> 关注公众号：大数据技术派，回复：资料，领取1024G资料。

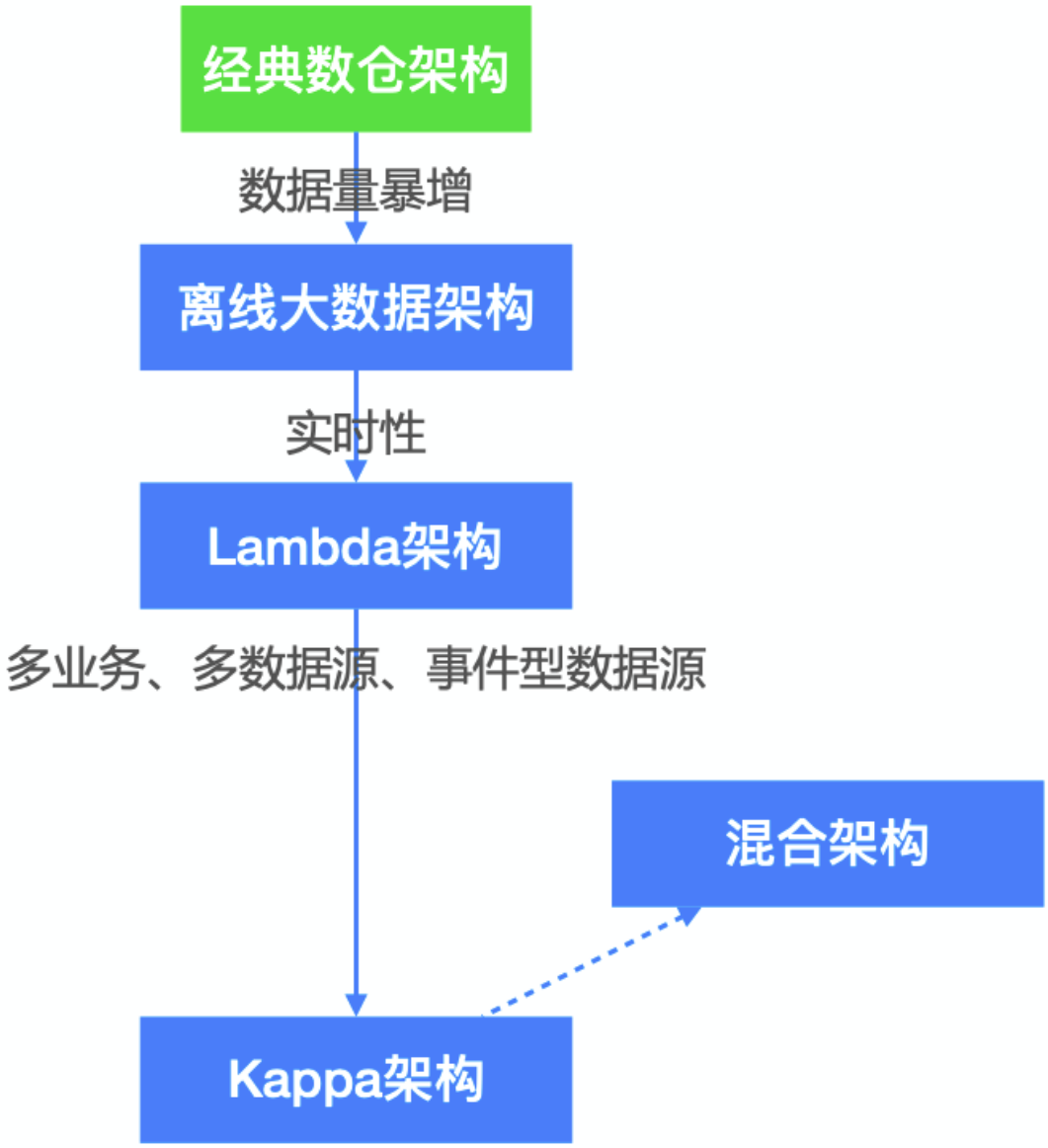
主要内容

发展史

时代的变迁，生死的轮回，历史长河滔滔，没有什么是永恒的，只有变化才是不变的，技术亦是如此，当你选择互联网的那一刻，你就相当于乘坐了一个滚滚向前的时代列车，开往未知的方向，不论什么样的技术架构只有放在当前的时代背景下，才是有意义的，人生亦是如此。

时间就是一把尺子，它能衡量奋斗者前进的进程；时间就是一架天平，它能衡量奋斗者成果的重量；时间就是一架穿梭机，它能带我们遨游历史长河,今天我们看一下数仓架构的发展，来感受一下历史的变迁，回头看一下那些曾经的遗迹。准备好了吗 let's go!，在此之前我们先看一下，数据仓库在整个数据平台中的地位

开始之前，我们先上一张大图，先有一个大概的认知，从整体到局部从概括到具体，看一下导致机构变化的原因是什么，探究一下时代背景下的意义，我们顺便看一下什么是数仓



那么什么是数仓，数据仓库是一个面向主题的（Subject Oriented）、集成的（Integrate）、相对稳定的（Non-Volatile）、反映历史变化（Time Variant）的数据集合，用于支持管理决策，数据仓库在数据平台中的建设有两个环节：一个是数据仓库的构建，另外一个就是数据仓库的应用。

数据仓库是伴随着企业信息化发展起来的，在企业信息化的过程中，**随着信息化工具的升级和新工具的应用，数据量变的越来越大，数据格式越来越多，决策要求越来越苛刻，数据仓库技术也在不停的发展**
这就是架构升级的原因，其实就是外部环境变了，现有的体系不能满足当前的需求，既然找到了原因，我们就来欣赏一下**历史长河中哪些闪亮的星**

> “我们正在从IT时代走向DT时代(数据时代)。IT和DT之间，不仅仅是技术的变革，更是思想意识的变革，IT主要是为自我服务，用来更好地自我控制和管理，DT则是激活生产力，让别人活得比你好”
>
> ——阿里巴巴董事局主席马云。

经典数仓

在开始之前，我们先说一点，其实数据仓库很早之前就有了，也就是说在离线数仓之前(基于大数据架构之前)，有很多传统的数仓技术，例如基于Teradata的数据仓库，只不过是数据仓库技术在大数据背景下发生了很多改变，也就是我们开始抛弃了传统构建数仓的技术，转而选择了更能满足当前时代需求的大数据技术而已，当然大数据技术并没有完整的、彻底的取代传统的技术实现，我们依然可以在很多地方看见它们的身影

经典数仓可以将数仓的数仓的不同分层放在不同的数据库中，也可以将不同的分层放在不同的数据库实例上，甚至是可以把不同的分层放在不同的机房

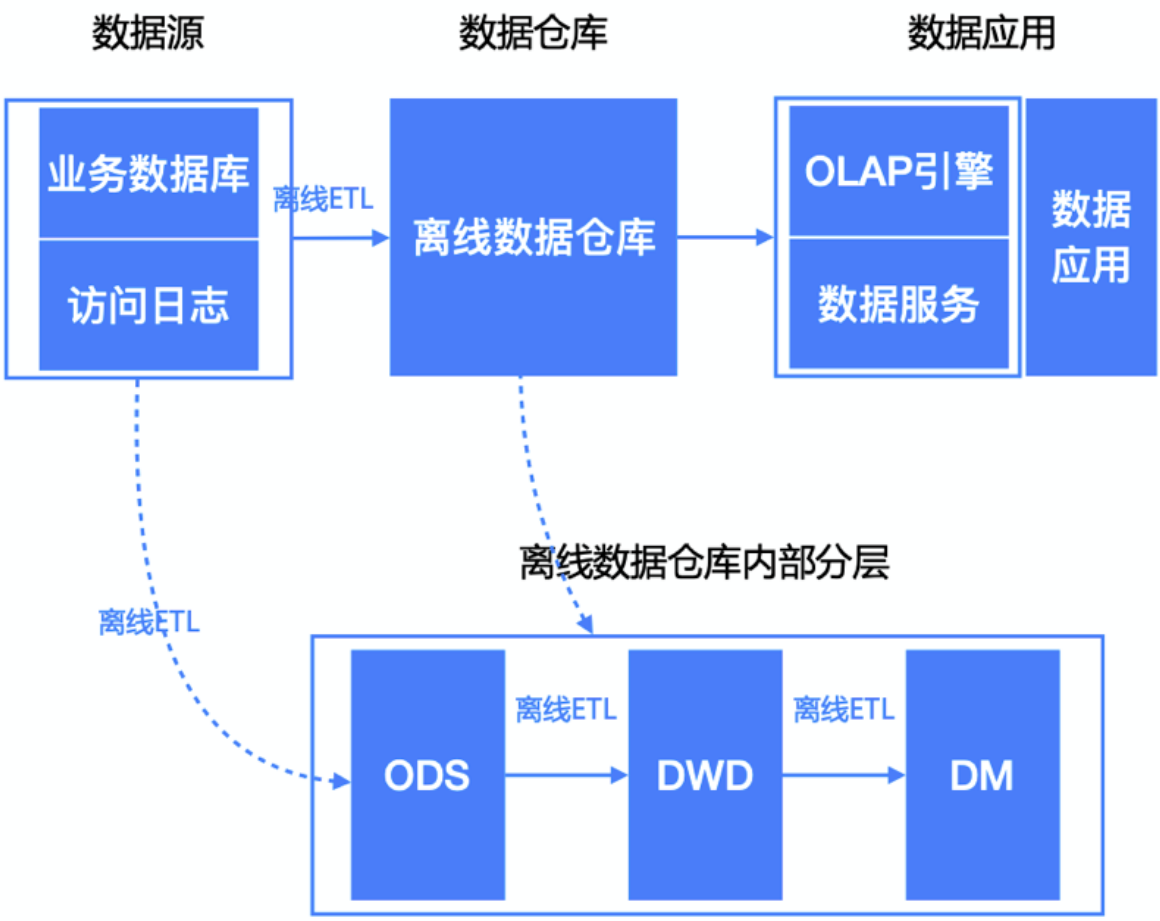
大数据技术改变了数仓的存储和计算方式，当然也改变了数仓建模的理念，例如经典数仓数据存储在mysql等关系型数据库上，大数据数仓存储在hadoop平台的hive中（实际上是HDFS中），当然也有其他的数仓产品比如TD、greenplum等。

离线数仓(离线大数据架构)

随着互联网时代来临，数据量暴增，开始使用大数据工具来替代经典数仓中的传统工具。此时仅仅是工具的取代，架构上并没有根本的区别，可以把这个架构叫做离线大数据架构。

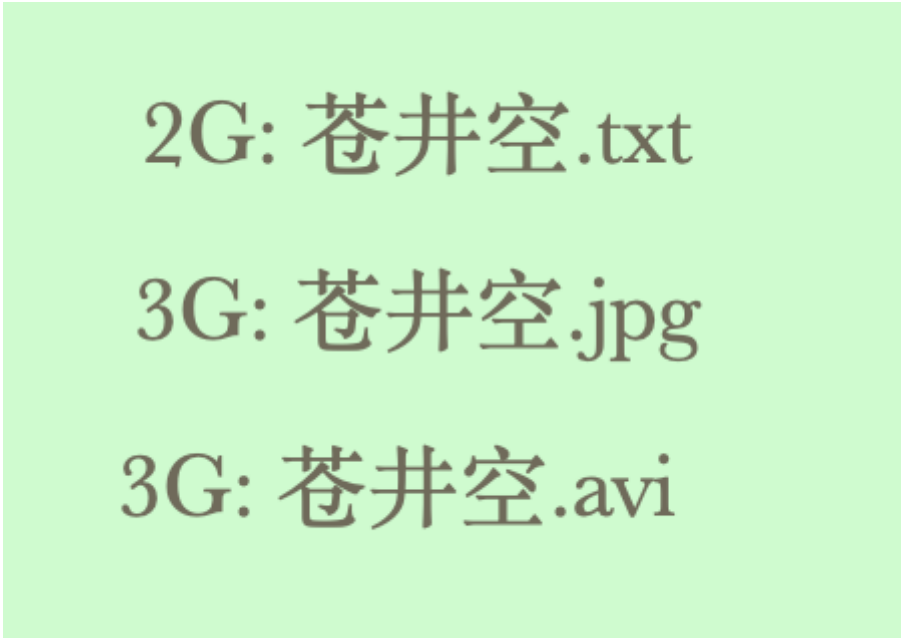
随着数据量逐渐增大，事实表条数达到千万级，kettle等传统ETL工具逐渐变得不稳定，数据库等存储技术也面临着存储紧张，每天都陷入和磁盘的争斗中单表做拉链的任务的执行时间也指数级增加，这个时候存储我们开始使用HDFS而不是数据库；计算开始使用HIVE(MR)而不是传统数仓技术架构使用的kettle、Informatica 等ETL工具；

公司开始考虑重新设计数仓的架构，使用hadoop平台的hive做数据仓库，报表层数据保存在mysql中，使用tableau做报表系统，这样不用担心存储问题、计算速度也大大加快了。在此基础上，公司开放了hue给各个部门使用，这样简单的提数工作可以由运营自己来操作。使用presto可以做mysql、hive的跨库查询，使用时要注意presto的数据类型非常严格。

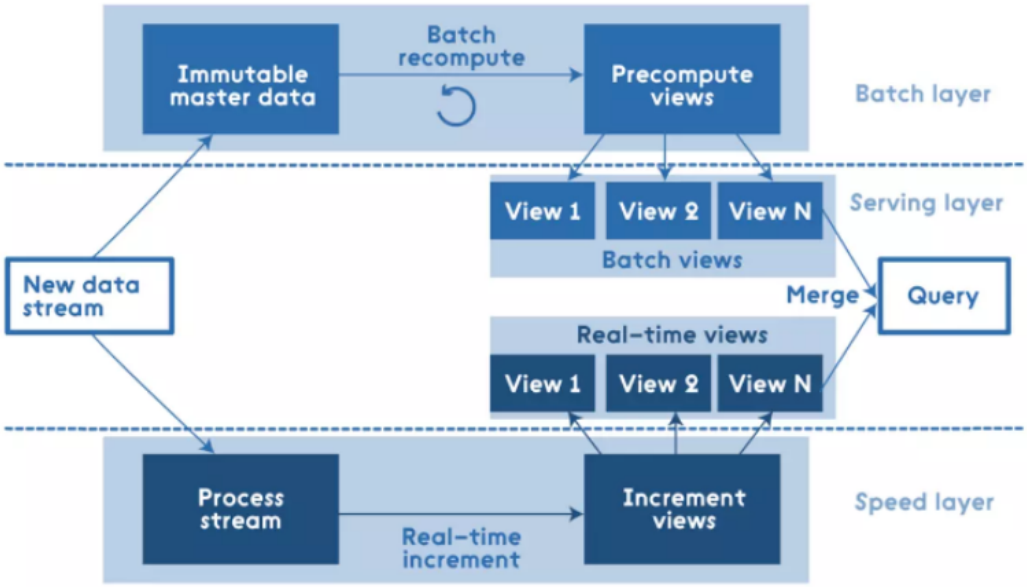


Lambda架构

后来随着网络技术、通信技术的发展，使得终端数据的实时上报传输成为可能，从而业务实系统发生变化，进而导致了我们对需求的时性要求的不断提高开始之前我们先看一下，网络技术和通信技术到底对我们的生活有什么样的影响



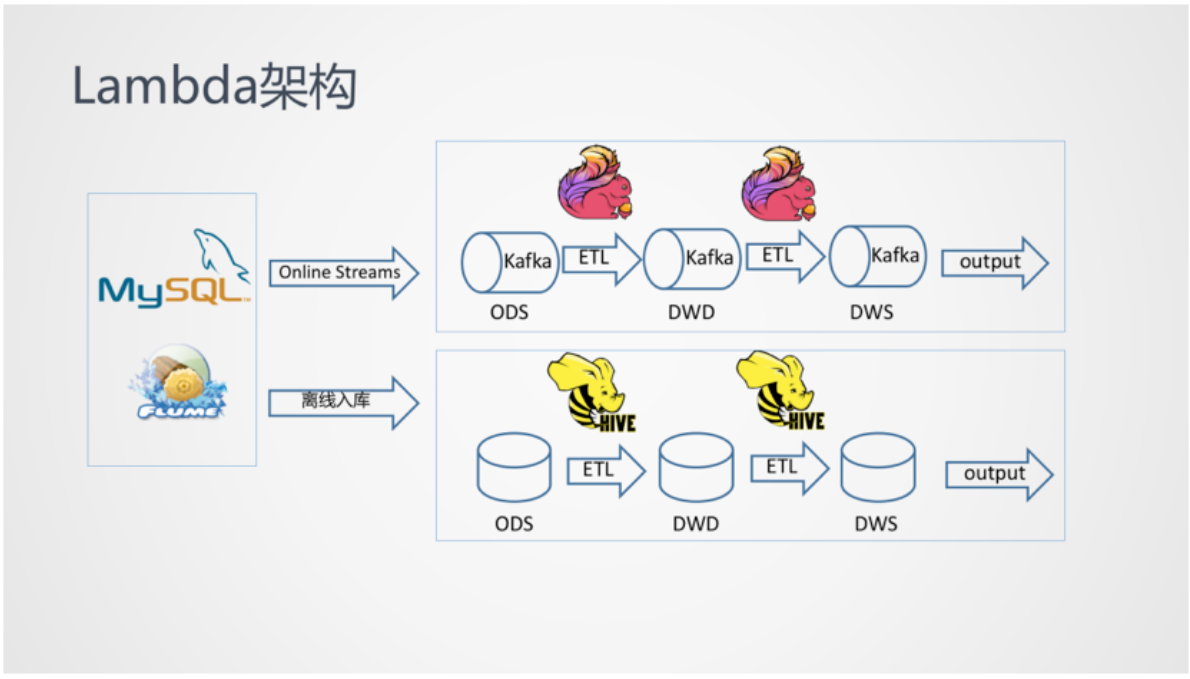
为了应对这种变化，开始在离线大数据架构基础上加了一个加速层，使用流处理技术直接完成那些实时性要求较高的指标计算，然后和离线计算进整合从而给用户**一个完整的实时计算结果**，这便是Lambda架构。



为了计算一些实时指标，就在原来离线数仓的基础上增加了一个实时计算的链路，并对数据源做流式改造（即把数据发送到消息队列），实时计算去订阅消息队列，直接完成指标增量的计算，推送到下游的数据服务中去，**由数据服务层完成离线&实时结果的合并。**

需要注意的是流处理计算的指标批处理依然计算，最终以批处理为准，即每次批处理计算后会覆盖流处理的结果（这仅仅是流处理引擎不完善做的折中），Lambda架构是由Storm的作者Nathan Marz提出的一个实时大数据处理框架。Marz在Twitter工作期间开发了著名的实时大数据处理框架Storm，Lambda架构是其根据多年进行分布式大数据系统的经验总结提炼而成。Lambda架构的目标是设计出一个能满足实时大数据系统关键特性的架构，包括有：高容错、低延时和可扩展等。Lambda架构整合离线计算和实时计算，融合不可变性（Immunability），读写分离和复杂性隔离等一系列架构原则，可集成Hadoop，Kafka，Storm，Spark，Hbase等各类大数据组件。

如果抛开上面的Merge 操作，那么Lambda架构就是两条完全不同处理流程，就像下面所示



存在的问题

同样的需求需要**开发两套一样的代码**，这是Lambda架构最大的问题，两套代码不仅仅意味着开发困难（同样的需求，一个在批处理引擎上实现，一个在流处理引擎上实现，还要分别构造数据测试保证两者结果一致），后期维护更加困难，比如需求变更后需要分别更改两套代码，独立测试结果，且两个作业需要同步上线。

资源占用增多：**同样的逻辑计算两次，整体资源占用会增多（多出实时计算这部分）·**

实时链路和离线链路计算结果容易让人误解，昨天看到的数据和今天看到的数据不一致**

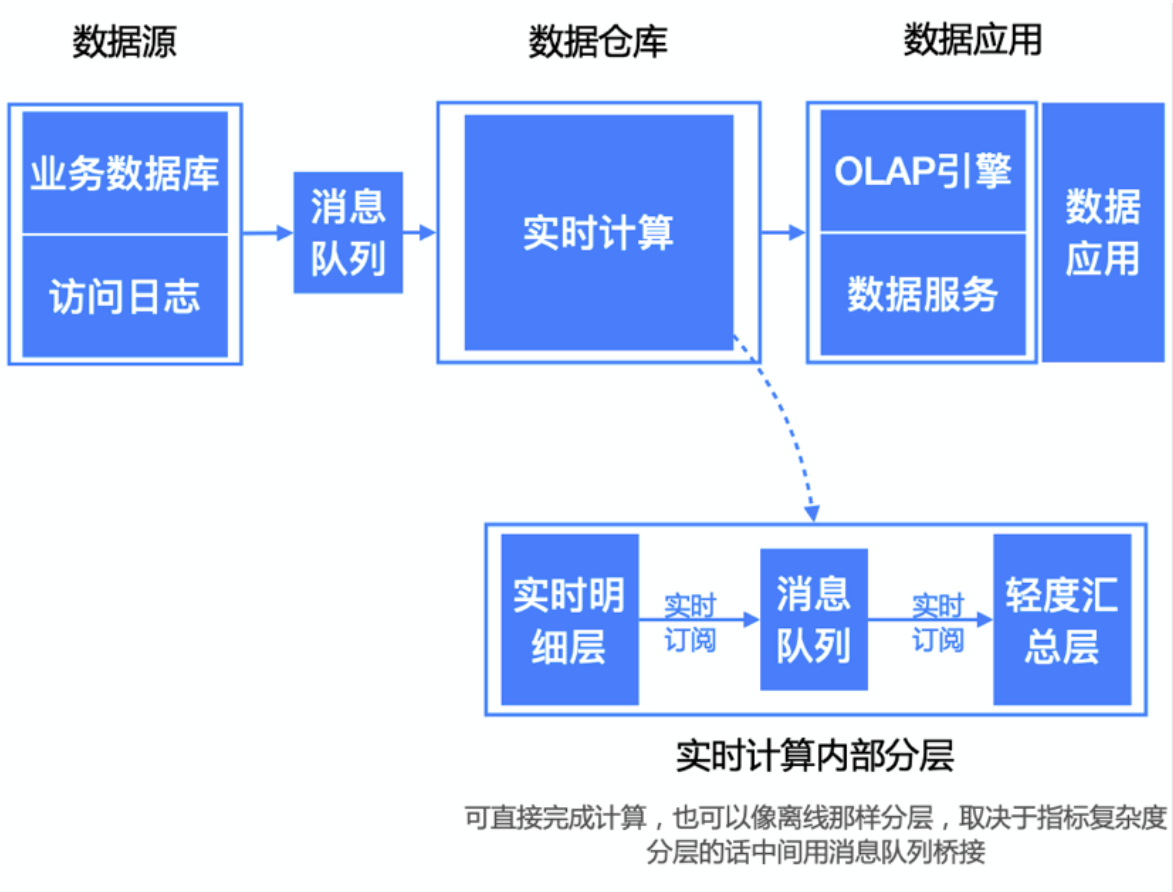
下游处理复杂，**需要整合实时和离线处理结果**，这一部分往往是我们在呈现给用户之前就完成了的

Kappa架构

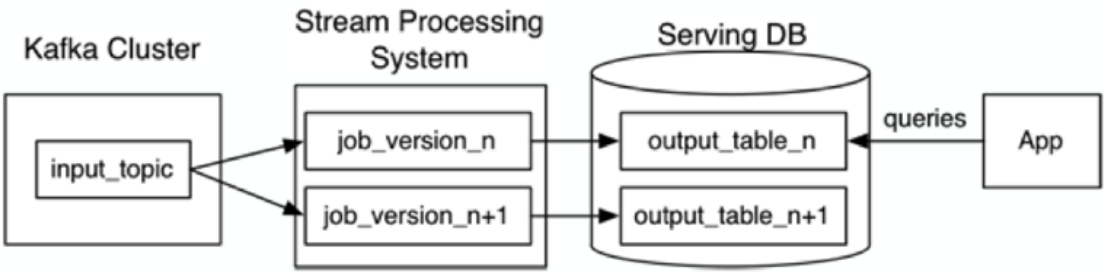
再后来，实时的业务越来越多，事件化的数据源也越来越多，实时处理从次要部分变成了主要部分，架构也做了相应调整，出现了以实时事件处理为核心的Kappa架构。当然这不要实现这一变化，还需要技术本身的革新——Flink，Flink 的出现使得Exactly-Once 和状态计算成为可能，这个时候实时计算的结果保证最终结果的准确性

Lambda架构虽然满足了实时的需求，但带来了更多的开发与运维工作，**其架构背景是流处理引擎还不完善，流处理的结果只作为临时的、近似的值提供参考。**后来随着Flink等流处理引擎的出现，流处理技术很成熟了，这时为了解决两套代码的问题，LickedIn 的Jay Kreps提出了Kappa架构

Kappa架构可以认为是Lambda架构的简化版（只要移除lambda架构中的批处理部分即可）。在Kappa架构中，需求修改或历史数据重新处理都通过上游重放完成。



Kappa架构的重新处理过程



选择一个具有重放功能的、能够保存历史数据并支持多消费者的消息队列，根据需求设置历史数据保存的时长，比如Kafka，可以保存全部历史数据,当然还有后面出现的Pulsar，以及专门解决实时输出存储的Pravega

当某个或某些指标有重新处理的需求时，按照新逻辑写一个新作业，然后从上游消息队列的最开始重新消费，把结果写到一个新的下游表中。

当新作业赶上进度后，应用切换数据源，使用新产生的新结果表。停止老的作业，删除老的结果表。

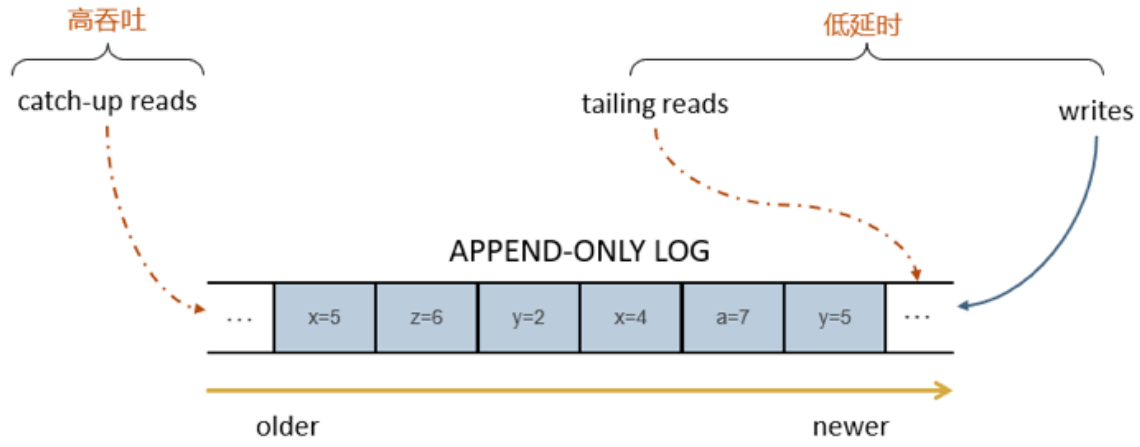
存在的问题

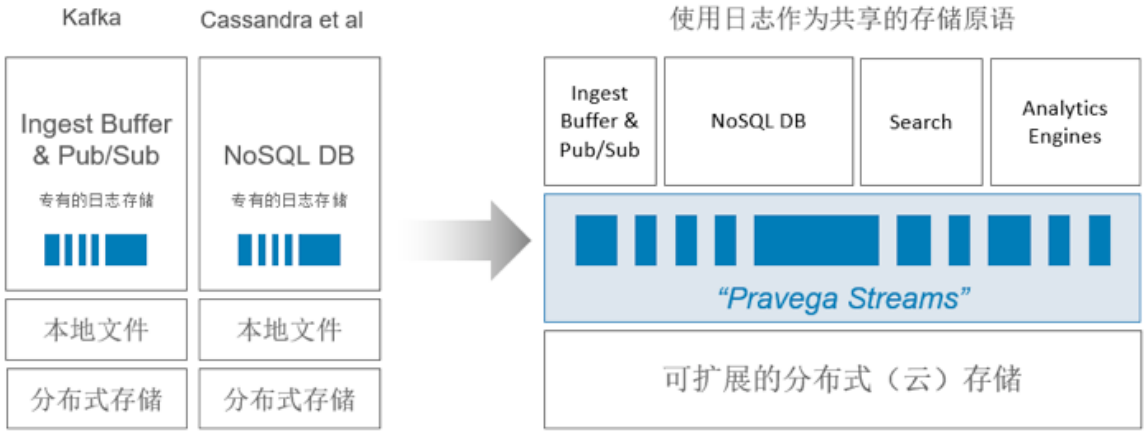
Kappa架构最大的问题是流式重新处理历史的吞吐能力会低于批处理，但这个可以通过增加计算资源来弥补

Pravega(流式存储)

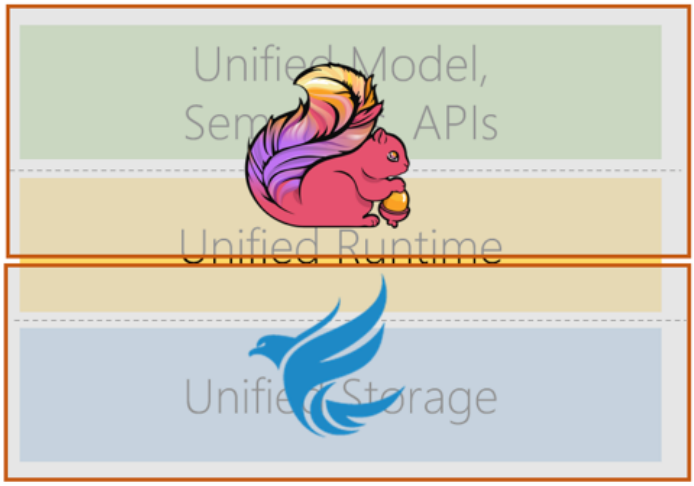
想要统一流批处理的大数据处理架构，其实对存储有混合的要求

> 对于来自序列旧部分的历史数据，需要提供高吞吐的读性能，即catch-up read对于来自序列新部分的实时数据，需要提供低延迟的 append-only 尾写 tailing write 以及尾读 tailing read





存储架构最底层是基于可扩展分布式云存储，中间层表示日志数据存储为 Stream 来作为共享的存储原语，然后基于 Stream 可以向上提供不同功能的操作:如消息队列，NoSQL，流式数据的全文搜索以及结合 Flink 来做实时和批分析。换句话说，Pravega 提供的 Stream 原语可以避免现有大数据架构中原始数据在多个开源存储搜索产品中移动而产生的数据冗余现象，其在存储层就完成了统一的数据湖。



统一的模型/API
处理实时和历史数据

能够同时应对高吞吐的
历史数据和低延时的实时数据

对历史和实时数据
有一致的访问方式

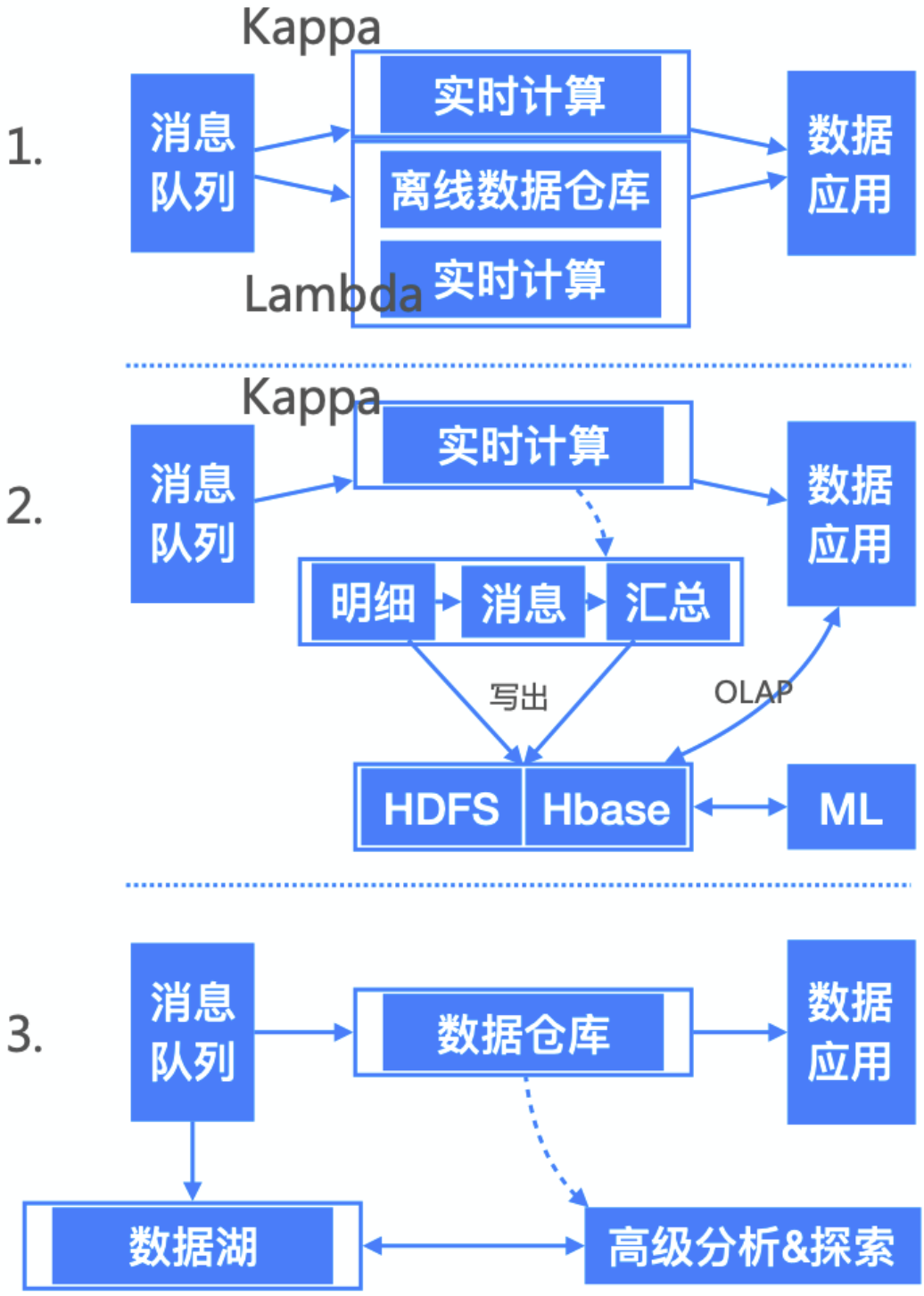
提出的大数据架构，以 Apache Flink 作为计算引擎，通过统一的模型/API来统一批处理和流处理。以 Pavega 作为存储引擎，为流式数据存储提供统一的抽象，**使得对历史和实时数据有一致的访问方式。**两者统一形成了从存储到计算的闭环，能够同时应对高吞吐的历史数据和低延时的实时数据。同时 Pravega 团队还开发了 Flink-Pravega Connector，为计算和存储的整套流水线提供 Exactly-Once 的语义。

混合架构

前面介绍了Lambda架构与Kappa架构的含义及优缺点，在真实的场景中，很多时候并不是完全规范的Lambda架构或Kappa架构，可以是两者的混合，比如大部分实时指标使用Kappa架构完成计算，少量关键指标（比如金额相关）**使用Lambda架构用批处理重新计算，增加一次校对过程。**

Kappa架构并不是中间结果完全不落地，现在很多大数据系统都需要支持机器学习（离线训练），所以实时中间结果需要落地对应的存储引擎供机器学习使用，另外有时候还需要对明细数据查询，这种场景也需要把实时明细层写出到对应的引擎中。

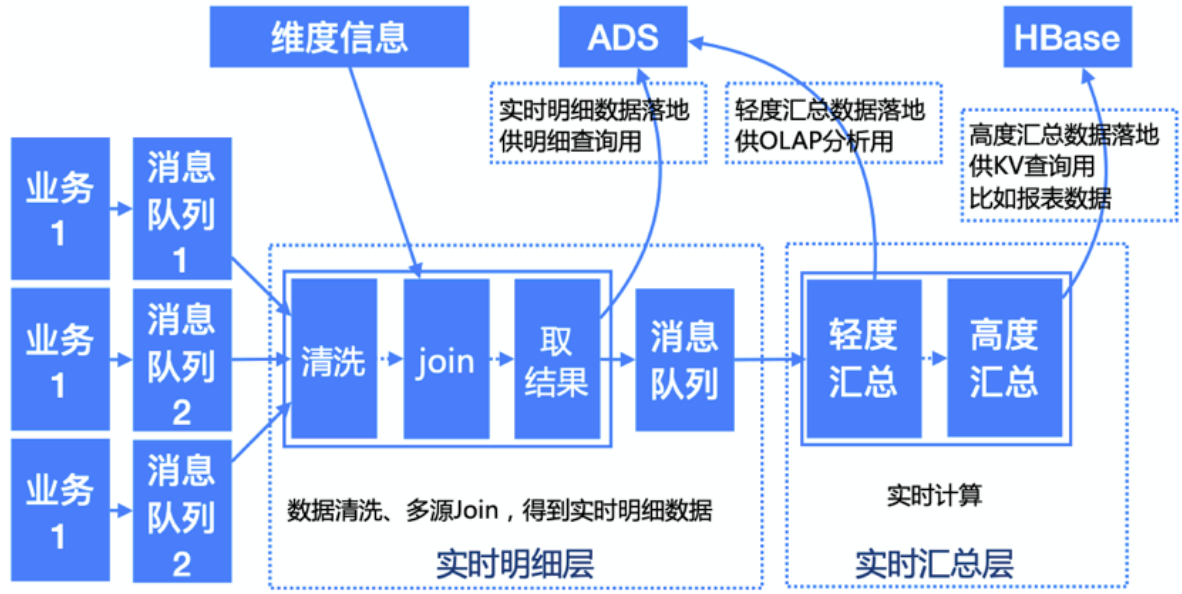
还有就是Kappa这种以实时为主的架构设计，除了增加了计算难度，对资源提出了更改的要求之外，还增加了开发的难度，所以才有了下面的混合架构，可以看出这个架构的出现，完全是处于需求和处于现状考虑的



实时数仓

实时数仓不应该成为一种架构，只能说是是Kappa架构的一种实现方式，或者说是实时数仓是它的一种在工业界落地的实现，在Kappa架构的理论支持下，实时数仓主要解决数仓对数据实时化的需求，例如数据的实时摄取、实时处理、实时计算等

其实实时数仓主主要解决三个问题 1. 数据实时性 2. 缓解集群压力 3. 缓解业务库压力。



第一层DWD公共实时明细层 实时计算订阅业务数据消息队列，然后通过数据清洗、多数据源join、流式数据与离线维度信息等的组合，将一些相同粒度的业务系统、维表中的维度属性全部关联到一起，增加数据易用性和复用性，得到最终的实时明细数据。这部分数据有两个分支，一部分直接落地到ADS，供实时明细查询使用，一部分再发送到消息队列中，供下层计算使用

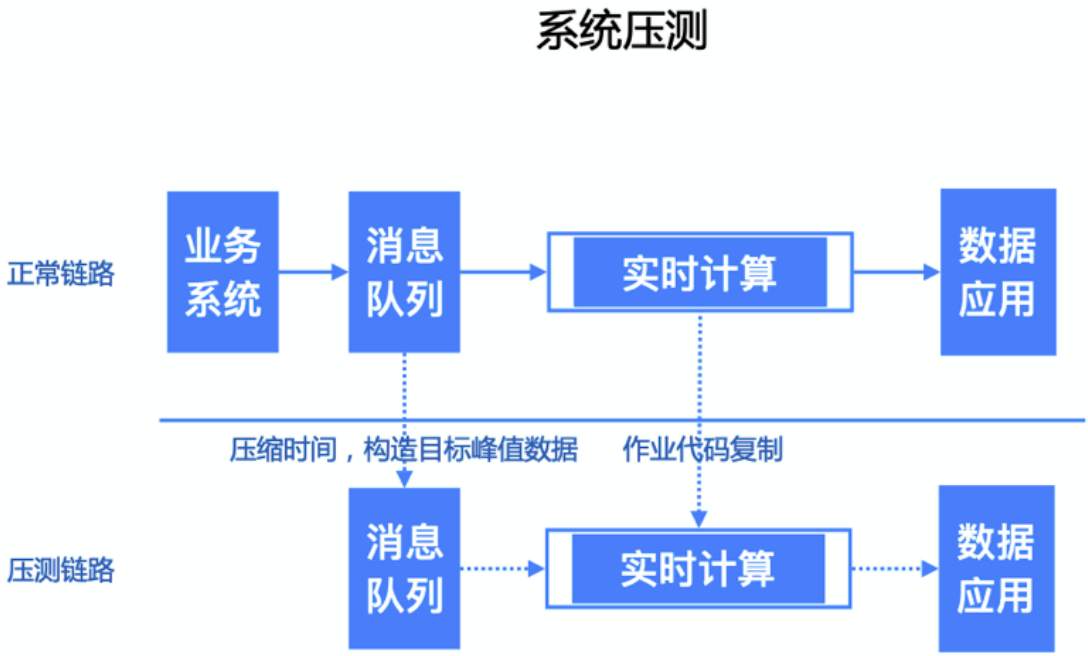
第二层DWS公共实时汇总层 以数据域+业务域的理念建设公共汇总层，与离线数仓不同的是，这里汇总层分为轻度汇总层和高度汇总层，并同时产出，轻度汇总层写入ADS，用于前端产品复杂的olap查询场景，满足自助分析；高度汇总层写入Hbase，用于前端比较简单的kv查询场景，提升查询性能，比如产出报表等

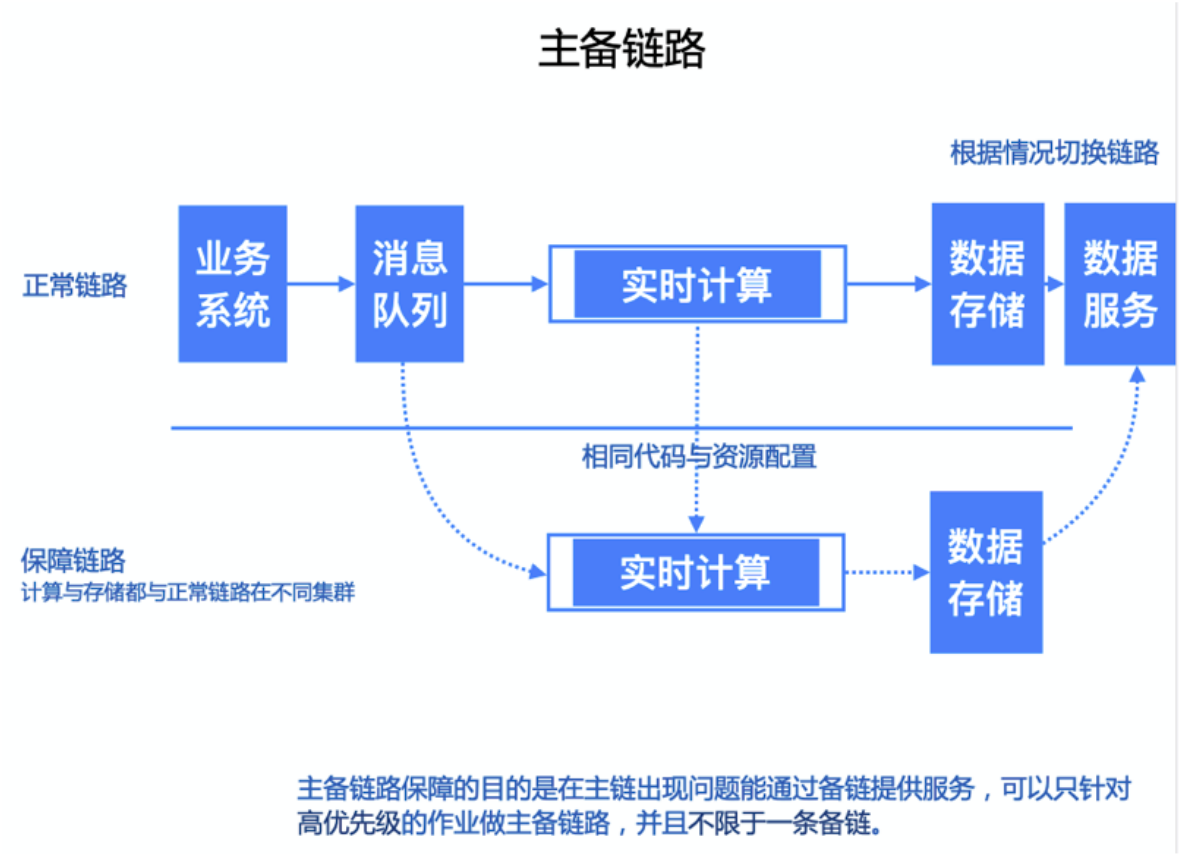
实时数仓的的实施关键点

1. 端到端数据延迟、数据流量量的监控
2. 故障的快速恢复能力力力 数据的回溯处理理，系统支支持消费指定时间端内的数据
3. 实时数据从实时数仓中查询，T+1数据借助离线通道修正
4. 数据地图、数据血血缘关系的梳理理
5. 业务数据质量量的实时监控，初期可以根据规则的方方式来识别质量量状况

数据保障

- 集团每年都有双十一等大促，大促期间流量与数据量都会暴增。实时系统要保证实时性，相对离线系统对数据量要更敏感，对稳定性要求更高
- 所以为了应对这种场景，还需要在这种场景下做两种准备： 1.大促前的系统压测； 2.大促中的主备链路保障





数据湖

最开始的时候，每个应用程序会产生、存储大量数据，而这些数据并不能被其他应用程序使用，这种状况导致数据孤岛的产生。随后数据集市应运而生，应用程序产生的数据存储在一个集中式的数据仓库中，可根据需要导出相关数据传输给企业内需要该数据的部门或个人，**然而数据集市只解决了部分问题。**剩余问题，包括数据管理、数据所有权与访问控制等都亟须解决，因为企业寻求获得更高的使用有效数据的能力。

为了解决前面提及的各种问题，**企业有很强烈的诉求搭建自己的数据湖，**数据湖不但能存储传统类型数据，也能存储任意其他类型数据(文本、图像、视频、音频)，并且能在它们之上做进一步的处理与分析，产生最终输出供各类程序消费。而且随着数据多样性的发展，**数据仓库这种提前规定schema的模式显得越来难以支持灵活的探索&分析需求，这时候便出现了一种数据湖技术，即把原始数据全部缓存到某个大数据存储上，后续分析时再根据需求去解析原始数据。**简单的说，**数据仓库模式是schema on write，数据湖模式是schema on read**

总结

Kappa对比Lambda架构

对比项	Lambda架构	Kappa架构
实时性	实时	实时
计算资源	批和流同时运行，资源开销大	只有流处理，仅针对新需求开发阶段运行两个作业，资源开销小
重新计算时吞吐	批式全量处理，吞吐较高	流式全量处理，吞吐较批处理低
开发、测试	每个需求都需要两套不同代码，开发、测试、上线难度较大	只需实现一套代码，开发、测试、上线难度相对较小
运维成本	维护两套系统（引擎），运维成本大	只需维护一套系统（引擎），运维成本小

在真实的场景中，很多时候**并不是完全规范的Lambda架构或Kappa架构**，可以是两者的混合，比如大部分实时指标使用Kappa架构完成计算，**少量关键指标（比如金额相关）使用Lambda架构用批处理重新计算，增加一次校对过程。**

这两个架构都是实时架构，都是对离线架构的扩展

实时数仓与离线数仓的对比

离线数据仓库主要基于sqoop、hive等技术来构建T+1的离线数据，通过定时任务每天拉取增量量数据导入到hive表中，然后创建各个业务相关的主题维度数据，对外提供T+1的数据查询接口

实时数仓当前主要是基于实时数据采集工具，如canal等将原始数据写入入到Kafka这样的数据通道中，最后一一般都是写 入到类似于HBase这样存储系统中，对外提供分钟级别、甚至至秒级别的查询方方案。