www.vip.com

大数据之实时离线融合

姜伟华 博士
weihua.jiang@vipshop.com

2016-12-17

唯品会
vip.com
一家专门做特卖的网站

Agenda

- 个人介绍
- 时效性与大数据
- 现状及问题
- 实时离线融合

个人介绍

2005年复旦大学博士毕业

在Intel工作12年

2011年起从事大数据的研究和开发

- 我们团队创建了国内最早、影响最大的Hadoop发行版：IDH
- 2014年起负责Intel大数据开源+性能优化

2016年初加入唯品会

- 负责唯品会实时计算业务

IDH

国内最早、影响最大的Hadoop发行版

- 最早由Intel 上海团队开发，成功成为Intel全球业务
- 为国内行业应用开发了大量的定制功能
- 极大的促进了国内大数据的应用
- 2014年初和Cloudera的CDH合并

IDH产品开发经理

– 负责所有IDH的产品开发工作

大数据开源

带领团队为开源社区开发和优化大数据stack

- 为开源贡献了许多重要功能
  - MapReduce NativeTask
  - Hive on Spark
  - HBase Medium Object Store (MOB)
  - HDFS Erasure Coding
  - Sqoop 2
  - …

团队在2年时间内培养出10位committer

培育新的开源项目

Apache Gearpump：基于Akka的实时流处理引擎

- 2016年3月已进入Apache孵化器
- 和Google以及LightBend合作

Apache Kerby：最好的Java Kerberos实现（客户端、服务器端）

- 成为ApacheDS的子项目
- 即将替换MiniCluster成为Hadoop生态的标准Kerberbos实现
- 可以让Hadoop支持Oauth（Token）认证

上海大数据流处理Meetup

创建了上海大数据流处理Meetup

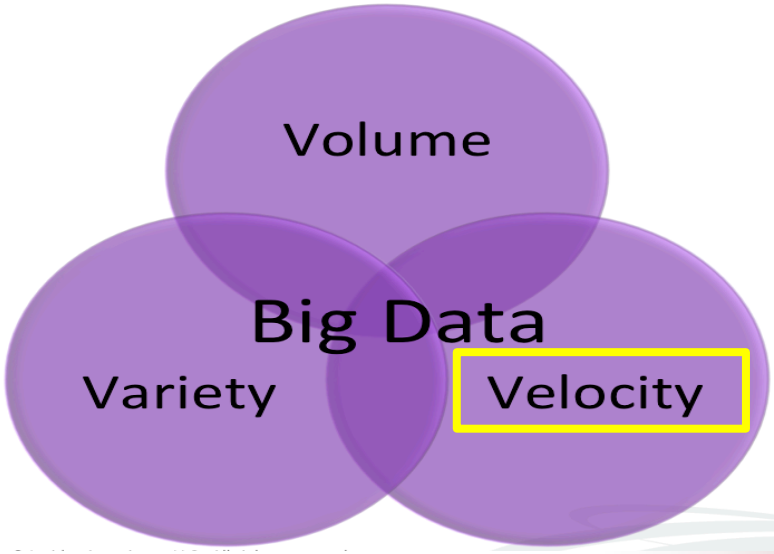
https://www.meetup.com/Shanghai-Big-Data-Streaming-Meetup/

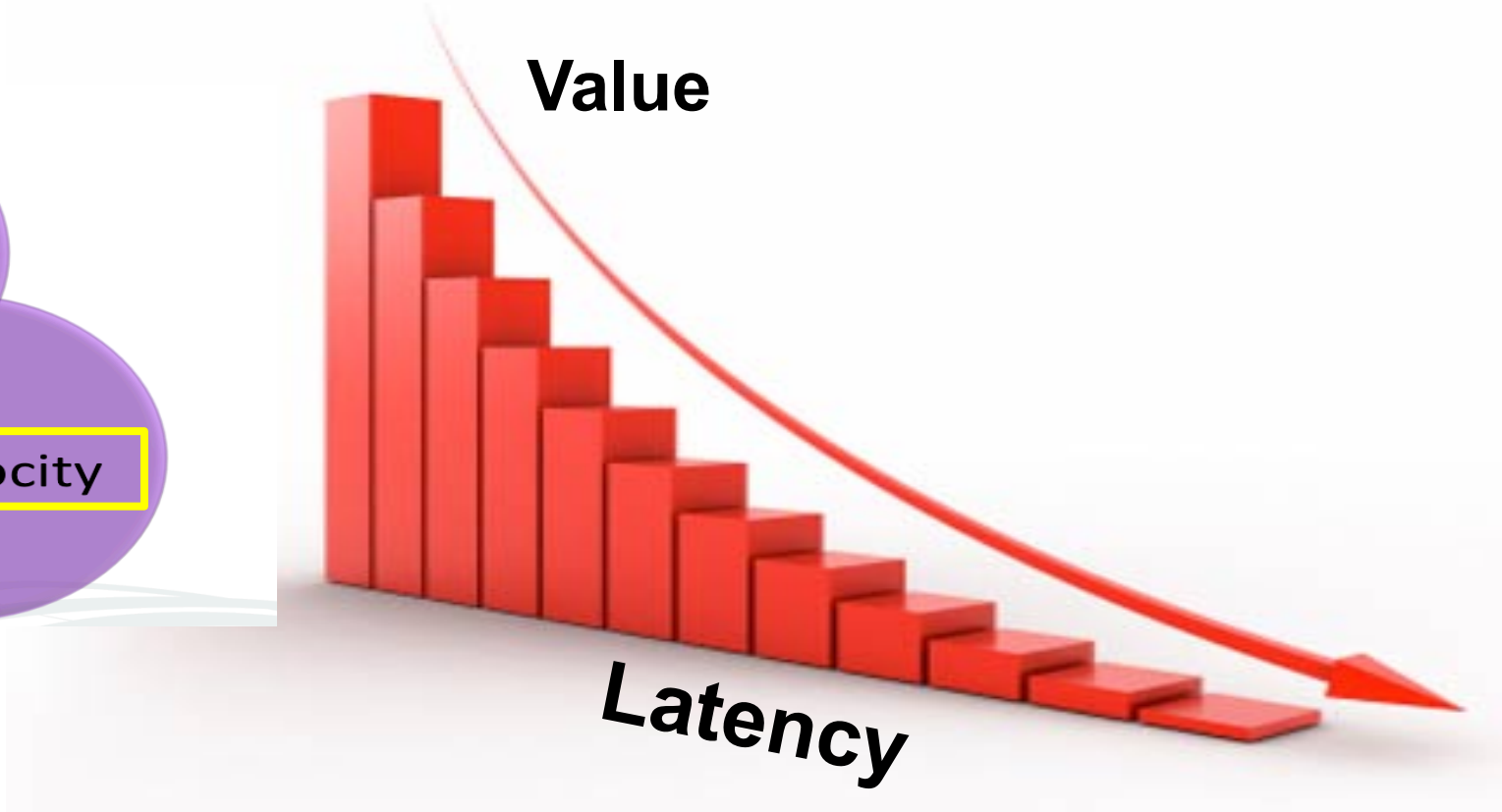
Agenda

- 个人介绍
- **时效性与大数据**
- 现状及问题
- 实时离线融合

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





什么是实时，什么是离线？

- **实时（Real-time）**：数据从产生到被消费中间无延迟或者延迟在毫秒、秒级
- **近实时（Near Real Time）**：延迟在1~5分钟级别
- **离线**：延迟比较大（例如：>5分钟）

延迟是端到端的，而不是Query的执行时间！！！

延迟 = 数据准备时间 + 查询计算时间

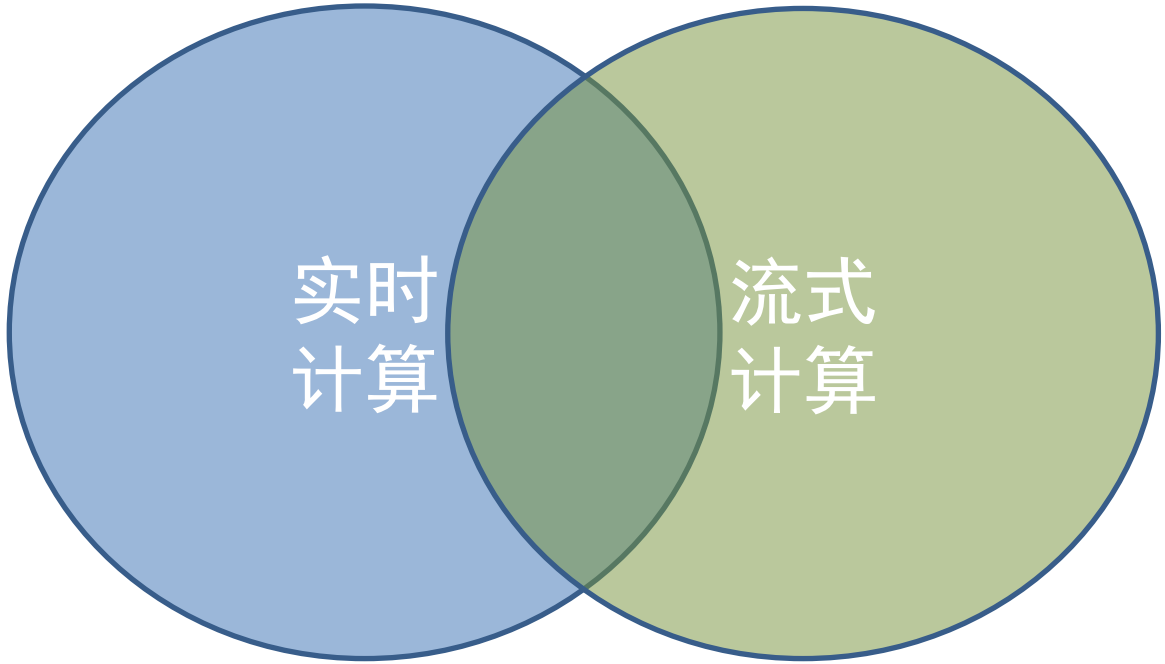
什么是流式计算，什么是批处理？

- **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
  - 一般数据会先落盘，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
  - 也称为“离线”
- **流式计算(Streaming Processing)**：数据以流式的方式（增量）被处理
  - 不可以重复计算
  - 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

大数据时效性的发展

2011年之前，用户对于使用MapReduce来处理HDFS上的数据已经非常满意了

- 解决了有和无的问题
- 数据一般是前一天的，MR执行需要分钟~小时
- 即使最简单的查询，都需要几十秒
- 无法和商业MPP数据库进行竞争

2012~2013年，用户广泛使用Hbase+Hive/MapReduce

- 数据实时落地Hbase，但一般HDFS还是前一天的数据
- 简单查询可以做到秒级，但复杂查询还是很慢
  - 2012年我们推出了Hive over HBase功能，广受客户欢迎
- HBase限制很多，数仓还是HDFS为主

大数据时效性的发展（2）

2014年~ Now 的典型架构

- 实时：
  - 数据采集（flume）、数据总线（kafka）、KV Store（Redis）、流式计算（Storm、Spark Streaming）
  - 全链路延迟控制在ms~分钟

- 离线：
  - 典型的数据仓库架构（ODS、DW、DM），经过多年建设很完善
  - 数据定时采集到HDFS，形成Hive小时表
  - 每天生成Hive天表
  - 计算使用Hive/Spark/Presto/Impala等，计算延迟在分钟到小时级别
  - OLAP工具开始流行（Kylin， ES），对于简单查询（特别是预定义维度的查询）可以在秒级甚至毫秒级完成

Agenda

- 个人介绍
- 时效性与大数据
- **现状及问题**
- 实时离线融合

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，然后进行简单查询）不在这个讨论范围内

大数据开发方法现状

离线：SQL为主、代码为辅

- 完善的SQL支持 （Hive、Spark SQL、Presto）
- 完善的开发工具支持（各种SQL开发环境、IDE）
- 完善的数据仓库支持（理论、元数据管理、数据清洗、数据组织、权限控制、异数据源的集成…）
- 开发门槛很低

实时：代码为主

- SQL支持非常不完善
- 开发工具很少
- 数据基本无管理（大家基本都是消费raw data）
- 开发门槛很高，需要对流式计算的理论有一定了解，强调经验和技巧，陷阱多
- 复杂点的问题需要非常高的技能，例如
  - 两个流的JOIN
  - 关联一个很大的码表和一个很高TPS的流

目前大数据处理中的问题：离线处理

离线延迟很高

- 采集到ODS（5~30分钟）、ODS到小时表（10~20分钟）
- 基于小时表的计算（若干分钟）--> 延迟在0.5~1.5小时
- 基于天表的计算 → 延迟在一天

系统资源占用很高

- 需要有大量的定时任务。比方说128个分数据库 → 128个定时任务
- 在一个典型的系统中，有几千到上万个定时任务
- 峰谷很明显
- 对计算平台的压力很大（各种错峰、队列...）

延迟不可控

- 比方说大促时，延迟可能增加几倍

目前大数据处理中的问题：实时处理

计算准确性不高

- 需要和离线对数，Lambda Architecture
- 目前流框架还无法保证end-to-end exactly once
- 无法重算

实时计算开发难度很大

- 无法通用化，ad-hoc，error prone

资源需求很大

- 为了保证实时性，计算资源需要独占
- 需要大量的高速存储（Redis、Hbase）等等

质量不能保证

- 都从裸日志开始消费，大量重复计算、资源浪费
- 数据质量很差，无全链路监控，对异常和上游变更的抵抗能力很差

无法提供高级的数据服务

- 比方说，对加购物车行为按照人群和商品部类两个维度分别实时统计金额
  - 埋点流只能告诉我们加购物车的user id和商品id

目前大数据处理中的问题：实时离线间的对数

一个典型的场景：实时提供结果，但需要定时和离线去比对正确性（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：

- 业务开发方法不同：离线SQL，实时代码
- 处理逻辑不同：离线对静态数据集，实时对动态数据流
- 处理方法不同：离线可精确计算，实时很多时候必须近似计算
- 精度不同：实时可通过watermark在event time上计算，离线只能是对划定的时间段数据进行计算
- 持续性不同：实时可对数据进行持续修正，离线不一定会修正。比方说订单退单
- 路径不同：**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！

Agenda

- 个人介绍
- 时效性与大数据
- 现状及问题
- **实时离线融合**

大数据处理之再思考

延迟 = 数据准备时间 + 查询时间！

1. 计算的本质是什么？
    1. 批处理和流处理都只是手段，本质上就是在合适的限定条件（比方说时延）下，完成用户所期望的计算

2. 时延的本质是什么？是不是一定要用流处理来实现低时延？
    1. 时延长是因为数据准备花了太多时间
    2. 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）

3. 如何才是对用户最友好的计算方法？开发人员的期望是什么？
    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

离线数据准备的近实时（NRT）暴力解法

假设我们忽略资源限制，暴力的加快小时表和天表的计算频度。是否可以实现近实时的数据准备？比方说，小时表每分钟计算一次，天表每10分钟计算一次？

很不幸，随着计算频度的提高，数据的精度会极大的下降

- 数据是乱序的（业务要求按照event time来统计，但数据是按照process time到达的）
- 数据会晚到（数据可能会到达的比较晚，极端情况下可能有小时级的延迟）

比方说，要统计每分钟的PV，你对小时表按照event time去统计，但最近一分钟的数据是非常不准的

- 最近一分钟的数据还没完全到达
- 每分钟的分片里有其他分钟的数据

原来离线小时表和天表之所以有足够的精度是因为其时间长度足够大，使得这两个因素的影响非常小

这种做法的一个极致就是Spark Streaming!!!

离线数据准备的近实时解法

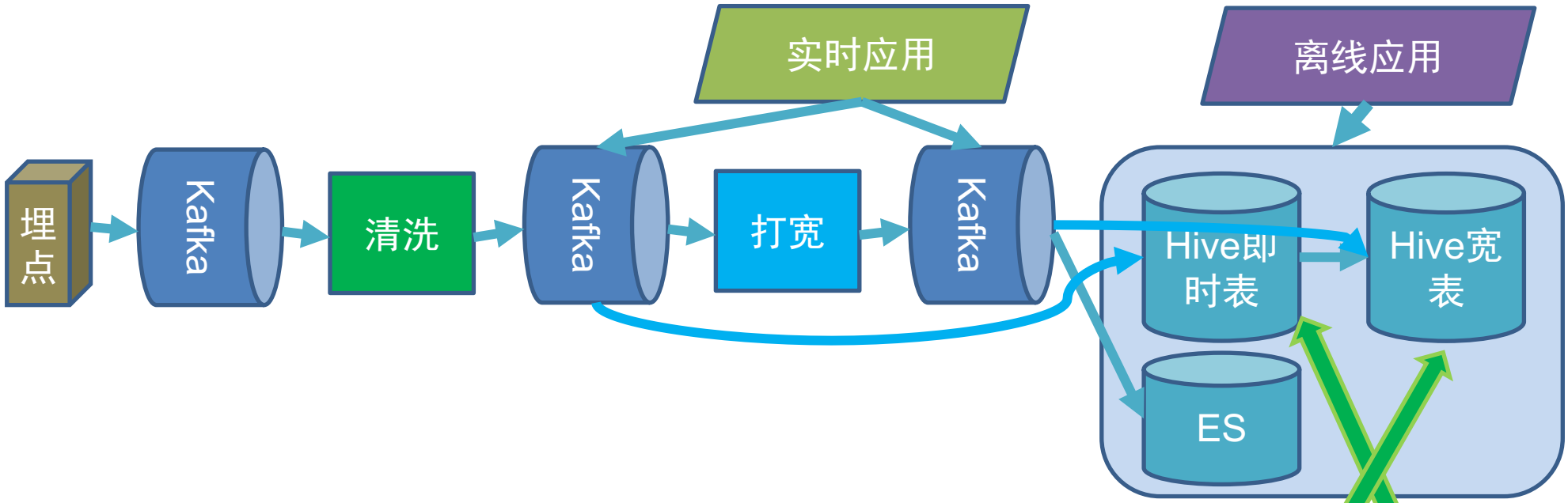
我们看到数据准备的越实时，那么实现方式就越接近于流式计算

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？

1. 提高时效
2. 节省资源

实时公共数据层：为离线和实时统一提供高质量、标准化、低延迟的基础数据！

实时离线融合



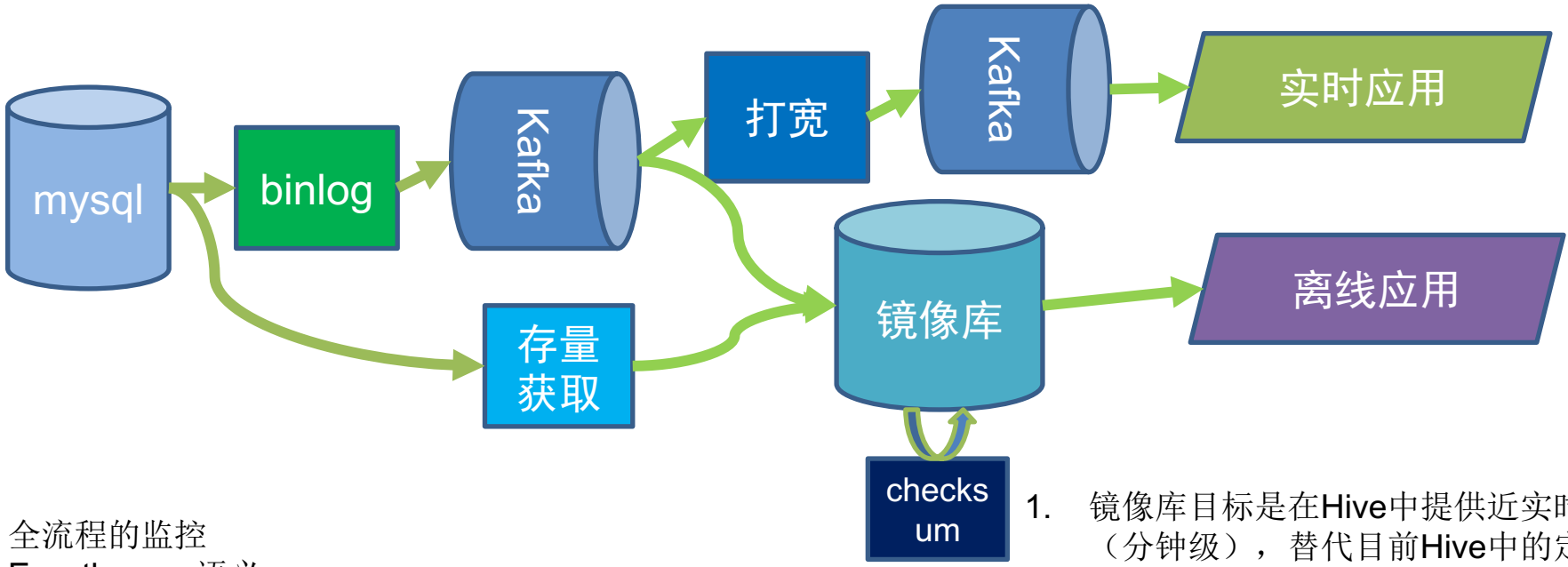
1. 延迟控制在分钟级别
2. 与离线调度系统整合
3. 易维护、结构清晰
4. 全链路监控，数据质量保证，考虑各种极端场景
5. 只处理基础数据，不涉及特定业务逻辑
6. 数据需要访问控制+审计
7. 数据schema可查询、可更新（向后兼容）
8. exactly once语义

即时表（类似现在的DW小时表，但会比现在的小时表有更多信息）延迟控制在分钟级别

宽表（类似现在的天表）计算频度可能会是在小时级别（分不同频度）
通过多种手段加速计算

实时离线融合

打宽是为了解决实时应用需要多表关联的问题
打宽控制延迟在秒级



1. 全流程的监控
2. Exactly once语义
3. 数据质量保证，考虑各种极端场景
4. 只处理基础数据，不涉及特定业务逻辑

1. 镜像库目标是在Hive中提供近实时的数据（分钟级），替代目前Hive中的定期mysql获取任务
2. 目前在测试Cassandra
3. 镜像数据入库是通过存量+增量来获取全部数据。同时通过定期checksum来确保数据质量

设计原则

- 有完善的数据字典，并且数据必须强格式
- 区分基础数据和业务数据，不针对特定业务逻辑
- 无缝和离线系统整合
- 离线和实时共用同一来源，极大的简化对数与逻辑移植
- 离线现有数据表可以降低频度存在，作为对数检验之用
- 严格的数据质量保证
  - 流式计算的步骤必须非常严谨，把各种极端情况和意外情况都考虑进去。要保证长期稳定可靠运行，在任何情况下都不丢数据
  - 要支持exactly once和数据重算
  - 全链路的监控
  - 各个业务方可以向这个框架提供数据，各自保证数据质量

实时离线边界

第一步，实时只负责清洗，近实时生成小时表，加速离线数据准备

如果某种数据准备是实时应用和离线应用都需要的，那么就前置到实时来做

- 比方说订单关联满减

如果某种数据准备离线很耗时，实时实现很方便，那么就实时来做

否则，就保持离线不动

数据落地Hive

实时数据落地Hive

- 分区：5分钟一个
- 晚到的数据（当前分区已经关闭）落地到下一个分区中
- 历史分区compact，减少文件数

Time是基于event time，而不是process time

Event time based 分区关闭 notification（5分钟通知一次）

离线怎么使用落地Hive的数据

访问方式：标准的Hive SQL表访问

两种模式：

1. 连续访问式。不关心数据是否全到了，只关心是否能及时访问到最新的数据。
    1. 对当前可访问到的所有数据文件进行查询
    2. 可以在任何时刻发起
    3. 查询结果可能会有误差，需要修正
    4. 延迟小，数据落地延迟在1~2分钟
2. 定期式。任务必须等特定时间的数据都到达了，才能开始计算。这样结果是精确的
    1. 需要和调度系统结合，等待分区关闭通知
    2. 适合报表等需要精确数据的应用
    3. 延迟较大，数据落地延迟在5~6分钟

如何加速离线SQL

延迟 = 数据准备时间 + 查询时间

当我们把数据准备时间缩短到1~2分钟的时候，那么查询时间就成为一个重要的限制因素了。

我们需要提升查询时间：

1. 使用ES、Kylin等做预计算
2. 将实时落地的数据直接落地在Alluxio等内存文件系统
3. 对于多个SQL串联的JOB改成使用Spark SQL等在内存中进行计算

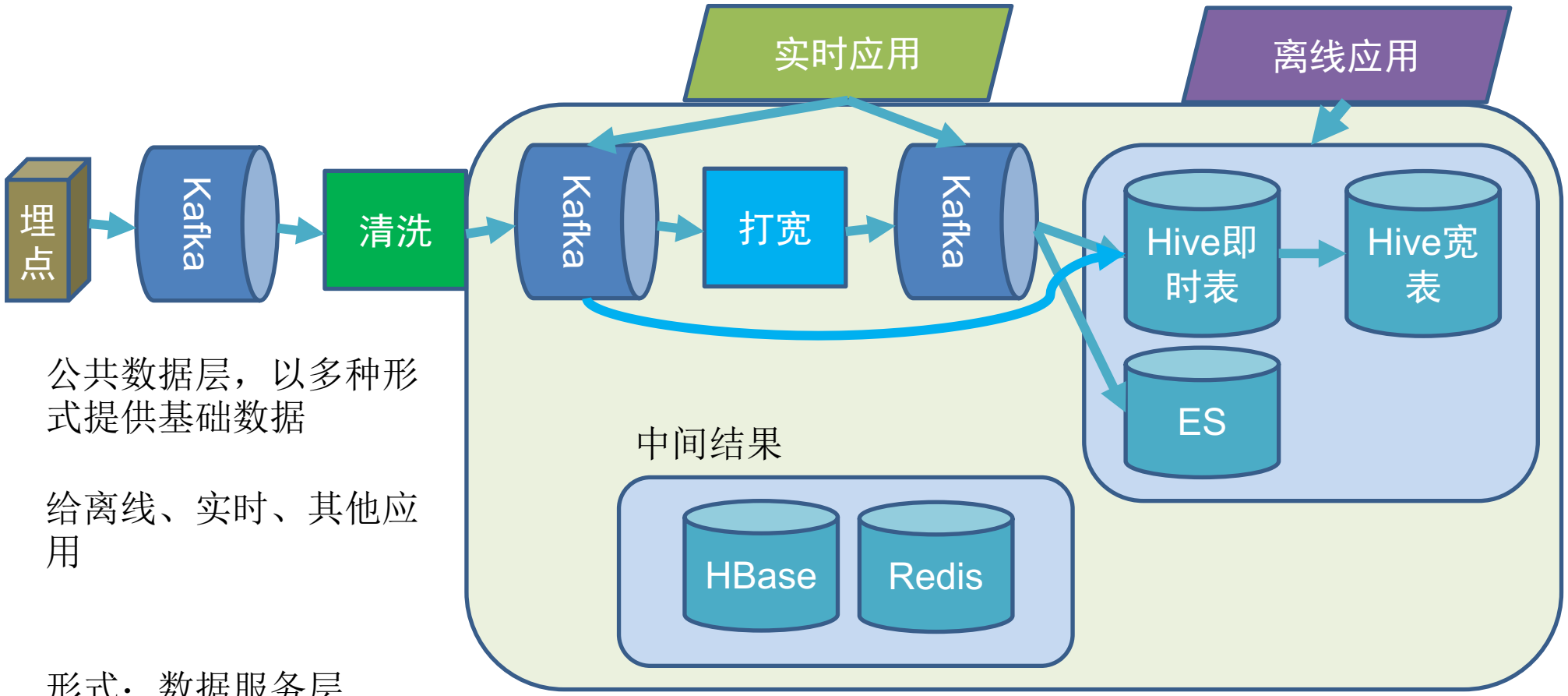
目标：大部分简单查询时间在秒级~分钟级别

应用

以最小的开发代价+可承受的计算代价去最快的产生结果

1. 数据维度是已知的 → Kylin
2. 常见的简单查询 → ElasticSearch
3. 复杂查询（精度要求不高，近实时） → Presto/Hive/Spark SQL访问连续数据
4. 精确结果 → 定期式计算
5. 时延要求很高 → 流式计算

公共数据层



公共数据层，以多种形式提供基础数据

给离线、实时、其他应用

形式：数据服务层 RESTful API，数据流、Hive表…

如果有了公共数据层

那么对于业务来说，

1. 如果可以直接通过数据服务层获取所需的数据，那么服务API获取Redis或者ES、Kylin的结果
2. 如果可以用SQL来访问离线Hive即时表或者宽表来实现的，那么就离线自助SQL
3. 如果延迟要求很高或者更加适合实时的，那么就流式任务
    1. 流式任务也需要SQL化，这样逐步消除实时和离线的区别

www.vip.com

THANK YOU！

唯品会
vip.com
一家专门做特卖的网站