大数据实时离线融合

及在唯品会的实战

姜伟华

唯品会
一家专门做特卖的网站

目录

CONTENTS

01 时效性与大数据

02 现状及问题

03 实时离线融合

04 实时离线融合带来的挑战

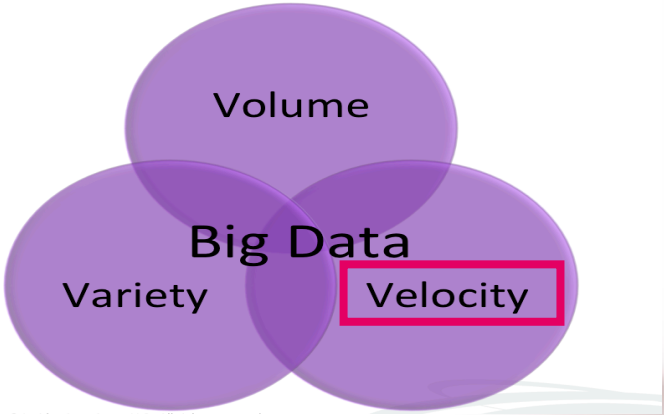
01 时效性与大数据

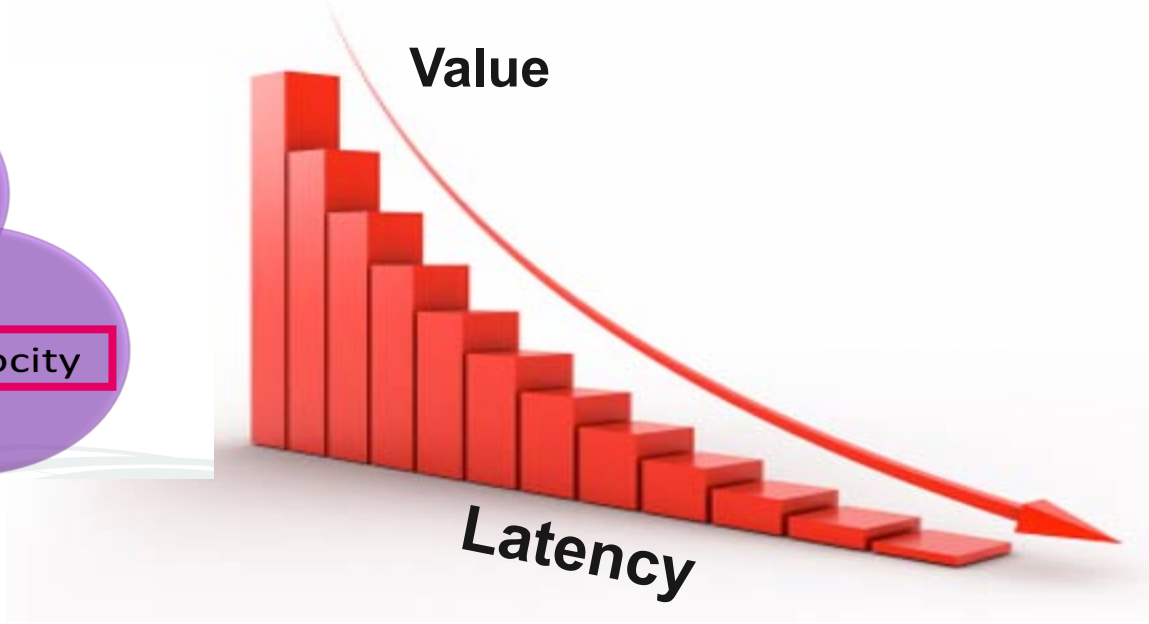
It is about the ability to make better decisions and take meaningful actions at the right time.

– Michael Minelli, Big Data, Big Analytics

时效性与大数据

Velocity（速度）：数据需要被及时处理，因为其价值会随着时间**快速**消失





时延：什么是实时？什么是离线？

时延：从数据产生到计算出结果的时间差

- 时延是端到端的，而不仅仅是Query的执行时间！！！
- 时延 = 数据准备时间 + 查询计算时间

毫秒、秒级时延

分钟级时延

时延 > 10分钟

什么是流处理，什么是批处理？

批处理
Batch Processing

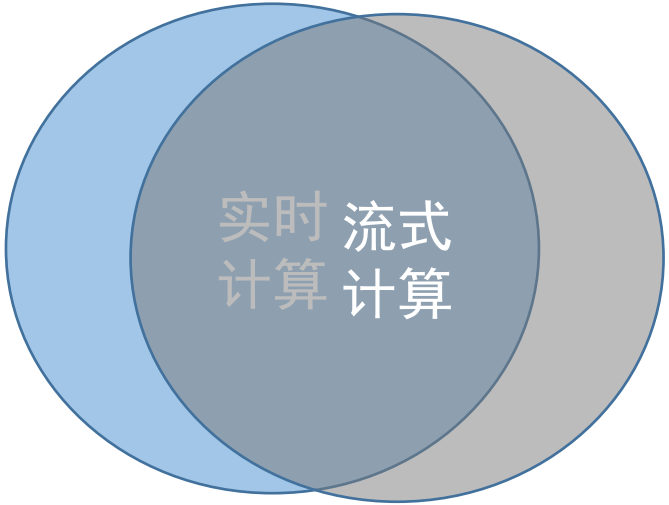
- 数据以一个完整的数据集被处理可以重复计算
- 数据先落盘，然后定时或者按需启动计算
- 一次处理的数据量大，延迟较大
- 经常需要全量计算
- 也称为“离线”

流处理
Streaming Processing

- 数据以流式的方式（增量）被处理
- 全内存计算，基本不落盘
- 一次处理的数据量小，延迟小
- 可以逐条计算，也可以mini-batch
- 也称为“实时”

实时计算 ≈ 流式计算

- 实时计算：大多是流式计算，但也可以用批处理来实现
- 流式计算：大多可以是实时或者准实时的，但也可能需要很长时间（比方说30分钟才出结果，例如使用30分钟的window）

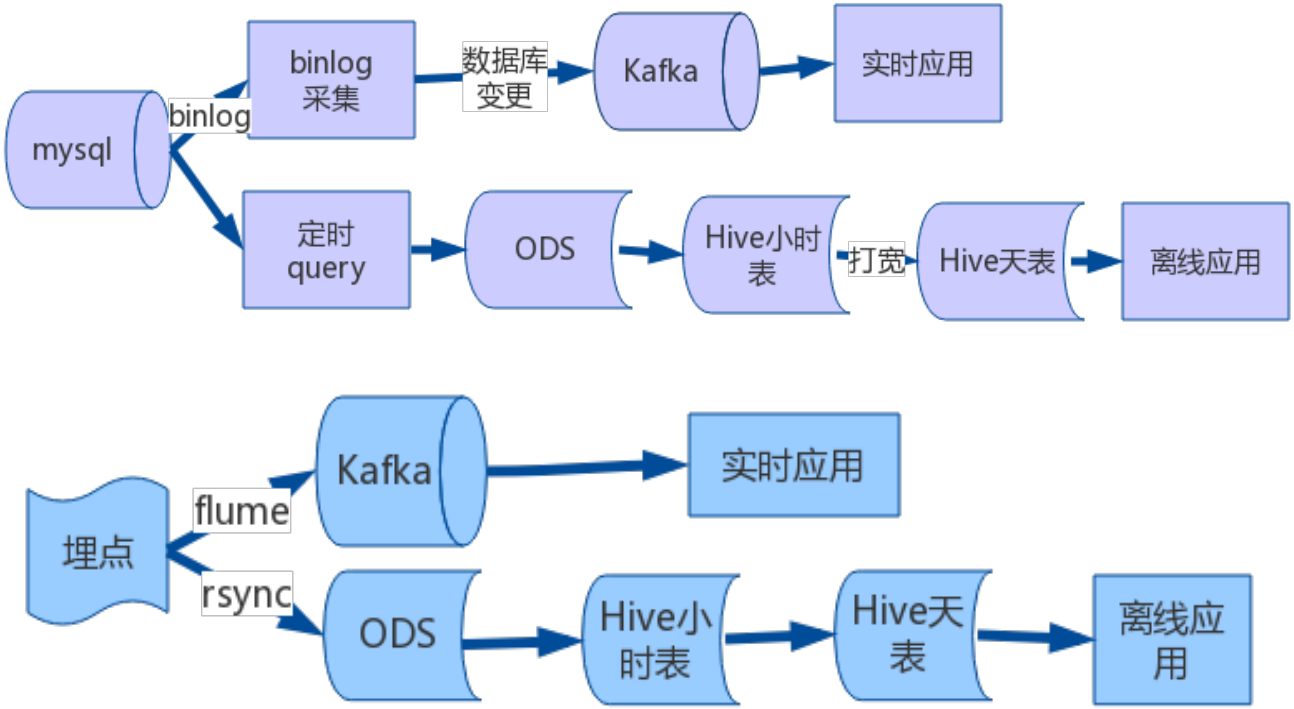


目前这两个词经常是互换使用的

02 现状及问题

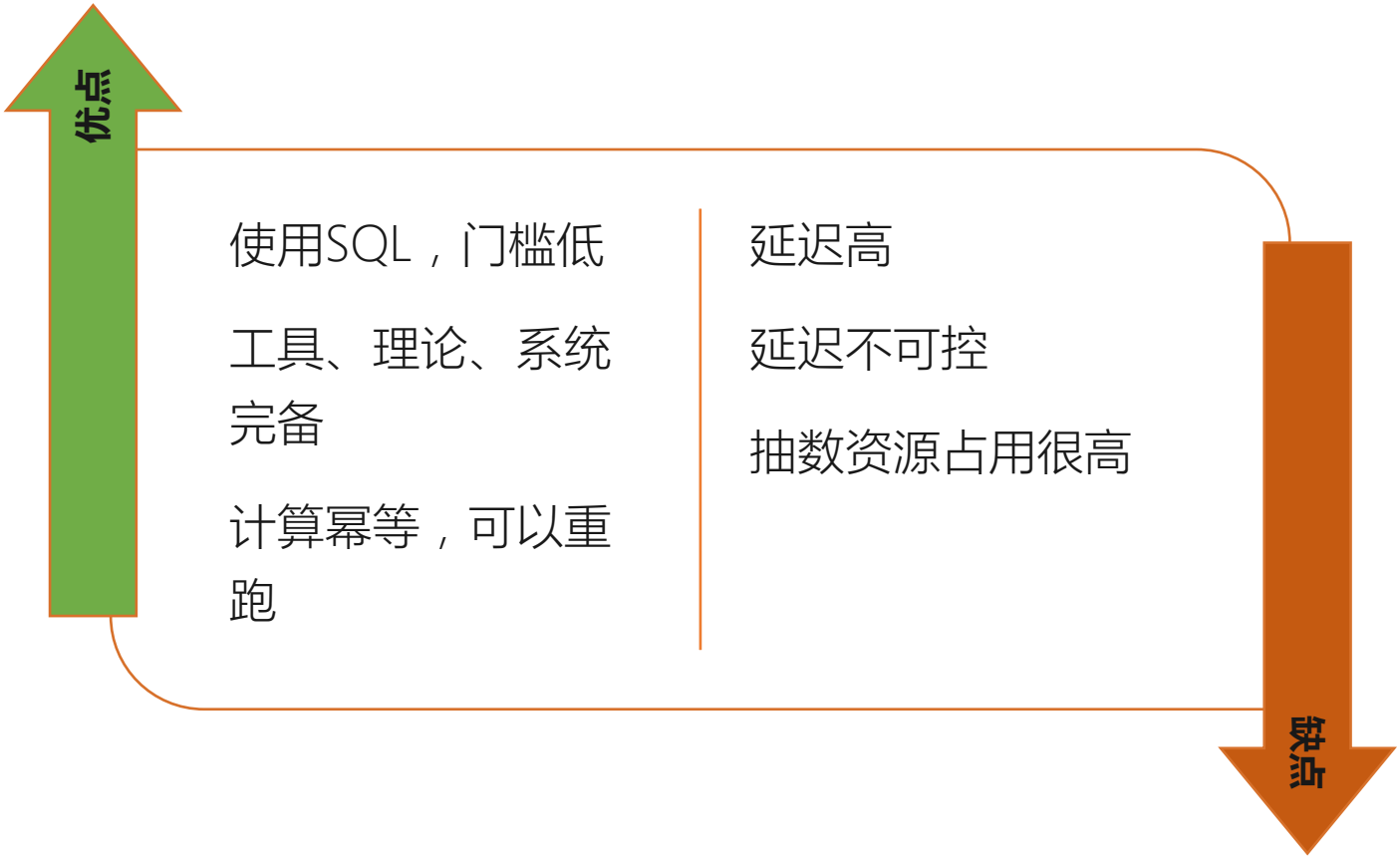
所有人都想实时，但只有实时团队有能力实时化

大数据时效性的发展：当前的典型架构

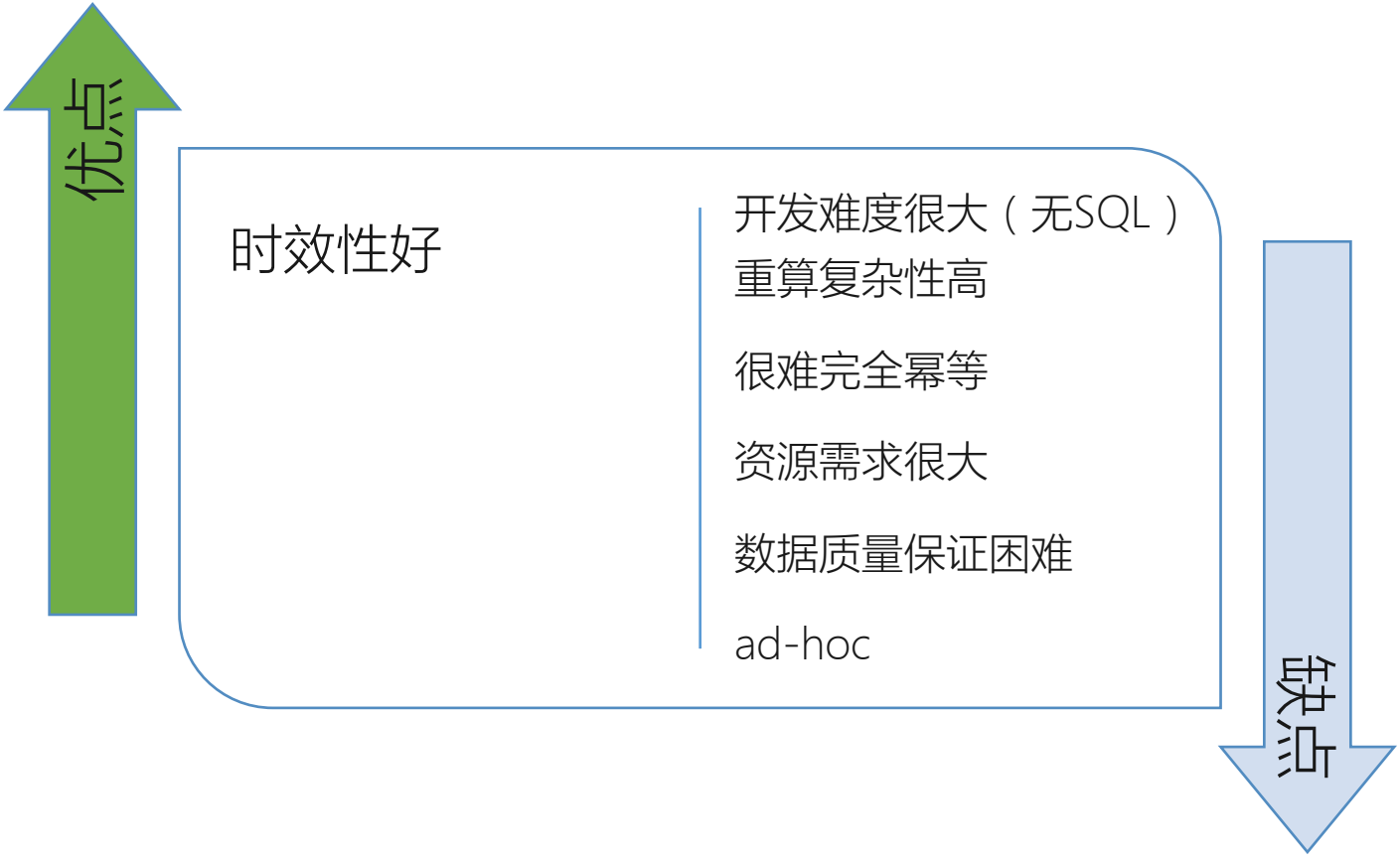


实时和离线是完全分离的两条路径

离线处理优缺点



目前大数据处理中的问题：实时处理



实时离线间的对数

- 典型场景：实时报表提供结果，但需要定时和离线报表去比对正确性 （Lambda Architecture）
- 一般认为离线的精度要高于实时
  - **大部分场景下是事实，部分场景下实时可能更加准确**

- 但实时和离线的处理方法是完全不同的：开发方式、方法，处理逻辑，持续性...
- 最根本的：**实时和离线从最本源开始就是两条计算路径**

•对数是一个脏活累活！

•对明细更加是痛苦到极点！

目前的简单选择方法

- 如果延迟要求在半小时以内 → **实时计算**（流式计算）
- 如果可以接受半小时以上的延迟 → **离线计算**

•原来业务方对时延要求比较低，大量的任务都是离线计算的
•但业务对时延的要求越来越高，所以被迫用流式计算来实现

*这个仅针对通用计算，特殊情况（比方说实时落地Hbase/ES，然后进行简单查询）不在这个讨论范围内

实时开发现状

- 实时同学疲于奔命，天天加班
  - 不理解业务
  - 功能点一个一个的对
  - 大促应该怎么应对？
- 业务方总是抱怨
  - 数据不准
  - 和离线对不上
  - 口径没更新
  - 开发效率低下，周期怎么这么长

03 实时离线融合

用户需要以他们容易掌控的方式实现近实时（NRT）

大数据处理之再思考

目前的实时化方法真的是正确的打开方式吗？

业务同学需要的是近实时（NRT）的结果

- 让业务同学自己用SQL开发

实时同学是平台层，应该专注于系统和平台，而不是业务

大数据处理之再思考（2）

1.计算资源优先于人力资源

2.是不是一定要用流处理来实现低时延？

1. **时延 = 数据准备时间 + 查询时间！**
2. 离线时延长是因为数据准备花了太多时间
3. 如果数据准备足够快+计算足够快，那离线计算也可以达到近实时（NRT）
4. 用流处理的方式解决NRT的问题，有杀鸡用牛刀之嫌

3.理想情况下的实现策略

1. 完全实时（real time）需求 →流式计算。需要提高数据质量和开发便利度
2. 近实时（NRT）需求 → 数据快速准备好（1~2分钟延迟）+快速离线计算

NRT数据准备

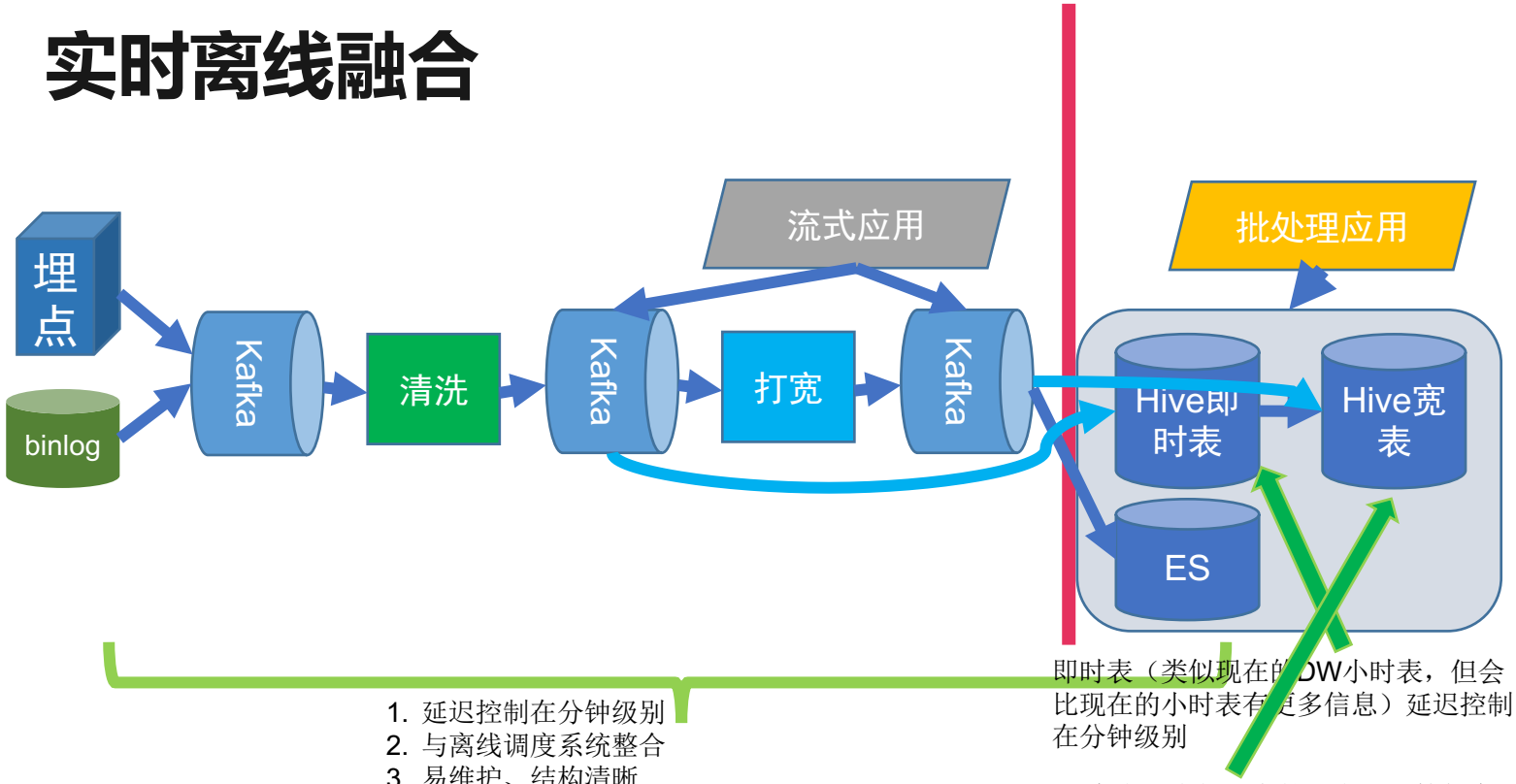
时延 = 数据准备时间 + 查询时间！
数据准备时间 = 定时调度时间 + 数据准备计算时间

- 只有两者都很小，数据准备时间才能短
- 高频调度 → Spark Streaming

- 那么，为什么不直接让流式计算提供实时+离线两边的公共数据层呢？

实时公共数据层：为离线和实时统一提供高质量、低延迟的基础数据！

实时离线融合



1. 延迟控制在分钟级别
2. 与离线调度系统整合
3. 易维护、结构清晰
4. 全链路监控，数据质量保证，考虑各种极端场景
5. 只处理基础数据，不涉及特定业务逻辑
6. 数据需要访问控制+审计
7. 严格的元数据管理
8. exactly once语义

数据落地Hive

实时数据落地Hive

- 分区：5分钟一个
- 文件打上timestamp （ event time)，每分钟一组文件，文件不跨分钟
- 晚到的数据（当前分区已经关闭）落地到下一个分区中

高频数据落地的问题：

- 小文件众多 vs 查询友好
- HDFS时延不稳定
- 离线调度系统无缝整合

数据落地Hive (2)

减少文件数

历史分区compact

减少落地波动

Alluxio

或者SSD文件系统

和离线调度系统整合

- Event time based watermark notification

如何加速离线应用

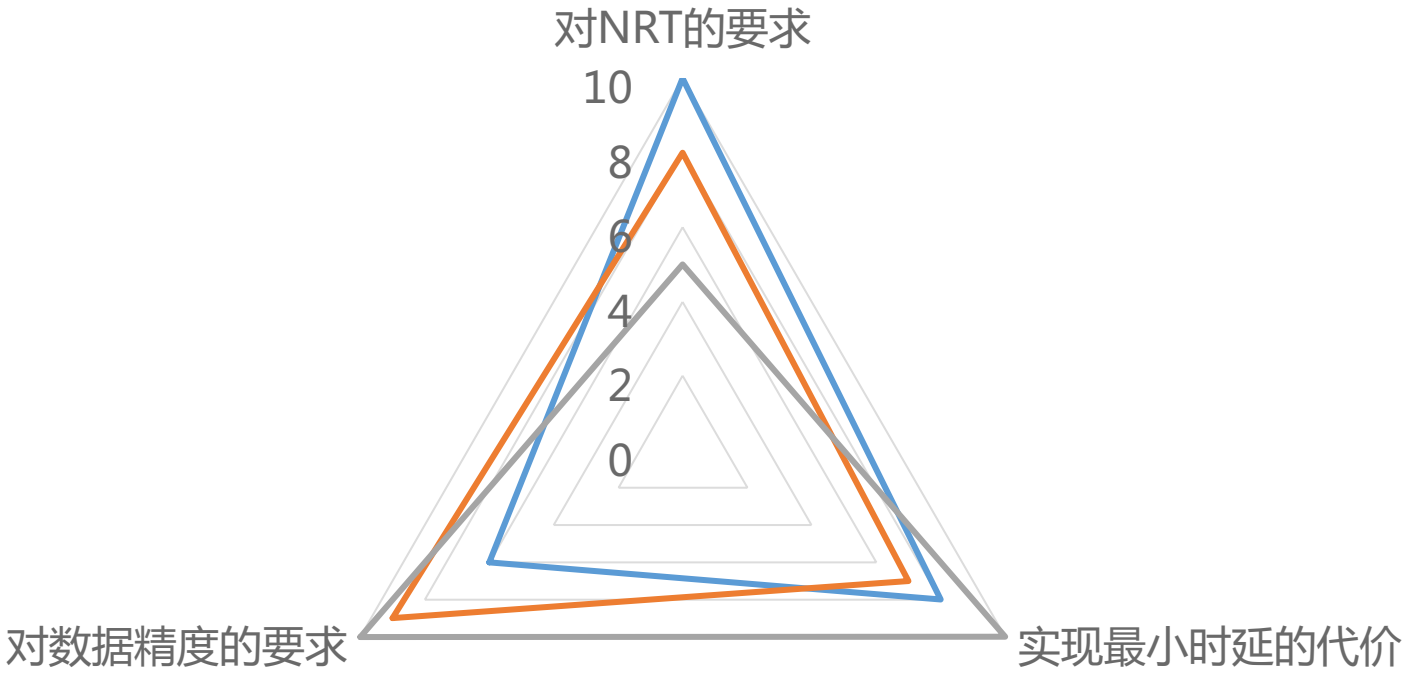
时延 = 数据准备时间 + 查询时间

查询时间 = 定时调度时间 + 查询计算时间

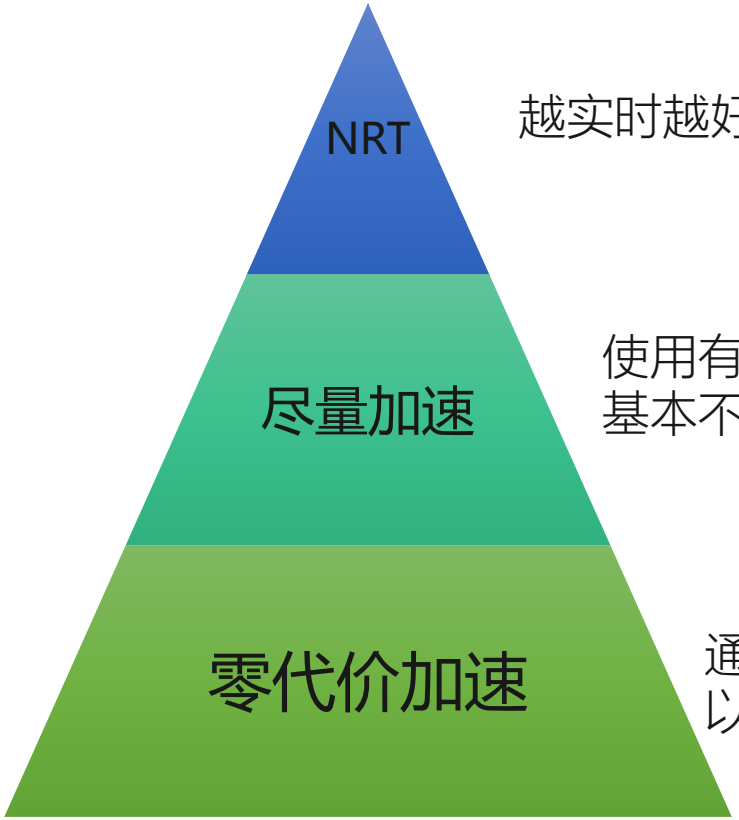
目标：定时调度时间（<5分钟） + 查询计算时间（秒到1~2分钟）

减少调度时间	减少查询计算时间
• 高频调度（5分钟甚至更少） • 调度系统如何改造？ • 如何控制资源使用量？	• 中间结果不落地 • 使用Spark SQL、Presto替代Hive • 全量计算 → 增量计算 • 使用ES、Druid、Kylin等做预计算

离线应用的3个维度



离线应用实时化：现存应用



越实时越好，不惜一切代价，完全重新实现逻辑

使用有限资源去加速，基本不增加计算负担：

1. Hive换成Presto/Spark SQ
2. 少量改写逻辑
3. 低成本的换用ES/Druid/Kylin

通过实时数据落地（提高调度频次），就可以透明的享受30~50分钟的加速

04 实时离线融合带来的挑战

世界上没有免费的午餐

实时的角色与挑战

- 作为平台而不是业务应用存在
  - 稳定性
  - 可靠性
  - 可管理性
  - 数据质量
  - SLA保证
- 现有实时框架尚不足以保证end-2-end exactly once
- 出了问题，如何确保下游不受影响？
- 如何确保数据质量？
  - 如何提供逻辑以供审计（SQL？）
  - 如何和离线对数

离线的角色与挑战

- 高频后对风吹草动的免疫能力下降
- 如何管理大量的高频任务调度？
  - 单独集群还是混合集群
  - 调度系统自身是否可以支持
  - 资源需求大增
- 如何区分热数据和冷数据？
  - 热数据需要单独的SSD或者Alluxio集群
- ES/Druid/Kylin的作用非常大
  - 大量需要实时化的需求都是简单查询
  - 容易开发，容易对数

平台中批处理/流处理的边界

- 实时的清洗、落地Hive/ES → 流处理
- 哪些打宽是离线和实时应用都需要的 → 流处理
  - 不做JOIN，只查表
- 哪些运算是批处理特别费时，而可以流式处理的 → 流处理
- 其他 → 批处理

感谢聆听

—

THANKS!

唯品会
一家专门做特卖的网站