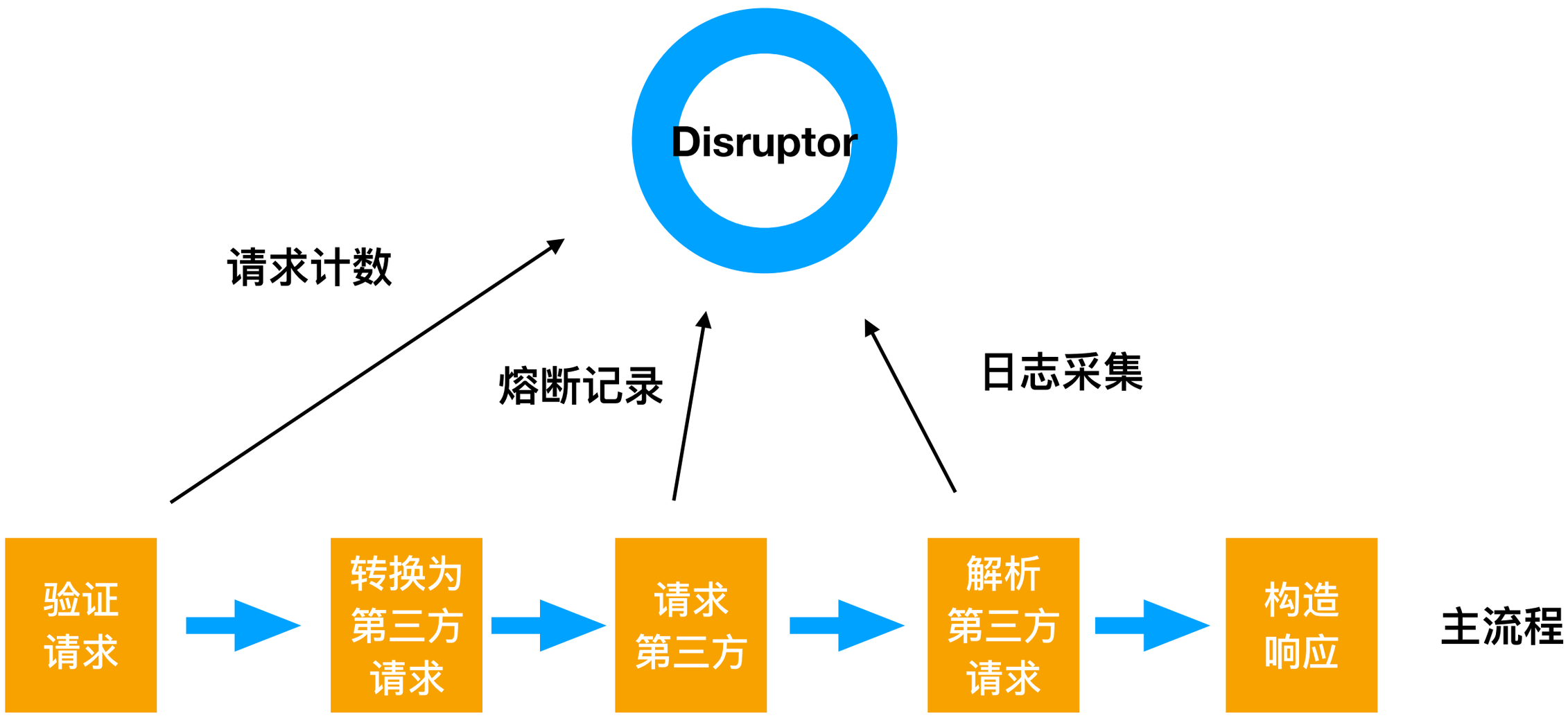
ADX 需要解决的问题

- 如何方便监控，排查问题
- 在满足业务的同时，如何收集必要数据
- 控量、均衡等流程，减少对Redis的请求
- 如何方便测试

DSP需要解决的问题

- 包含ADX的所有问题
- 复杂的状态机、级联的修改
- 如何在业务不断增加的时候，保持代码、架构的简洁、可靠

非主流程异步化


Disruptor
请求计数
熔断记录
日志采集
验证请求
转换为第三方请求
请求第三方
解析第三方请求
构造响应
主流程

日志的采集

数据仓库分析

采集
请求参数

采集
请求结果

采集
响应原始
报文

采集
构造结果

通过Disruptor
录入
Kafka

拼装数据

验证
请求

转换为
第三方
请求

请求
第三方

解析
第三方
请求

构造
响应

主流程

采样实时分析

Logback

打印
请求参数

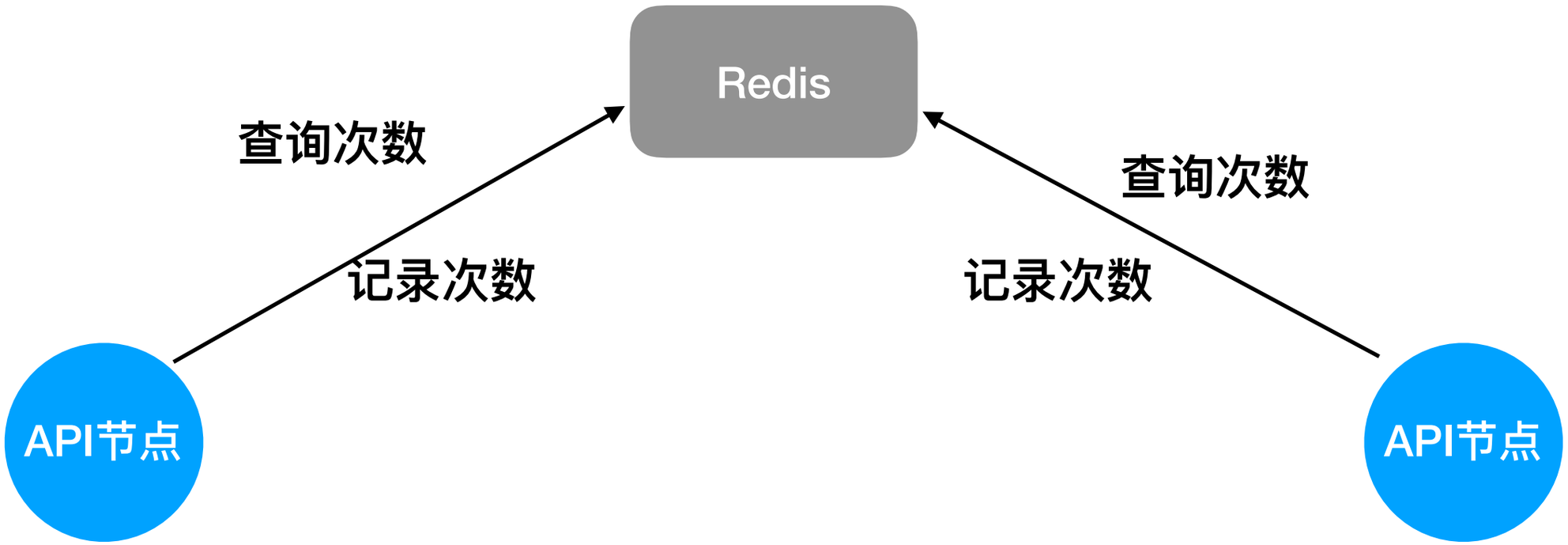
打印
HTTP请
求结果

打印
响应报文

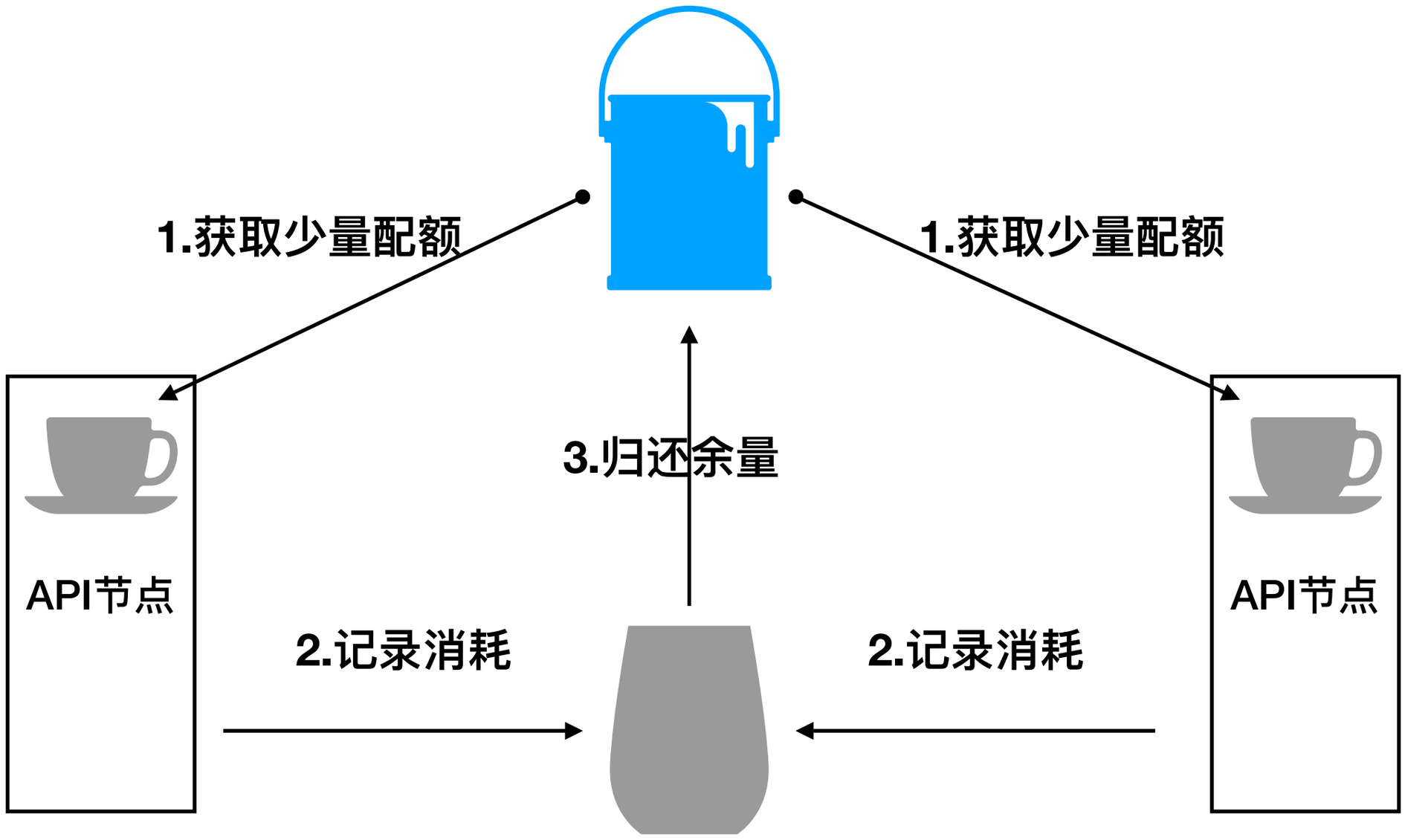
打印
链路最终
结果

通过
日志收集
录入
Elasticsearch

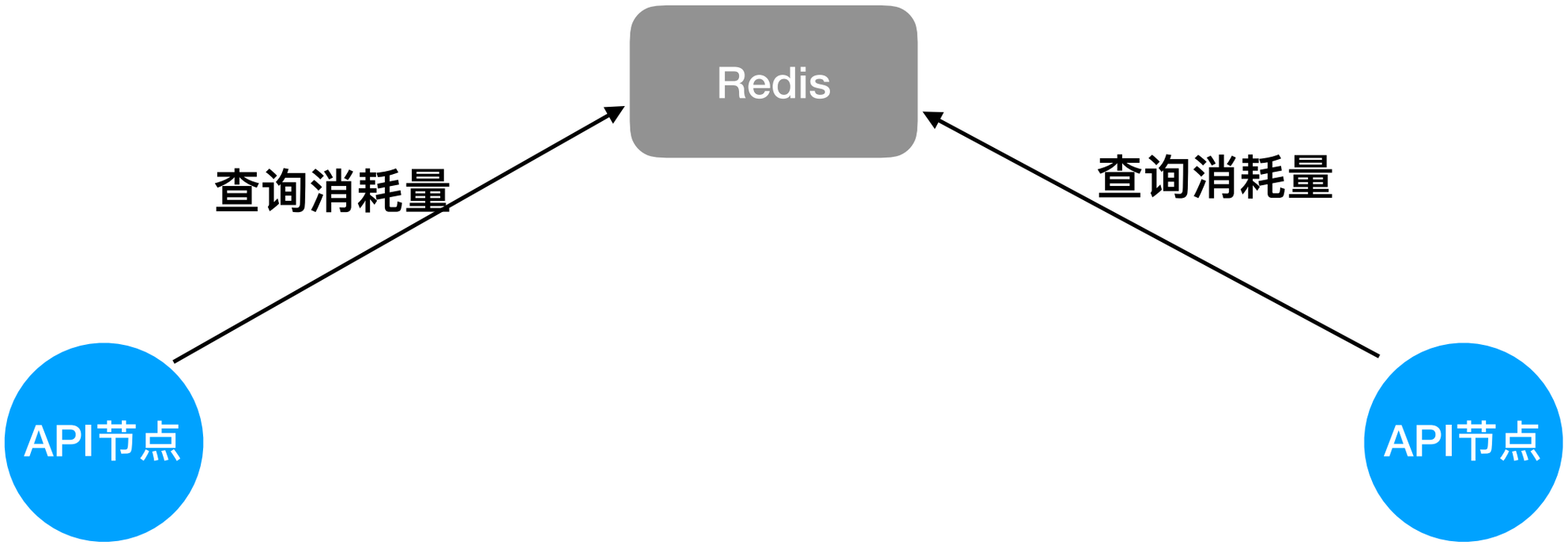
Request控量——方案一



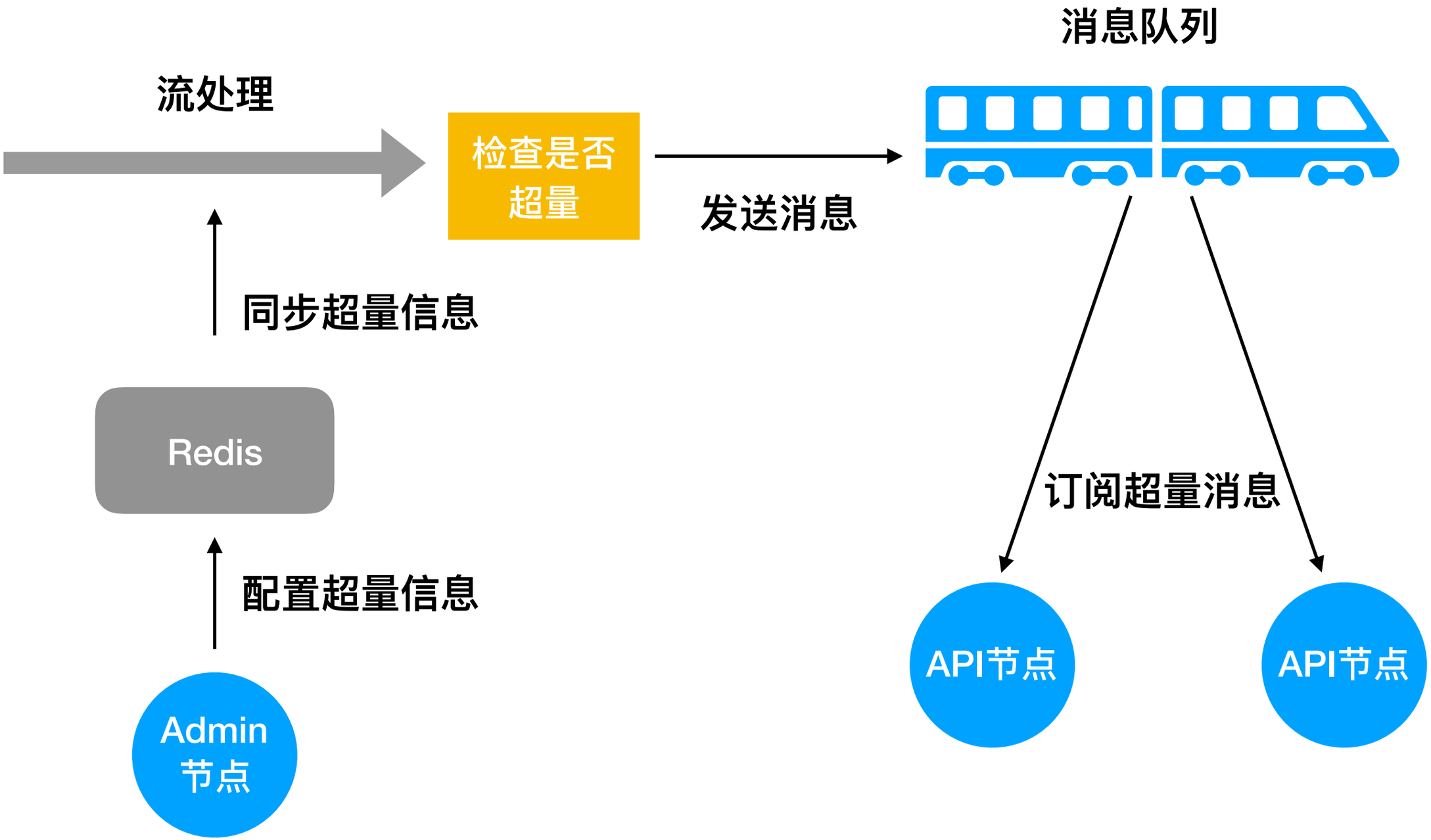
Request控量——方案二
(备选升级方案)



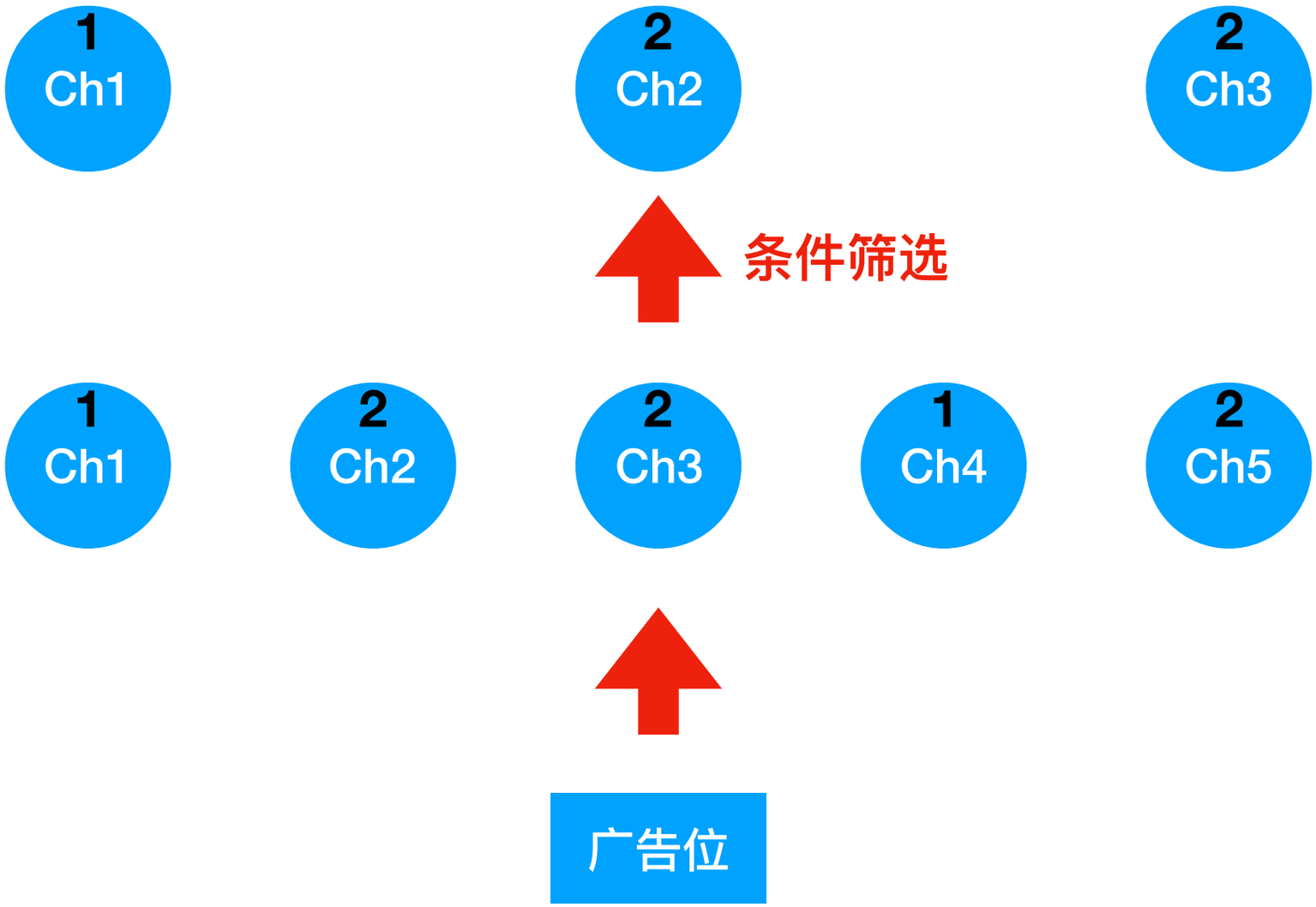
PV,Click控量——方案一



PV,Click控量——方案二
（备选升级方案）

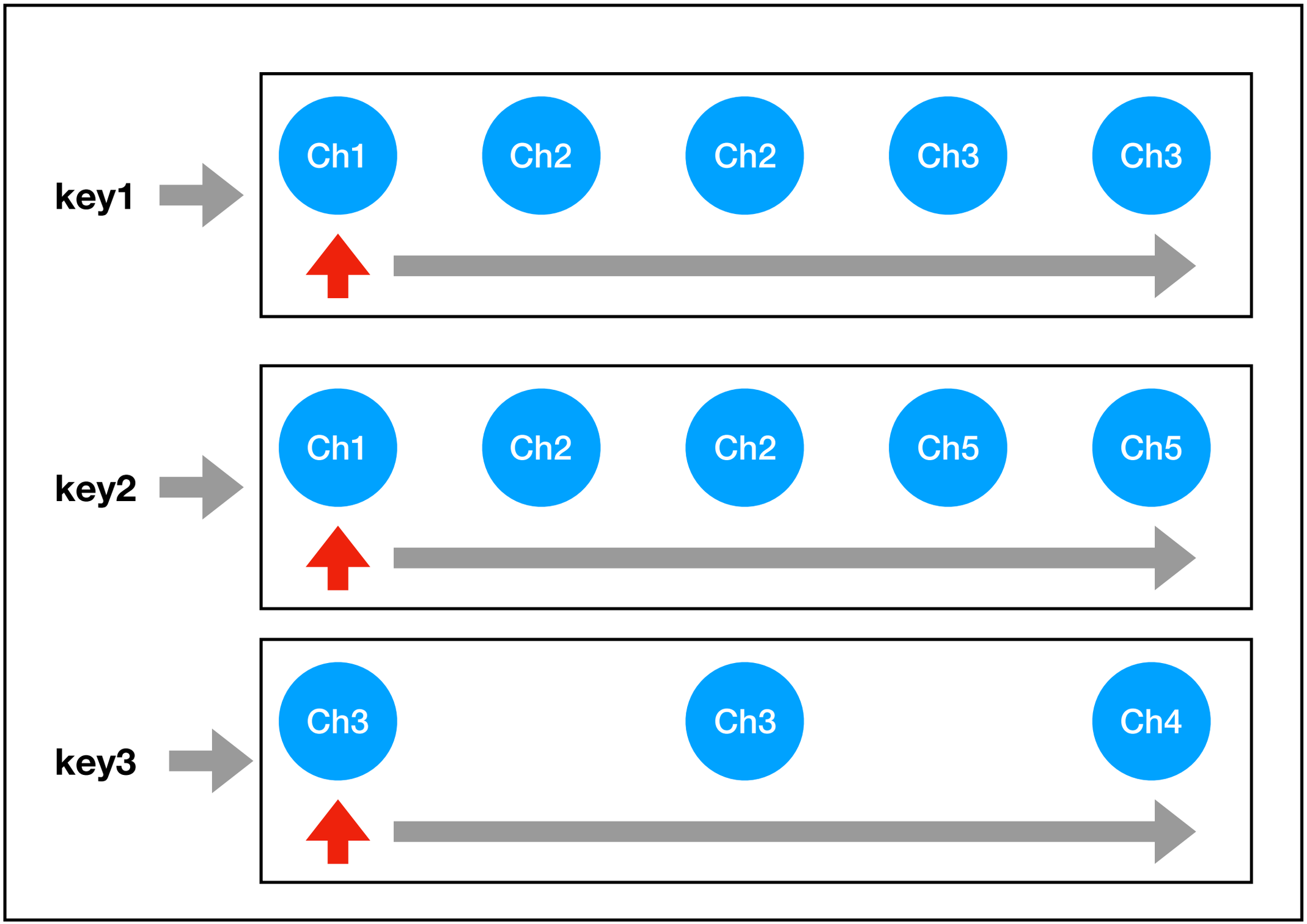


渠道的流量配置

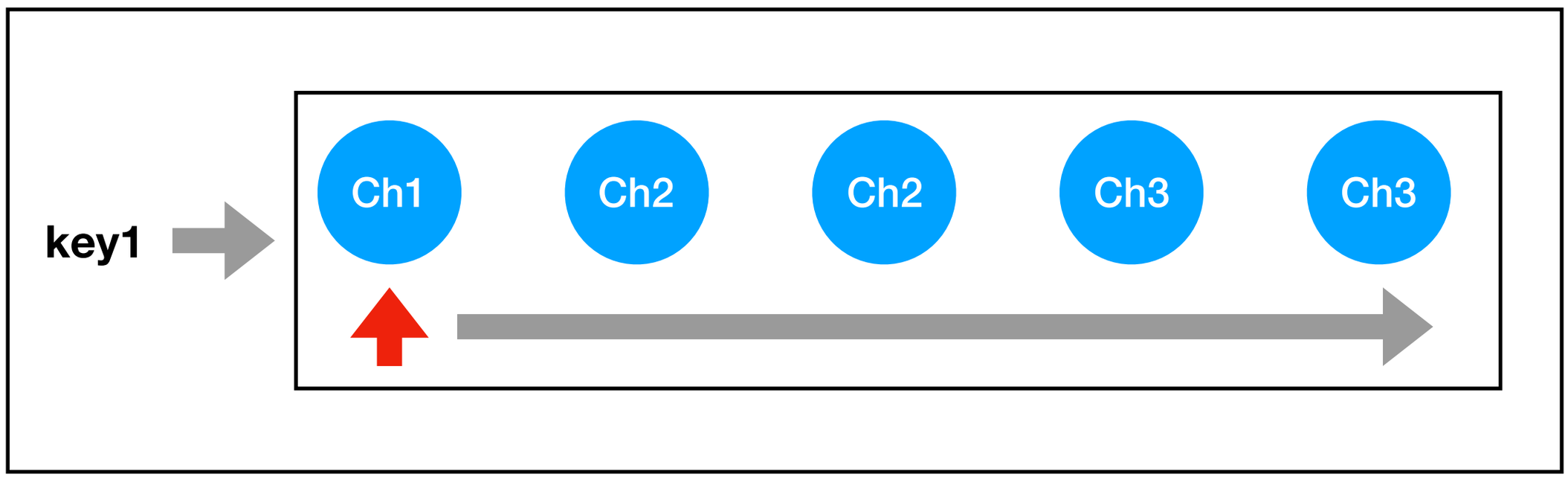


渠道的流量配置

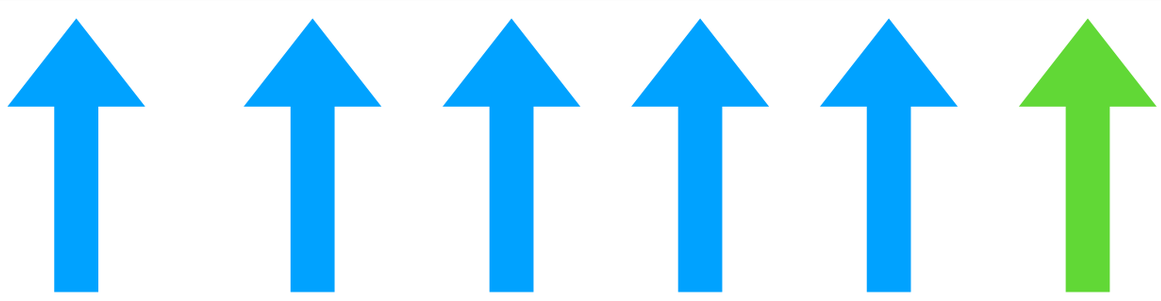
使用的数据结构



渠道的流量配置



多线程请求



1、随机返回

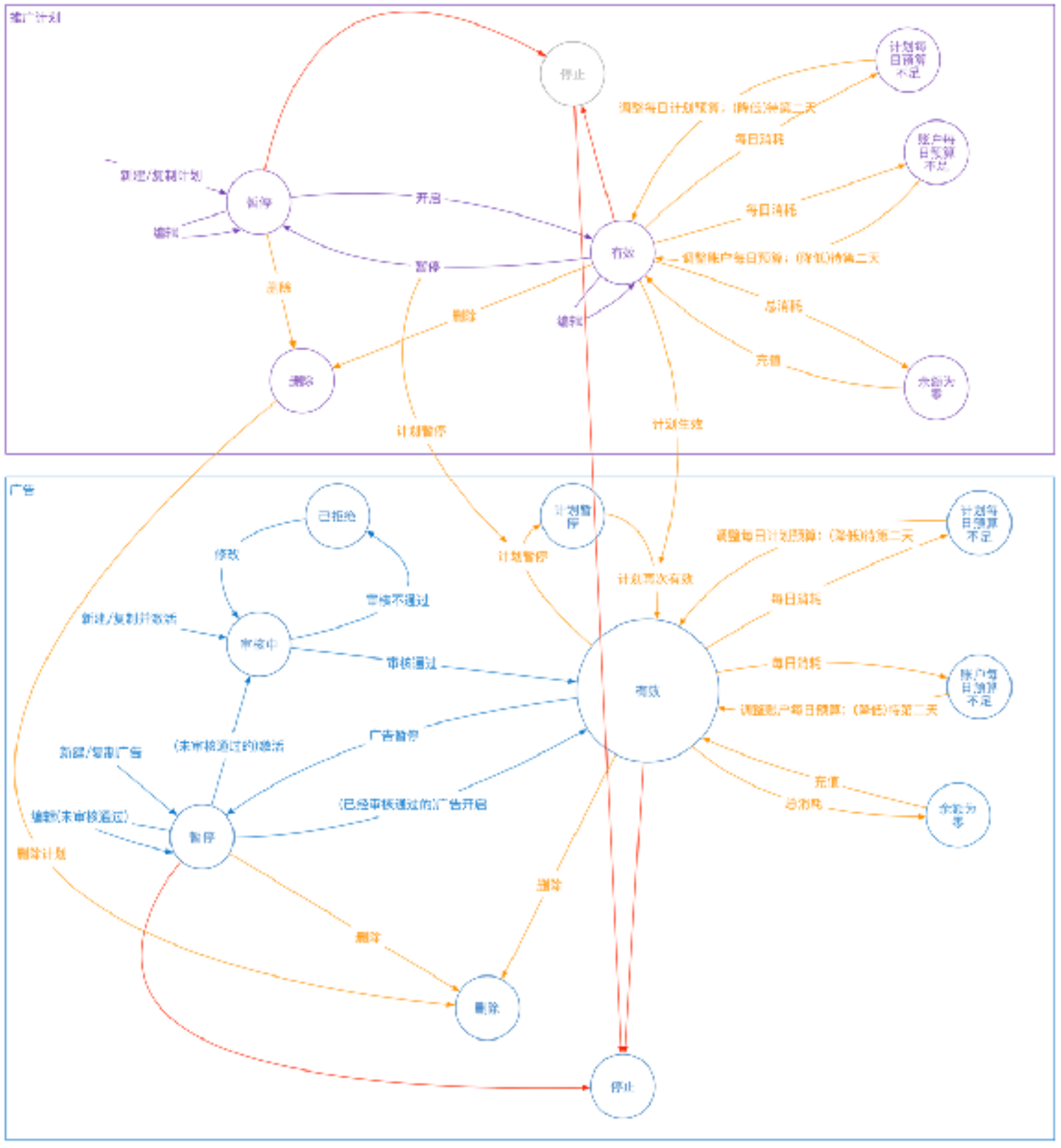
直到能通过key取到该结构

2、发送消息，异步创建上述数据结构

方便测试

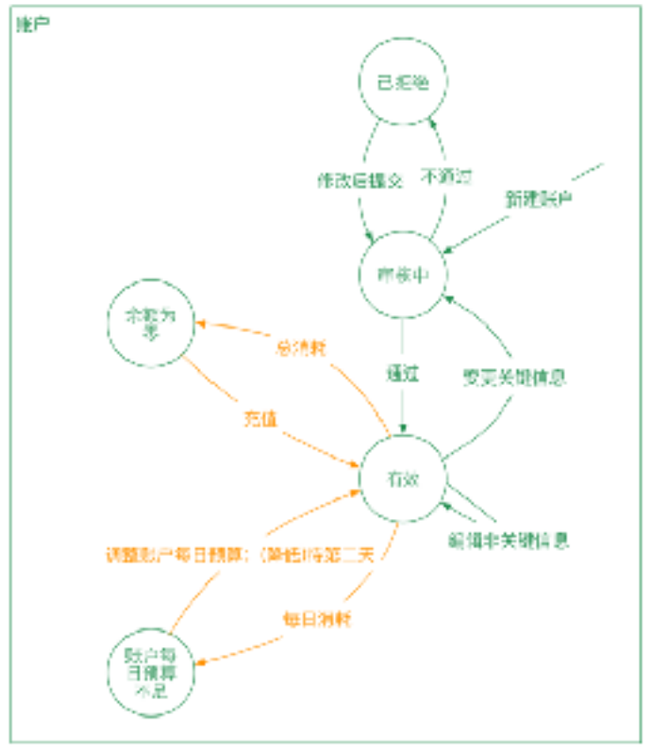
- 分析测试的最小依赖
- 分析测试的数据前提
- 隔离对外部流程、服务的依赖
- 将依赖用假数据、假服务代替
- 编写测试用例
- 构建独立、封闭的测试环境，自动化执行

复杂的状态机



橙色表示该操作与其它实体操作有关联

余额为0 > 账户每日预算不足 > 计划每日预算不足



问题

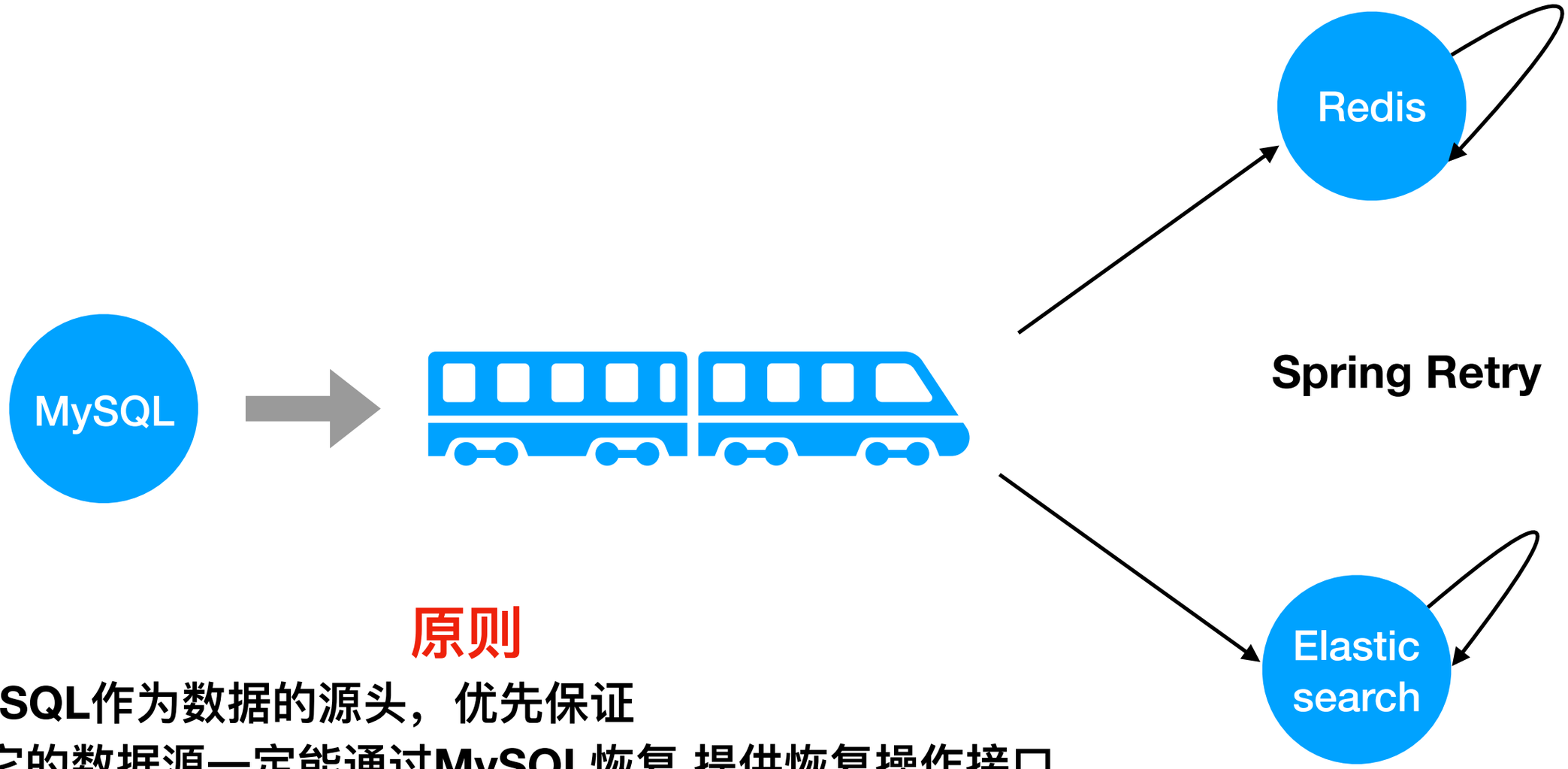
- 状态的级联修改

- 数据修改需分散到Mysql，Redis，Elasticsearch

解决思路

- 异步
- 幂等
- 高内聚

复杂的状态机



原则

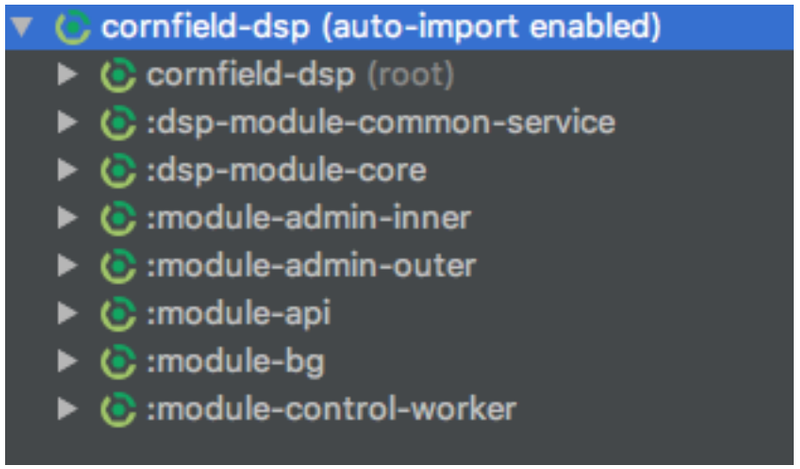
1、MySQL作为数据的源头，优先保证

2、其它的数据源一定能通过MySQL恢复,提供恢复操作接口

3、其它数据源的更新在保证幂等设计的基础上，
进行重试，如果重试失败记录详情并告警

高内聚

分模块



关注点单一

原则

广告是否可以被下发的第一层判断，
由搜索引擎中数据的有无决定

高内聚

线性的代码结构

```
public SuishenActionResult<Boolean> planStatusOperation(Long id, final int status,long uid,String username) throws IOException {
    AccountAdPlanStatusEnum statusAction = AccountAdPlanStatusEnum.valueOf(status);
    if (statusAction == null) {
        return SuishenActionResult.newBuilder().setActionStatus(ActionStatus.PARAMAS_ERROR).setMsg("状态操作不合法").build();
    }

    //判断是否是允许传的状态值
    if (statusAction != AccountAdPlanStatusEnum.SUSPENDED
            && statusAction != AccountAdPlanStatusEnum.AVAILABLE
            && statusAction != AccountAdPlanStatusEnum.DELETED) {
        return SuishenActionResult.newBuilder().setActionStatus(ActionStatus.PARAMAS_ERROR).setMsg("状态操作不合法").build();
    }

    //查看是否是广告的拥有者
    DspAdPlanDO theAdPlan = dspAdPlanDao.getByIdAndUserId(uid, id);
    if (theAdPlan == null) {
        return SuishenActionResult.newBuilder().setActionStatus(ActionStatus.PARAMAS_ERROR).setMsg("计划不存在").build();
    }

    // 对不同状态执行不同操作
    SuishenActionResult<Boolean> processResult = SuishenActionResult.newBuilder().setActionStatus(ActionStatus.NORMAL_RETURNED).build();
    if (statusAction == AccountAdPlanStatusEnum.AVAILABLE) {
        processResult = advertiserActivePlan(theAdPlan,uid,username);
    }

    if (statusAction == AccountAdPlanStatusEnum.SUSPENDED) {
        processResult = advertiserSuspendPlan(theAdPlan,uid,username);
    }

    if (statusAction == AccountAdPlanStatusEnum.DELETED) {
        processResult = advertiserDeletePlan(theAdPlan,uid,username);
    }

    return processResult;
}
```

高内聚

线性的代码结构

```
@TransactionMethod
public SuishenActionResult<Boolean> advertiserActivePlan(DspAdPlanDO thePlan,long uid,String username) throws IOException {
    AccountAdPlanStatusEnum currentPlanStatus = AccountAdPlanStatusEnum.valueOf(thePlan.getStatus());
    if (currentPlanStatus != AccountAdPlanStatusEnum.SUSPENDED) {
        SuishenActionResult.newBuilder().setActionStatus(ActionStatus.PARAMAS_ERROR).setMsg("当前计划状态不支持开启操作").build();
    }
    //查看账户的状态
    DspAdvertiserDO theAccount = dspAdvertiserService.getDspAdvertiserByUserId(thePlan.getUserId());
    AccountAdPlanStatusEnum accountStatus = AccountAdPlanStatusEnum.valueOf(theAccount.getStatus());
    if (accountStatus == AccountAdPlanStatusEnum.ACCOUNT_BALANCE_0 || accountStatus == AccountAdPlanStatusEnum.ACCOUNT_DAILY_BUDGET_LACK) {
        thePlan.setStatus(accountStatus.getStatus());
    } else {
        thePlan.setStatus(AccountAdPlanStatusEnum.AVAILABLE.getStatus());
    }
    //更新计划数据库
    updateForTransaction(thePlan,uid,username);

    //增加到本地缓存中
    addLocalCache(thePlan);
    //更新相关联的广告
    List<DspAdvertisementDO> adList = dspAdvertisementService.getListByPlanId(thePlan.getId());
    List<DspAdvertisementDO> updateList = new ArrayList<>();
    for (DspAdvertisementDO theAd : adList) {
        if (theAd.getStatus() == AccountAdPlanStatusEnum.SUSPENDED.getStatus()) {
            continue;
        }
        if (accountStatus == AccountAdPlanStatusEnum.ACCOUNT_BALANCE_0 || accountStatus == AccountAdPlanStatusEnum.ACCOUNT_DAILY_BUDGET_LACK) {
            theAd.setStatus(accountStatus.getStatus());
        } else {
            theAd.setStatus(AccountAdPlanStatusEnum.AVAILABLE.getStatus());
        }
        dspAdvertisementService.updateForTransaction(theAd, JsonUtils.json2Object(thePlan.getScheduleInfo(), ScheduleInfo.class),uid,username);
        updateList.add(theAd);
    }
    //更新计划ES
    adPlanExecutor.onPublish(DisruptorOperationType.UPDATE, thePlan);

    //更新广告ES
    advertisementExecutor.onPublishList(DisruptorOperationType.UPDATE, updateList, scheduleInfo: null);

    return SuishenActionResult.newBuilder().setActionStatus(ActionStatus.NORMAL_RETURNED).build();
}
```

用消息队列解耦

Thank You