ClickHouse在vivo ABTest的应用

vivo互联网 · 刘兆坤

刘兆坤

vivo互联网

- 2016年从事大数据工作,对大数据计算引擎源码有深入研究，参与开源贡献
- 2019年加入vivo，从事内容分发、线上/线下营销业务的数仓建模、数据应用
- 2020年开始负责ABTest的数据能力建设

目录

CONTENTS

一、ABTest简介

什么是ABTest、常见的统计陷阱、vivo ABTest的规模

什么是ABTest？

APP通知弹窗案例-哪个方案的同意率更高？

对照组

实验组A

实验组B

目标：通过实验，**验证高同意率组件**，持续提升弹窗同意率，降低弹窗对app访问的影响，提升app活跃

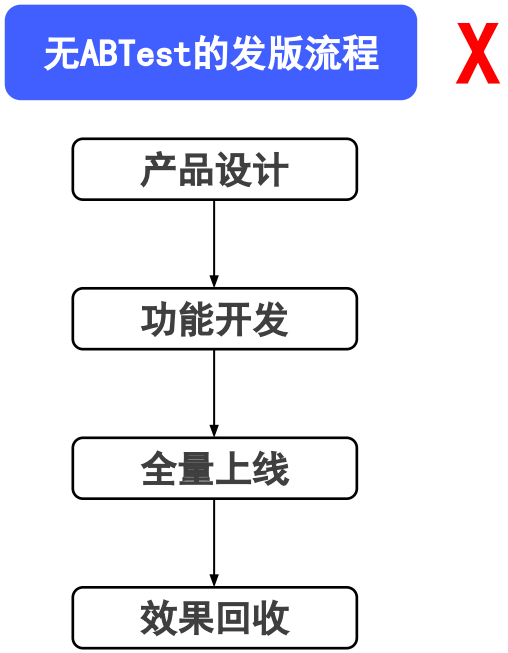
用数据做客观评价

日均同意率：

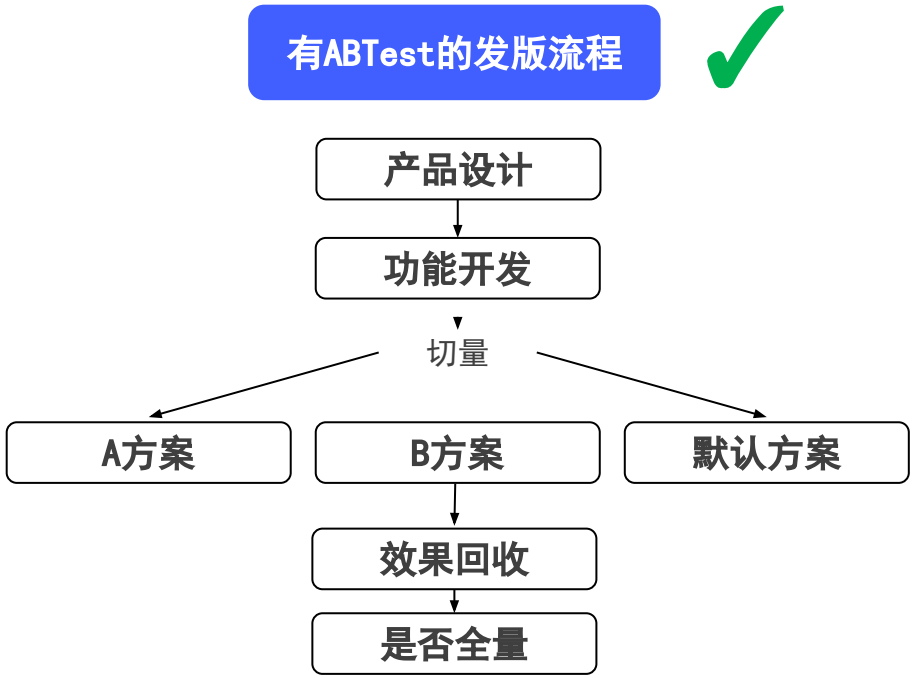
- 实验组A相对于对照组约提升3%
- 实验组B相对于对照组下降约10%

什么是ABTest？

通过统计学方法，对比只有一个变量不同的同一产品的两个或多个不同版本的表现来研究该变量的作用以及影响。



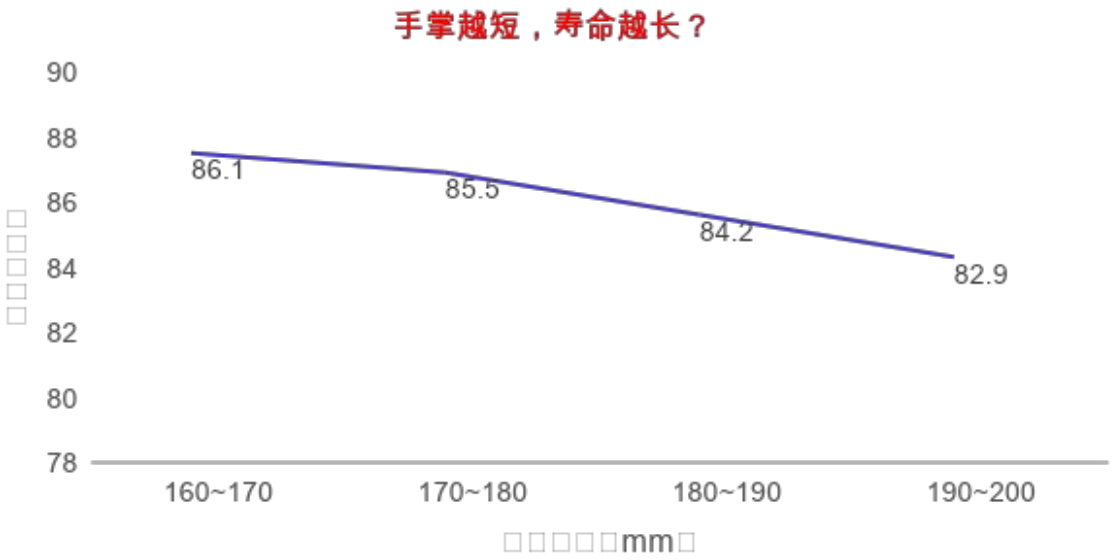
若新版本的效果负向，影响面大

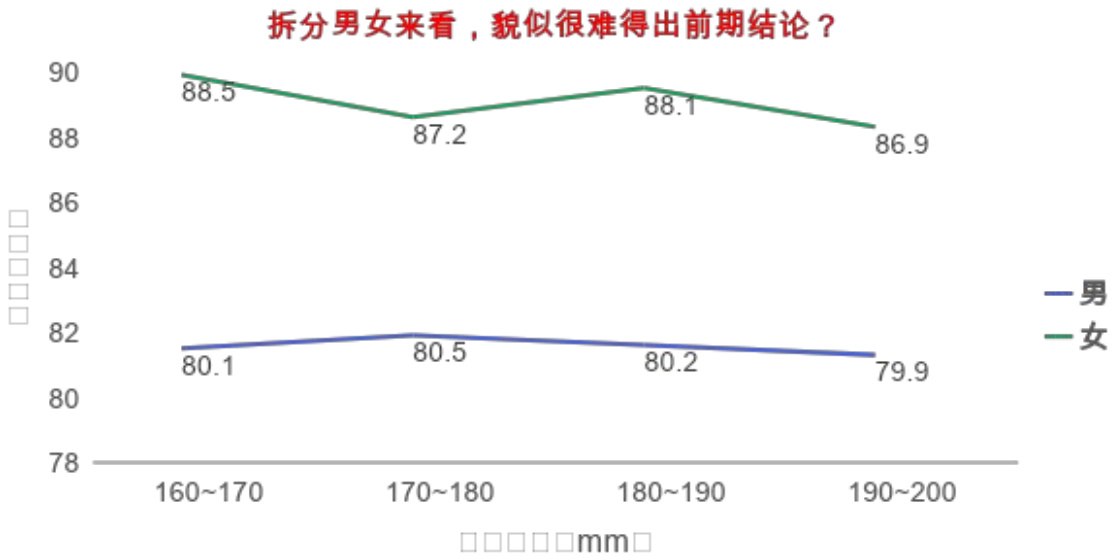


低成本试错

数据不会骗人？所见即所得？

统计学中的经典案例-手掌长度与寿命长短存在关联吗？

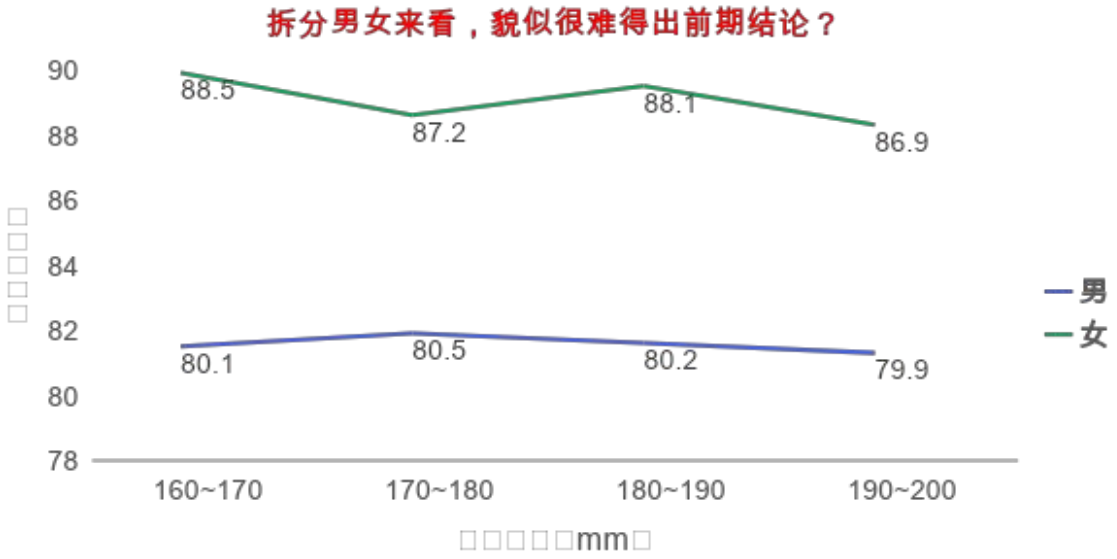


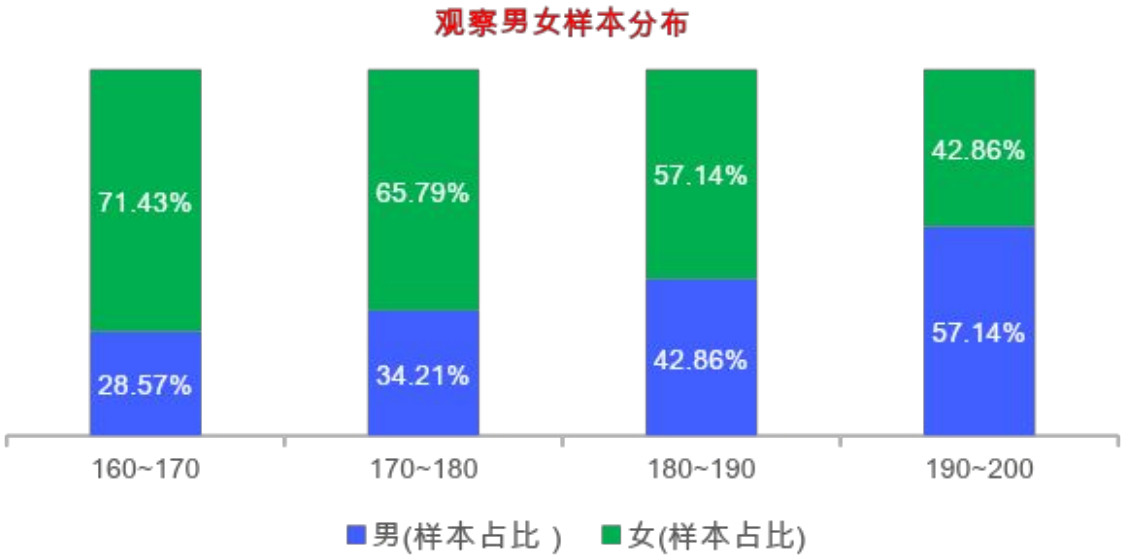


数据不会骗人？所见即所得？

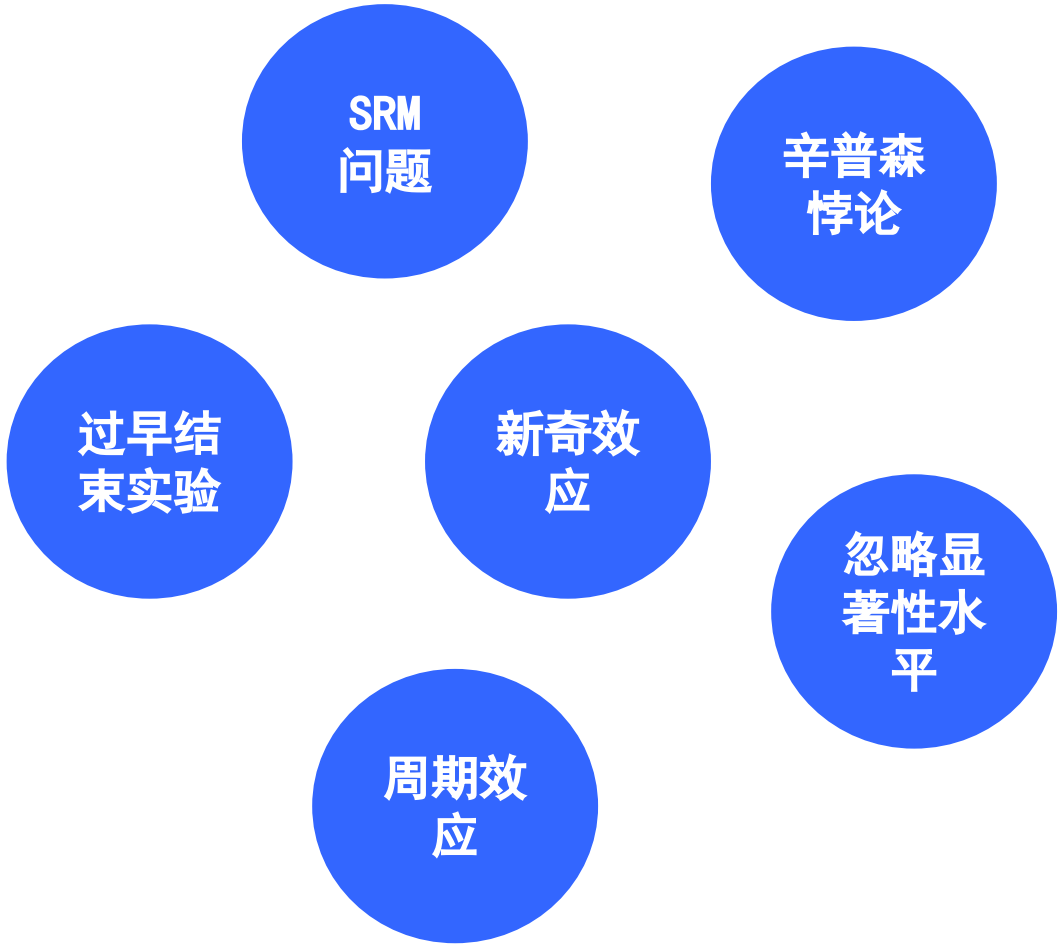
统计学中的经典案例-手掌长度与寿命长短存在关联吗？

初始结论得出的根本原因是样本比例差异，且女性比男性更长寿 ，女性比男性手更小。因此手掌长度与寿命并无直接关系，结论错误





常见的统计陷阱



统计陷阱是数据分析过程中容易发生的偏差和误解，需要谨慎对待并采取多方面的对策来消除其影响。

ABTest的目的

规避实验效果统计陷阱，用合理的置信结果衡量实验效果，从而降低试错成本，提升决策效率！

数据统计相关能力

- 最小样本量
- AA分析
- 多重校验
- 方差缩减
- 统计功效&MDE
- 卡方检验
- 涨跌幅
- P&显著性
- 数据模型矩阵
- Hive
- Spark
- ClickHouse
- *Flink*
- HDFS
- *Hudi*
- *Kafka*

平台概况

互联网&系统侧

100+业务模块

亿级dau产品

业务覆盖

数据计算

千亿级数据吞吐

SLA>99.99%

亚秒级查询响应

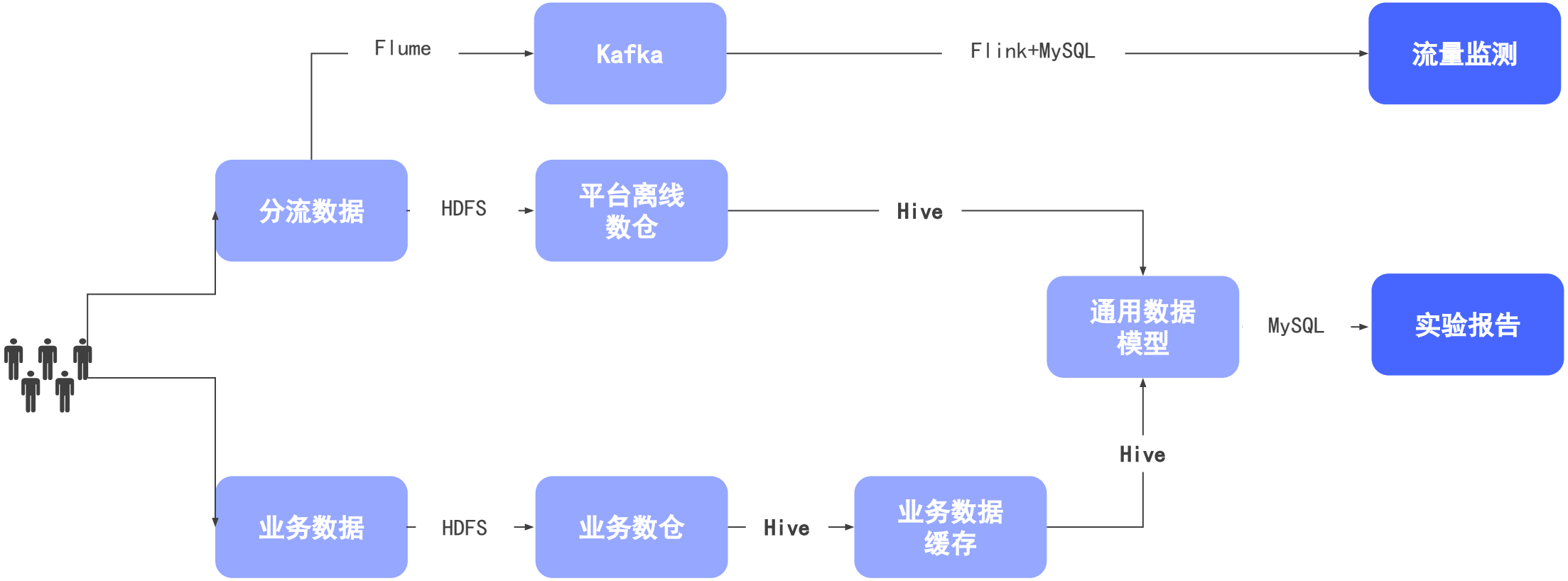
二、ABTest计算架构演进

计算链路概况、技术选型

计算架构演进

v1.0

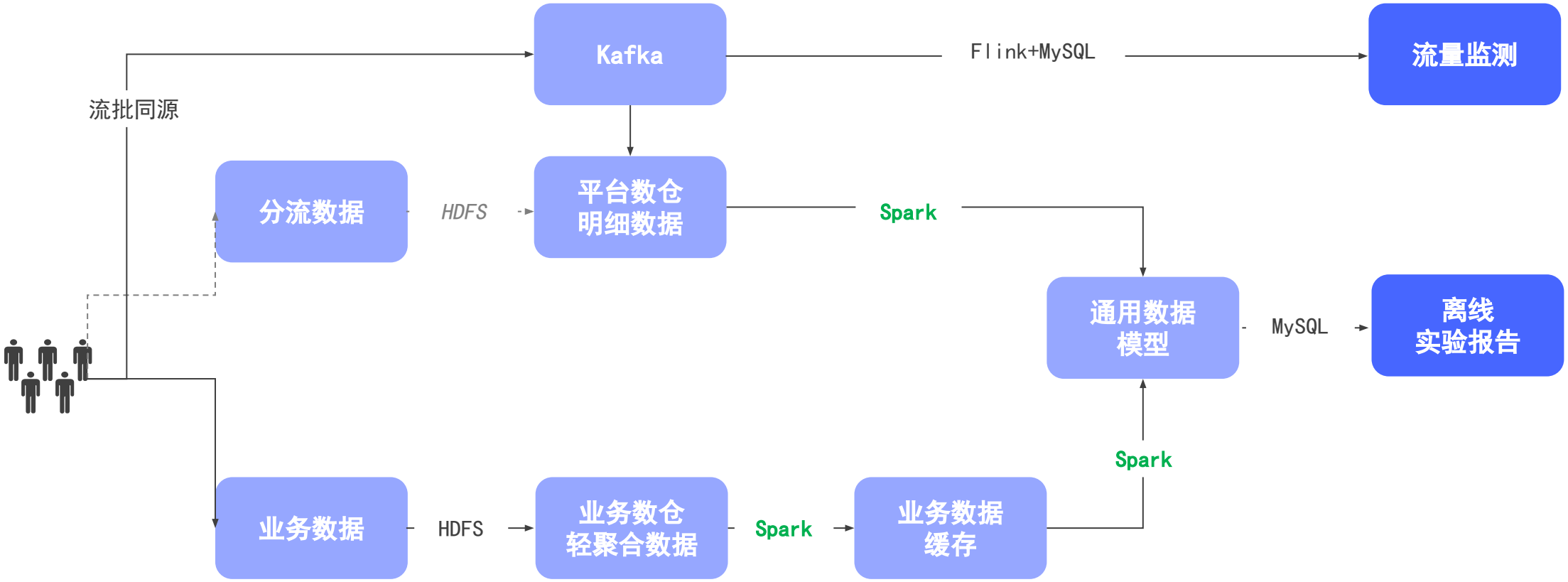
面临的问题：Hive磁盘计算，任务积压严重



计算架构演进

v2.0-任务执行时长整体缩短30%、计算资源节省20%，任务积压延迟情况有效缓解，队列弹性增加

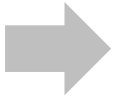
面临的问题：Spark难支持多维即席分析，无法满足业务实验效果下钻归因诉求

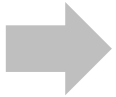


计算架构演进

什么是实验效果下钻归因？

产出实验报告





增加维度看看呢？

dau提升18%

实验组	118
对照组	100

下沉一层

实验组	资源位1	41
对照组	资源位1	38
实验组	资源位2	32
对照组	资源位2	40
实验组	资源位3	45
对照组	资源位3	22

业务上可以采取针对性措施

计算架构演进

OLAP组件选型

诉求：支持百亿数据秒级查询、sql友好、支持灵活的维度选择、数据集构建简单

- 查询响应快
- 预聚合
- 实时数据摄入
- 不适合大数据量计算
- 支持sql
- 社区活跃平稳

- 大数据量下，查询响应一般
- 支持明细查询
- 支持联邦查询、join查询
- 运维成本低
- 支持sql
- 社区活跃平稳

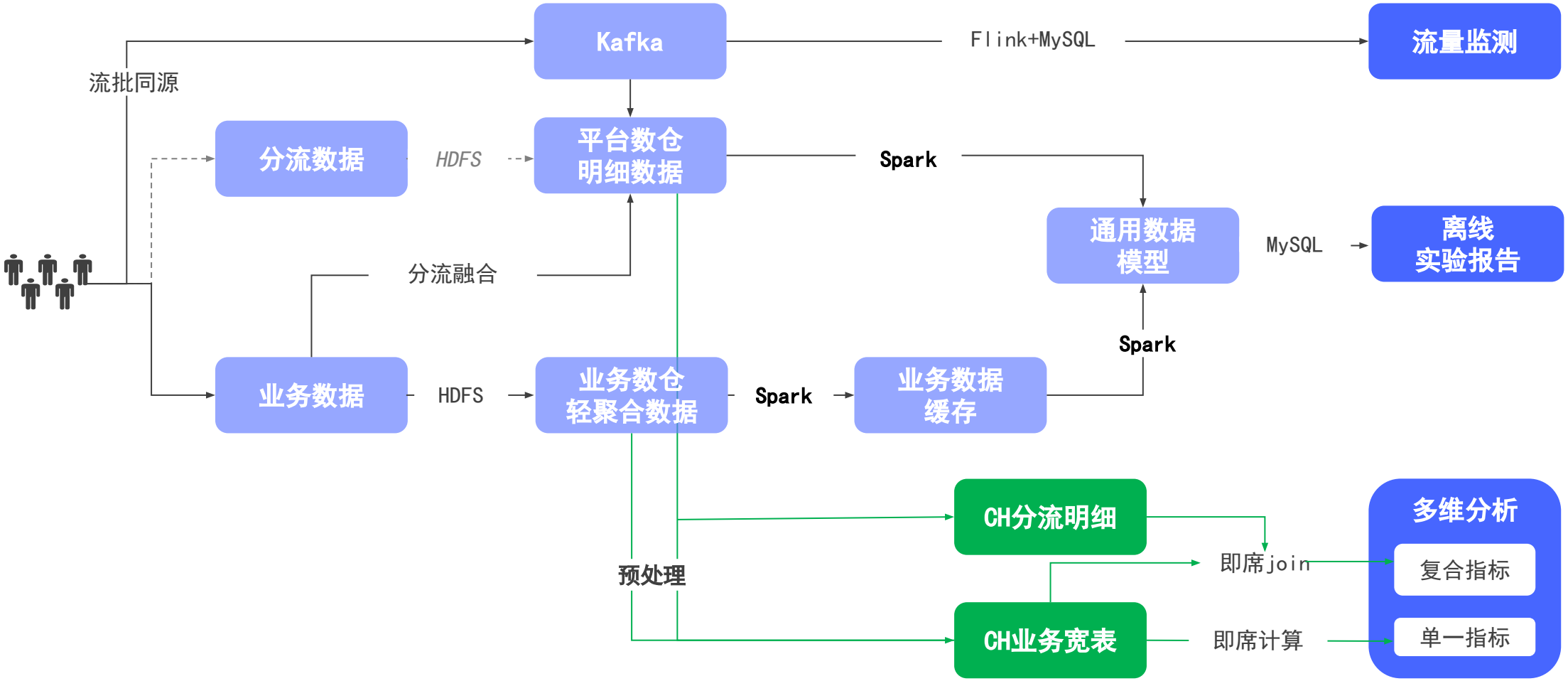
- 查询响应快
- 预聚合
- Sql支持完善
- 运维成本高
- 扩展差
- 社区活跃相对低

- 单表性能强悍（向量化等）
- 支持明细查询
- 支持sql
- 灵活性强
- 支持主键索引、二级索引
- 社区活跃

计算架构演进

v3.0-提供多维归因分析能力

面临的问题：1、即席查询的性能问题 2、随着业务接入量级增大，仍然会出现大批量的任务积压问题



三、ClickHouse应用实践

性能优化实践、ClickHouse定时计算提效

性能优化实践-场景1

分流数据1亿，业务数据100亿，原查询耗时100s+，目标<10s

```
(
    SELECT
      day,
      实验id,
      方案id,
      主键id
    FROM
      ${分流数据}
    WHERE
      day between '日期范围'
      AND testappid = 'xx'
      AND 实验id = 'XXXX'
      AND 方案id IN ('71', '72', '73')
  ) AS t1
```

```
(
    SELECT
      day,
      主键id,
      维 度 1,
      维 度 2,
      sum(指标) AS agg_value
    FROM
      ${业务数仓}
    WHERE
       day between '日期范围'
    GROUP BY
      day,
      主键id,
      维 度 1,
      维 度 2,...
  ) AS t2
```

t1 join t2 on t1. 主键id=t2.主键id group by day,实验id, 方案id,维度….

性能优化实践-场景1

分流数据1亿，业务数据100亿，原查询耗时100s+, 目标<10s

思路一、进行join优化

join性能可否满足即席查询需要未知

思路二、宽表化处理

增加数据处理链路

性能优化实践-场景1

Join优化

优化前（100+s）

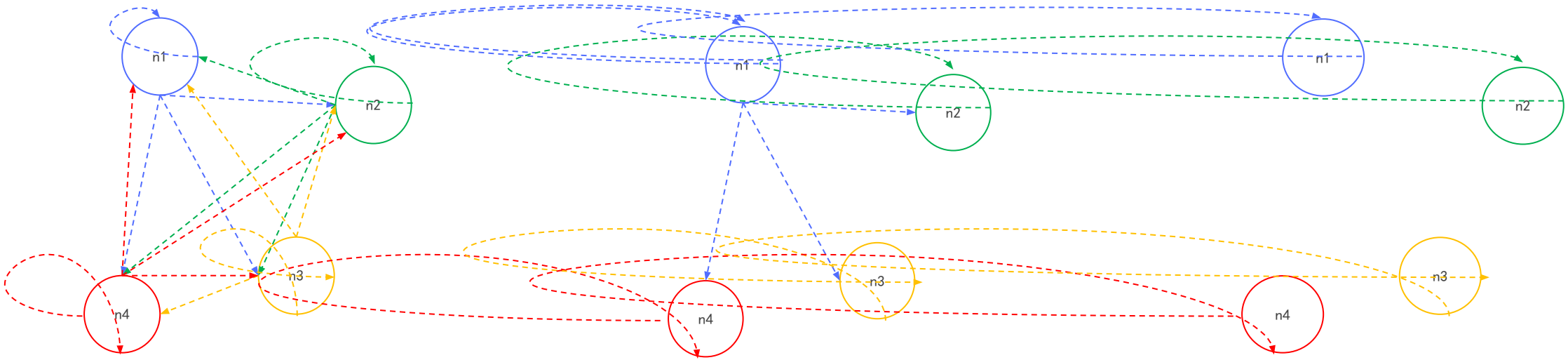
普通join

第一次优化（73s）

global join

第二次优化（40s）

colocate join（数据预分布）



读放大，查询约n*n次

如果右表数据量较大，同样会带来性能损失，查询约2n次

Join key一定相同，执行本地化查询，查询约n次

性能优化实践-场景1

宽表化处理

分流数据

主键id	实验id	方案id
id1	1	1001
id1	2	2001
id2	1	1002
id3	2	2002

主键id	test_plan_id
id1	1-1001, 2-2001
id2	1-1002
id3	2-2002

业务轻聚合数据

主键id	city	cnt
id1	sz	10
id1	nj	13
id2	nj	20
id3	hz	15

Spark2CH

主键id	test_plan_id	city	cnt
id1	1-1001, 2-2001	sz	10
id1	1-1001, 2-2001	nj	13
id2	1-1002	nj	20
id3	2-2002	hz	15

性能优化实践-场景1

宽表化处理

SQL简化，查询15s

```
    SELECT
      day,
      splitByString(‘-’,replace(extract(test_plan_ids,’,1-1001,|,
2-2001’),’,’,’’))[2] as test方案id,
      count(主键id) as pv
    FROM
      ${CH宽表}
    WHERE
      day between '日期范围'
      AND multiMatchAny(test_plan_ids,[‘,1-1001,’,’2-2001’])
    group by
      day,

splitByString(‘-’,replace(extract(test_plan_ids,’,1-1001,|,2-2001
’),’,’,’’))[2]
```

如何进一步优化？

性能优化实践-场景1

宽表化处理

通过explain发现主键索引失效，考虑使用二级索引（跳数索引）

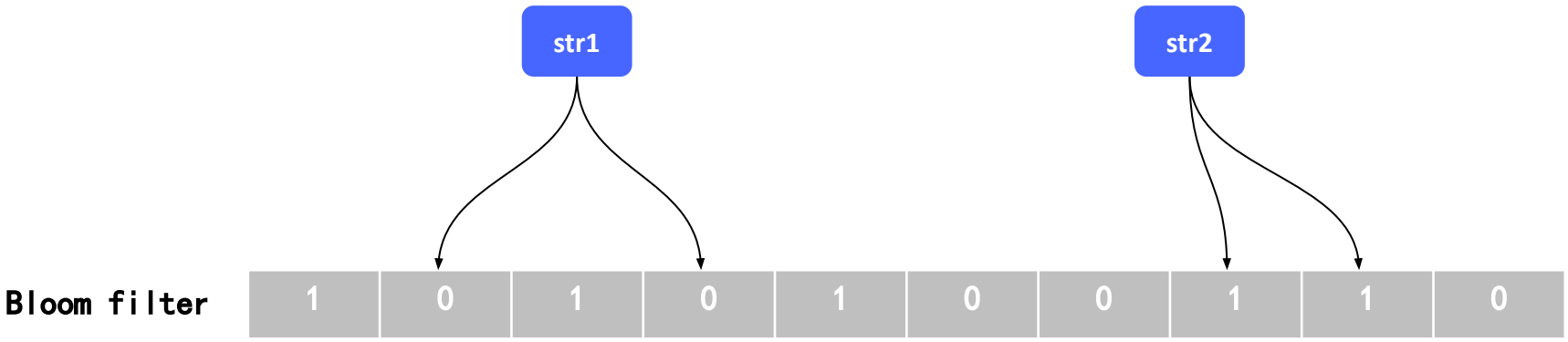
```
explain
Expression ((Projection + Before ORDER BY))
  MergingAggregated
    Union
      Aggregating
        Expression (Before GROUP BY)
          Filter (WHERE)
            ReadFromMergeTree
            Indexes:
              MinMax
                Keys:
                  day
                Condition: and(and((day in (-Inf, 19453]), (day in [19447, +Inf))), and((day in (-Inf, 19453]), (day in [19447, +Inf))))
                Parts: 4/27
                Granules: 87844/133899
              Partition
                Keys:
                  day
                Condition: and(and((day in (-Inf, 19453]), (day in [19447, +Inf))), and((day in (-Inf, 19453]), (day in [19447, +Inf))))
                Parts: 4/4
                Granules: 87844/87844
              PrimaryKey
                Keys:
                  imei
                Condition: (     not in ['', ''])
                Parts: 4/4
                Granules: 87255/87844
      ReadFromRemote (Read from remote replica)
```

性能优化实践-场景1

宽表化处理

跳数索引可排除不符合筛选条件的granule

INDEX idx_ids test_plan_ids TYPE ngrambf_v1(8, 256, 2, 0) GRANULARITY 20



如果hash结果不在集合中，则元素肯定不在，反之，可能在

性能优化实践-场景1

宽表化处理

使用跳数索引并且multiMatchAny->multiSearchAny ，优化后7s

```
explain
Expression ((Projection + Before ORDER BY))
  MergingAggregated
    Union
      Aggregating
        Expression (Before GROUP BY)
          Filter (WHERE)
            ReadFromMergeTree
            Indexes:
              MinMax
                Keys:
                  day
                Condition: and(and((day in (-Inf, 19453]), (day in [19447, +Inf))), and((day in (-Inf, 19453]), (day in [19447, +Inf))))
                Parts: 4/27
                Granules: 87844/133899
              Partition
                Keys:
                  day
                Condition: and(and((day in (-Inf, 19453]), (day in [19447, +Inf))), and((day in (-Inf, 19453]), (day in [19447, +Inf))))
                Parts: 4/4
                Granules: 87844/87844
              PrimaryKey
                Keys:
                  imei
                Condition: (     not in ['', ''])
                Parts: 4/4
                Granules: 87255/87844
              Skip
                Name: e
                Description: ngrambf_v1 GRANULARITY 20
                Parts: 4/4
                Granules: 940/87255
      ReadFromRemote (Read from remote replica)
```

性能优化实践-场景2

百亿数据，count distinct计算uv响应慢

优化前（120s）

```
select
  sum(show_count)AS index_value,
  countDistinct(主键id) AS uv
FROM
  ${CH表 }
WHERE
  day between '日期范围'
```

第一次优化（48s）

根据主键id做分片，同时仅在本地表做聚合，分布式表汇总

```
select
    sum(index_value),
    count(*)
from
(
SELECT
  sum(show_count) AS index_value,
  主键id
FROM
  ${CH表 }
WHERE
  day between '日期范围'
 group by 主键id
Settings
distributed_group_by_no_merge = 1
)t1
```

性能优化实践-场景2

百亿数据，count distinct计算uv响应慢

第二次优化（10.8s）

```
select
  sum(show_count)AS index_value,
  groupBitmap(rn) AS uv
FROM
  ${CH表 }
WHERE
  day between '日期范围'
```

关键问题解决：groupBitmap入参需要为数值，但主键id为字符串

分流数据

主键id	实验id	方案id
id1	1	1001
id1	2	2001
id2	1	1002
id3	2	2002

主键id	test_plan_id
id1	1-1001,2-2001
id2	1-1002
id3	2-2002

业务轻聚合数据

主键id	city	cnt
id1	sz	10
id1	nj	13
id2	nj	20
id3	hz	15

Spark2CH

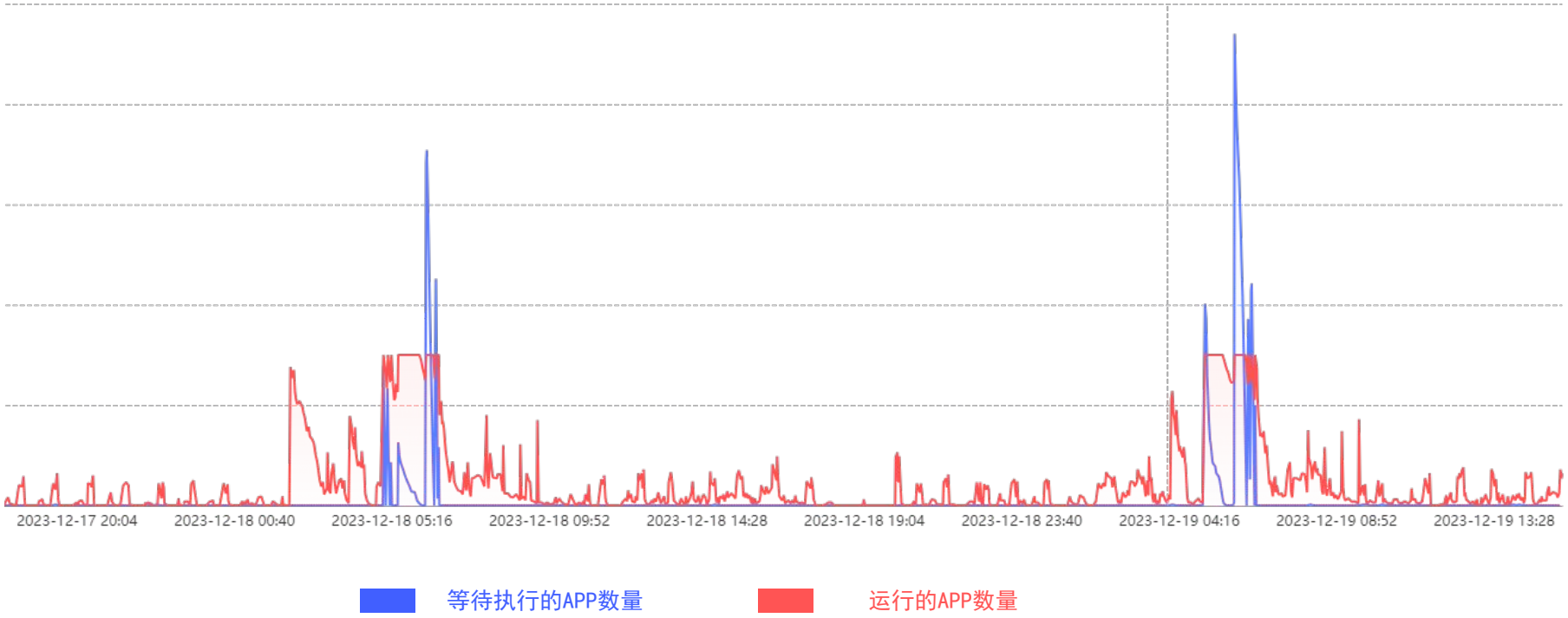
分桶排序，规避数据倾斜，提升计算效率

主键id	test_plan_id	city	cnt	rn
id1	1-1001,2-2001	sz	10	1
id1	1-1001,2-2001	nj	13	1
id2	1-1002	nj	20	2
id3	2-2002	hz	15	3

性能优化实践-场景3

CH计算定时触发，缓解Spark计算压力

yarn队列执行情况



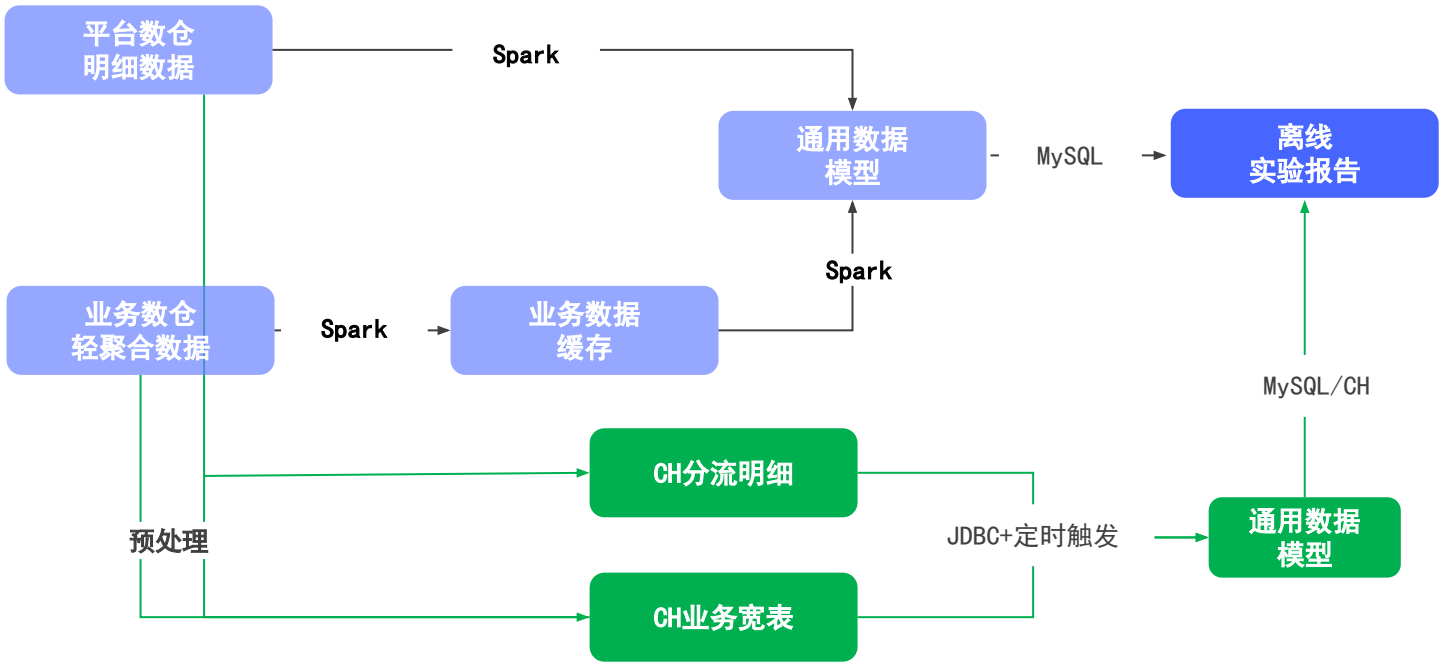
解决问题的思路

√ 任务打散

√ 提升计算效率

性能优化实践-场景3

CH计算定时触发，缓解Spark计算压力



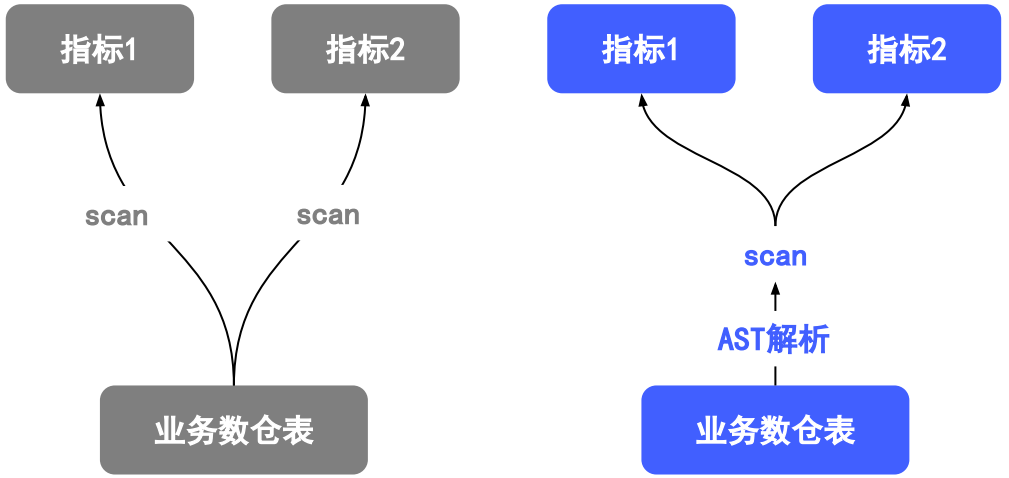
优化效果

时间(分钟)	min	avg	max
Spark	14	25	40
CH	0.03	0.4	1

未来规划

① 缩短实验效果输出周期，进一步提升决策效率

② 实验指标合并，提升计算效率



THANKS