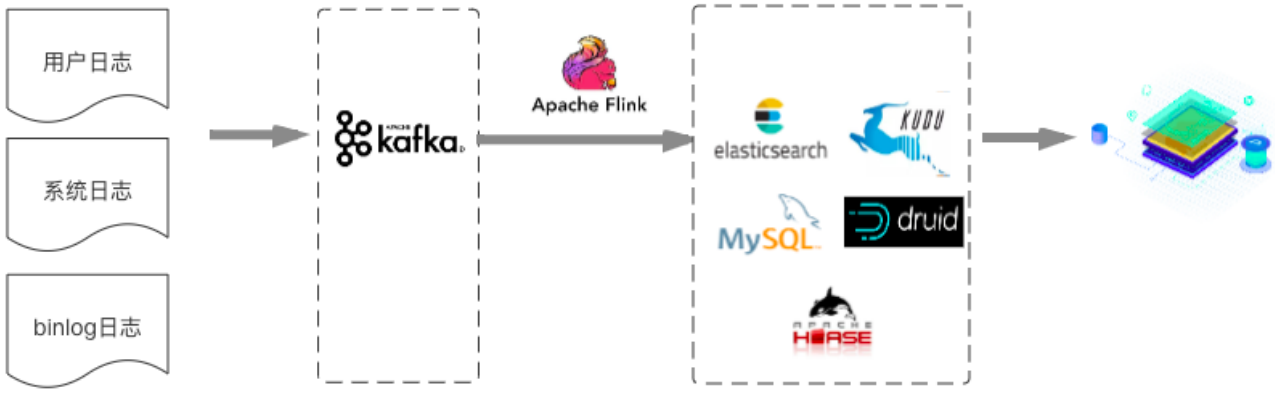
一文理解实时数据仓库的演进

数据处理现状：当前基于Hive的离线数据仓库已经非常成熟，数据中台体系也基本上是围绕离线数仓进行建设。但是随着实时计算引擎的不断发展以及业务对于实时报表的产出需求不断膨胀，业界最近几年就一直聚焦并探索于两个相关的热点问题：实时数仓建设和大数据架构的批流一体建设。

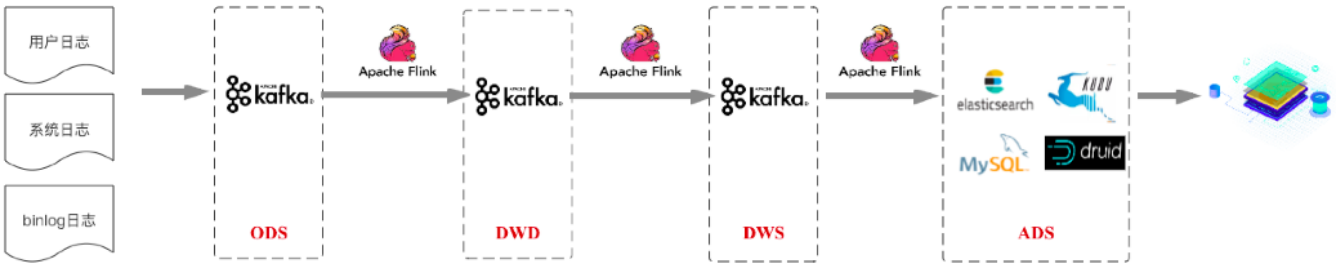
实时数仓建设：实时数仓1.0

传统意义上我们通常将数据处理分为离线数据处理和实时数据处理。对于实时处理场景，我们一般又可以分为两类，一类诸如监控报警类、大屏展示类场景要求秒级甚至毫秒级；另一类诸如大部分实时报表的需求通常没有非常高的时效性要求，一般分钟级别，比如10分钟甚至30分钟以内都可以接受。

对于第一类实时数据场景来说，业界通常的做法比较简单粗暴，一般也不需要非常仔细地进行数据分层，数据直接通过Flink计算或者聚合之后将结果写入MySQL/ES/HBASE/Druid/Kudu等，直接提供应用查询或者多维分析。如下所示：

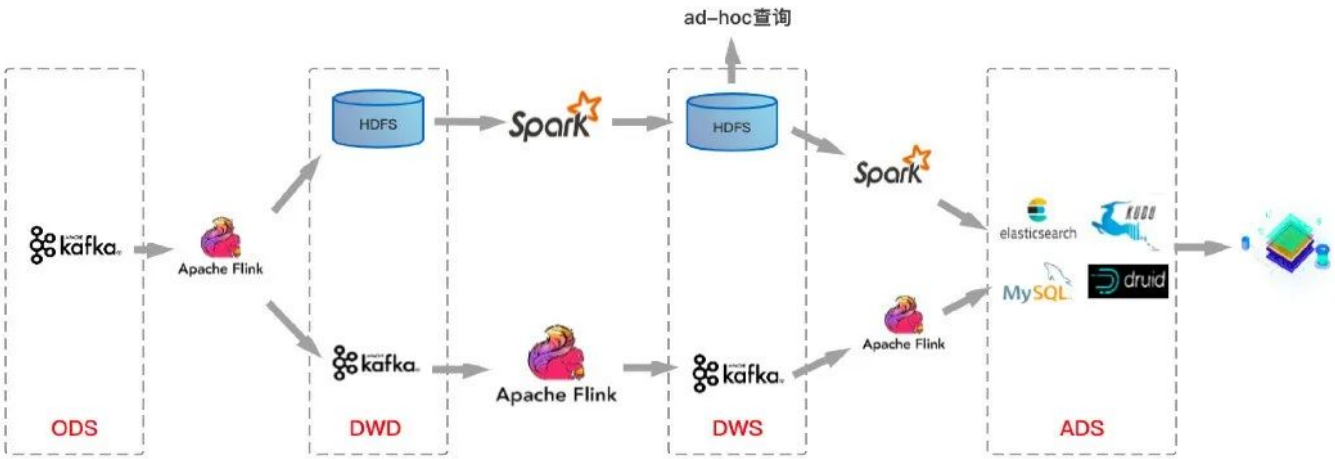


而对于后者来
说，通常做法会按照数仓结
构进行设计，我们称后者这种应用场景为实时数仓
，将作为本篇文章讨论的重点。从业界情况来看，当前主流的实时数仓架构基本都是基于Kafka+Flink的架构（为了行文方便，就称为实时数仓1.0）。下图是基于业界各大公司分享的实时数仓架构抽象的一个方案：



这套架构总体依然遵循标准的数仓分层结构，各种数据首先汇聚于ODS数据接入层。再接着经过这些来源明细数据的数据清洗、过滤等操作，完成多来源同类明细数据的融合，形成面向业务主题的DWD数据明细层。在此基础上进行轻度的汇总操作，形成一定程度上方便查询的DWS轻度汇总层（注：这里没有画出DIM维度层，一般选型为Redis/HBase，下文架构图中同样没有画出DIM维度层，在此说明）。最后再面向业务需求，在DWS层基础上进一步对数据进行组织进入ADS数据应用层，业务在数据应用层的基础上支持用户画像、用户报表等业务场景。

基于Kafka+Flink的这套架构方案很好的解决了实时数仓对于时效性的业务诉求，通常延迟可以做到秒级甚至更短。基于上图所示实时数仓架构方案，笔者整理了一个目前业界比较主流的整体数仓架构方案：



上图中上层链路是离线数仓数据流转链路，下层链路是实时数仓数据流转链路，当然实际情况可能是很多公司在实时数仓建设中并没有严格按照数仓分层结构进行分层，与上图稍有不同。

然而基于Kafka+Flink的实时数仓方案有几个非常明显的缺陷：

（1）Kafka无法支持海量数据存储。对于海量数据量的业务线来说，Kafka一般只能存储非常短时间的数据，比如最近一周，甚至最近一天；

（2）Kafka无法支持高效的OLAP查询。大多数业务都希望能在DWD₩DWS层支持即席查询的，但是Kafka无法非常友好地支持这样的需求；

（3）无法复用目前已经非常成熟的基于离线数仓的数据血缘、数据质量管理体系。需要重新实现一套数据血缘、数据质量管理体系；

（4）Lambad架构维护成本很高。很显然，这种架构下数据存在两份、schema不统一、数据处理逻辑不统一，整个数仓系统维护成本很高；

（5）Kafka不支持update/upsert。目前Kafka仅支持append。实际场景中在DWS轻度汇聚层很多时候是需要更新的，DWD明细层到DWS轻度汇聚层一般会根据时间粒度以及维度进行一定的聚合，用于减少数据量，提升查询性能。假如原始数据是秒级数据，聚合窗口是1分钟，那就有可能产生某些延迟的数据经过时间窗口聚合之后需要更新之前数据的需求。这部分更新需求无法使用Kafka实现。

所以实时数仓发展到现在的架构，一定程度上解决了数据报表时效性问题，但是这样的架构依然存在不少问题，随着技术的发展，相信基于Kafka+Flink的实时数仓架构也会进一步往前发展。那会往哪里发展呢？

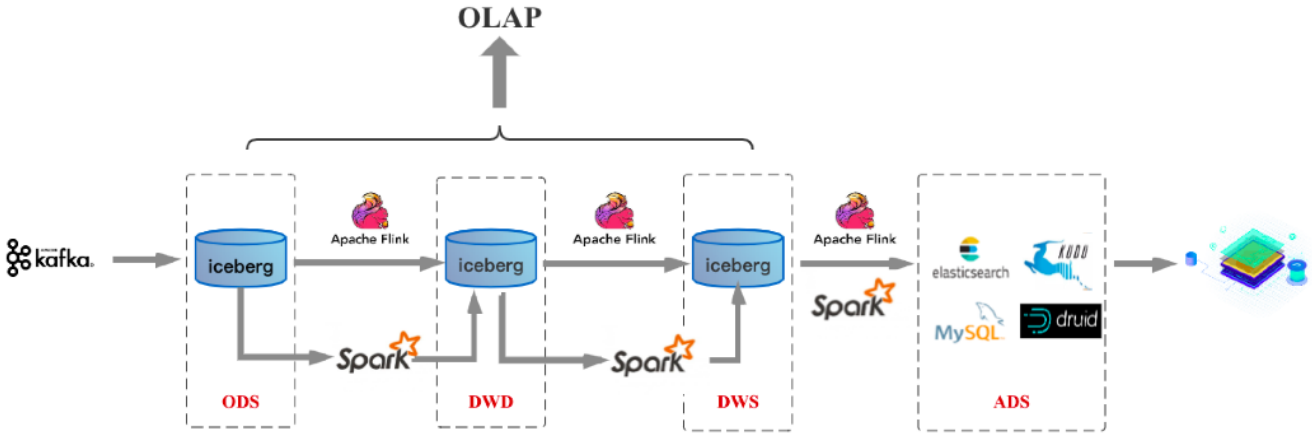
大数据架构的批流一体建设。

带着上面的问题我们再来接着聊一聊最近一两年和实时数仓一样很火的另一个概念：批流一体。对于批流一体的理解，笔者发现有很多种解读，比如有些业界前辈认为批和流在开发层面上都统一到相同的SQL上是批流一体，又有些前辈认为在计算引擎层面上批和流可以集成在同一个计算引擎是批流一体，比如Spark/Spark Structured Streaming就算一个在计算引擎层面实现了批流一

体的计算框架，与此同时另一个计算引擎Flink，目前在流处理方面已经做了很多的工作而且在业界得到了普遍的认可，但在批处理方面还有一定的路要走。

实时数仓2.0

笔者认为无论是业务SQL使用上的统一还是计算引擎上的统一，都是批流一体的一个方面。除此之外，批流一体还有一个最核心的方面，那就是存储层面上的统一。在这个方面业界也有一些走在前面的技术，比如最近一段时间开始流行起来的数据湖三剑客-- delta/hudi/iceberg，就在往这个方向走。存储一旦能够做到统一，上述数据仓库架构就会变成如下模样（以Iceberg数据湖作为统一存储为例），称为实时数仓2.0：



这套架构中无论是流处理还是批处理，数据存储都统一到数据湖Iceberg上。那这么一套架构将存储统一后有什么好处呢？很明显，可以解决Kafka+Flink架构实时数仓存在的前面4个问题：

（1）可以解决Kafka存储数据量少的问题。目前所有数据湖基本思路都是基于HDFS之上实现的一个文件管理系统，所以数据体量可以很大。

（2）DW层数据依然可以支持OLAP查询。同样数据湖基于HDFS之上实现，只需要当前的OLAP查询引擎做一些适配就可以进行OLAP查询。

（3）批流存储都基于Iceberg/HDFS存储之后，就完全可以复用一套相同的数据血缘、数据质量管理体系。

（4）Kappa架构相比Lambad架构来说，schema统一，数据处理逻辑统一，用户不再需要维护两份数据。

有的同学说了，这不，你直接解决了前4个问题嘛，还有第5个问题呢？对，第5个问题下文会讲到。

又有的同学会说了，上述架构确实解决了Lambad架构的诸多问题，但是这套架构看起来就像是一条离线处理链路，它是怎么做到报表实时产出呢？确实，上述架构图主要将离线处理链路上的HDFS换成了数据湖Iceberg，就号称可以实现实时数仓，听起来容易让人迷糊。这里的关键就是

数据湖Iceberg，它到底有什么魔力？

为了回答这个问题，笔者就上述架构以及数据湖技术本身做一个简单的介绍（接下来也会基于Iceberg出一个专题深入介绍数据湖技术）。上述架构图中有两条数据处理链路，一条是基于Flink的实时数据链路，一条是基于Spark的离线数据链路。通常数据都是直接走实时链路处理，而离线链路则更多的应用于数据修正等非常规场景。这样的架构要成为一个可以落地的实时数仓方案，数据湖Iceberg是需要满足如下几个要求的：

（1）支持流式写入-增量拉取
。流式写入其实现在基于Flink就可以实现，无非是将checkpoint间隔设置的短一点，比如1分钟，就意味每分钟生成的文件就可以写入到HDFS，这就是流式写入。没错，但是这里有两个问题，第一个问题是小文件很多，但这不是最关键的，第二个问题才是最致命的，就是上游每分钟提交了很多文件到HDFS上，下游消费的Flink是不知道哪些文件是最新提交的，因此下游Flink就不知道应该去消费处理哪些文件。这个问题才是离线数仓做不到实时的最关键原因之一，离线数仓的玩法是说上游将数据全部导入完成了，告诉下游说这波数据全部导完了，你可以消费处理了，这样的话就做不到实时处理。

数据湖就解决了这个问题。实时数据链路处理的时候上游Flink写入的文件进来之后，下游就可以将数据文件一致性地读走。这里强调一致性地读，就是不能多读一个文件也不能少读一个文件。上游这段时间写了多少文件，下游就要读走多少文件。我们称这样的读取叫增量拉取。

（2）解决小文件多的问题。
数据湖实现了相关合并小文件的接口，Spark/Flink上层引擎可以周期性地调用接口进行小文件合并。

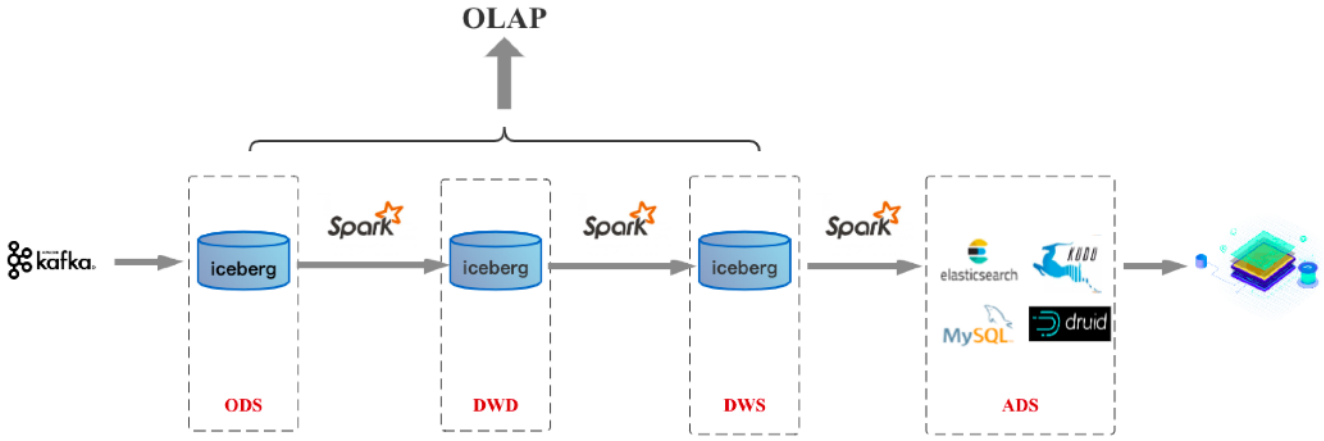
（3）支持批量以及流式的Upsert(Delete)功能。
批量Upsert/Delete功能主要用于离线数据修正。流式upsert场景上文介绍了，主要是流处理场景下经过窗口时间聚合之后有延迟数据到来的话会有更新的需求。这类需求是需要一个可以支持更新的存储系统的，而离线数仓做更新的话需要全量数据覆盖，这也是离线数仓做不到实时的关键原因之一，数据湖是需要解决掉这个问题的。

（4）支持比较完整的OLAP生态。
比如支持Hive/Spark/Presto/Impala等OLAP查询引擎，提供高效的多维聚合查询性能。

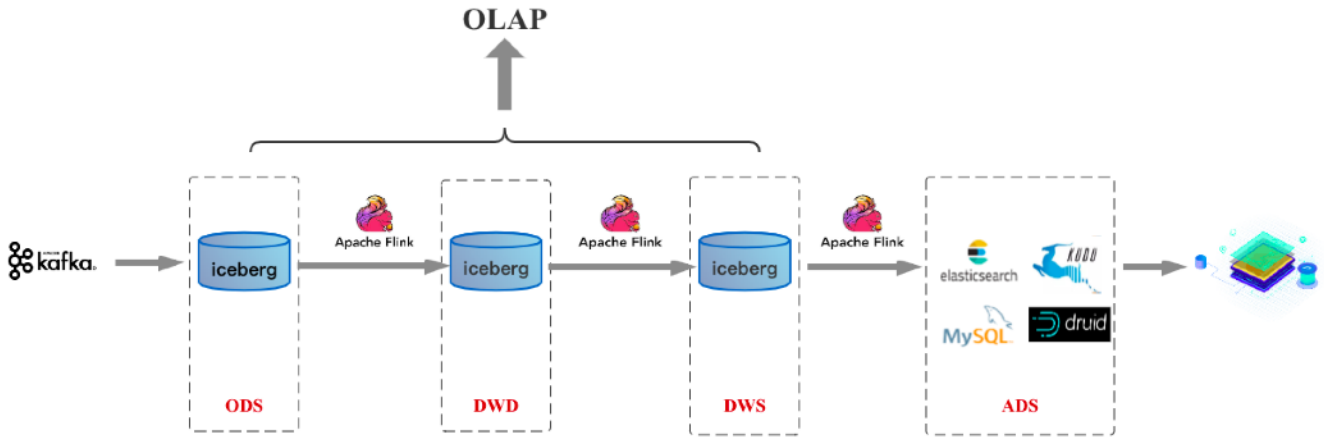
这里需要备注一点，目前Iceberg部分功能还在开发中。具体技术层面Iceberg是怎么解决上述问题的，请持续关注本号，接下来一篇文章会详细讲解哦。

实时数仓3.0

按照批流一体上面的探讨，如果计算引擎做到了批流一体的统一，就可以做到SQL统一、计算统一以及存储统一，这时就迈入实时数仓3.0时代。对于以Spark为核心技术栈的公司来说，实时数仓2.0的到来就意味着3.0的到来，因为在计算引擎层面Spark早已做到批流一体。基于Spark/数据湖的3.0架构如下图：



假如未来Flink在批处理领域成熟到一定程度，基于Flink/数据湖的3.0架构如下图：



上面所介绍的，是笔者认为接下来几年数据仓库发展的一个可能路径。对于业界目前实时数仓的一个发展预估，个人觉得目前业界大多公司都还往实时数仓1.0这个架构上靠；而在接下来1到2年时间随着数据湖技术的成熟，实时数仓2.0架构会成为越来越多公司的选择，其实到了2.0时代之后，业务同学最关心的报表实时性诉求和大数据平台同学最关心的数据存储一份诉求都可以解决；随着计算引擎的成熟，实时数仓3.0可能和实时数仓2.0一起或者略微滞后一些普及。

作者简介

子和，网易大数据开发工程师，长期从事分布式KV数据库、分布式时序数据库以及大数据底层组件等相关工作。