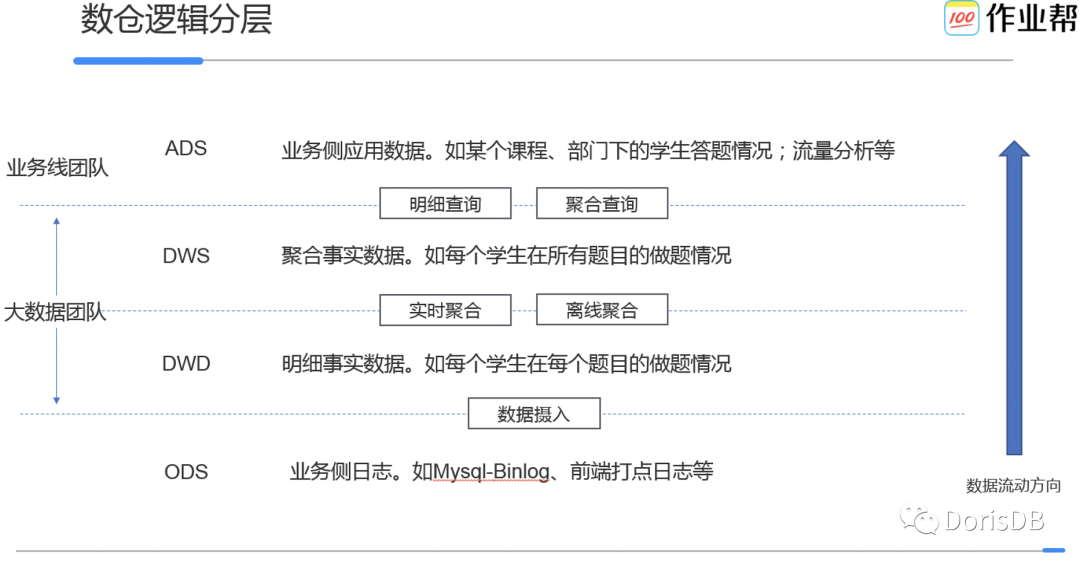
**基于 Apache Doris 数仓在作业帮的实践**

***1 背景***

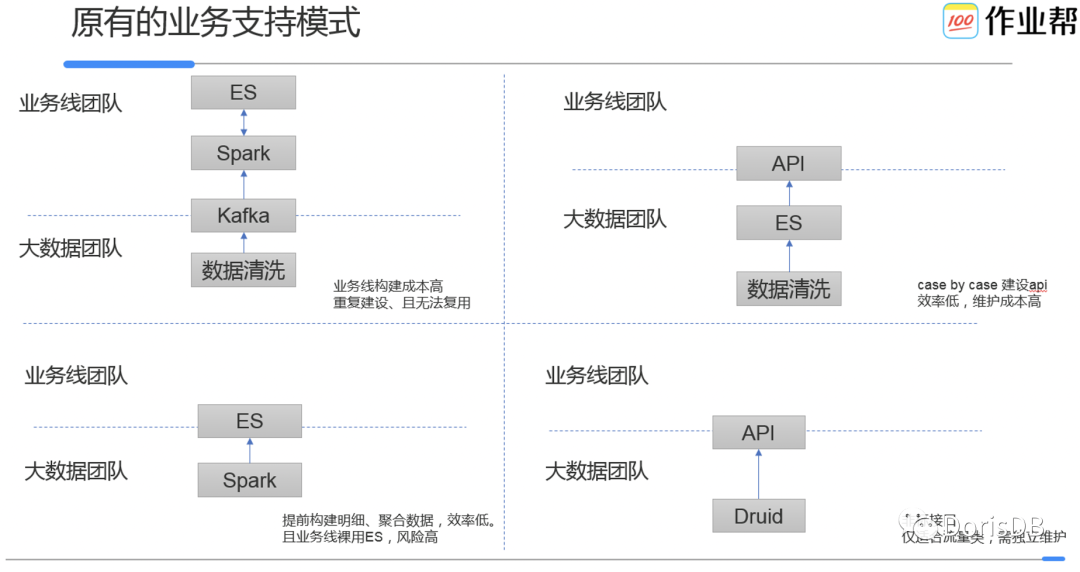
作业帮大数据团队主要负责建设公司级数仓，向公司各个重要产品线（拉新、教学、BI等）提供面向业务的数据信息，如到课时长、答题情况等。在过去半年多时间内，我们基于Apache Doris，构建了数仓实时查询系统。本文总结并分享下期间的工作内容，也欢迎大家一起讨论。

典型的数仓从逻辑上划分为下图所示的几个部分。



大数据团队主要负责到ODS-DWS的建设，从DWS到ADS一般是数仓系统和业务线系统的边界。

在过去，由于缺失统一的查询系统，我们探索了很多模式来支持各个业务线发展。



**非流量类**：

* Kafka 。业务线从Kafka接数据自己做数据的聚合计算。主要问题在于完全没有数仓的概念，业务线在做大量重复的建设。
* Spark + ES。每来一个业务需求，就构建一个Spark+ES集群（spark负责计算写入到ES，ES业务层直接使用）。效率低、构建成本高，且ES高效的使用本身就需要学习ES的接口以及内部原理，对于业务线很难有这样的精力去做。
* ES + 自定义API。大数据将数据写入ES后，case by case构建API。这样做初步建立了数仓的接口，但是由于接口不具备SQL的能力，只能基于需求case by case的构建，效率太低。

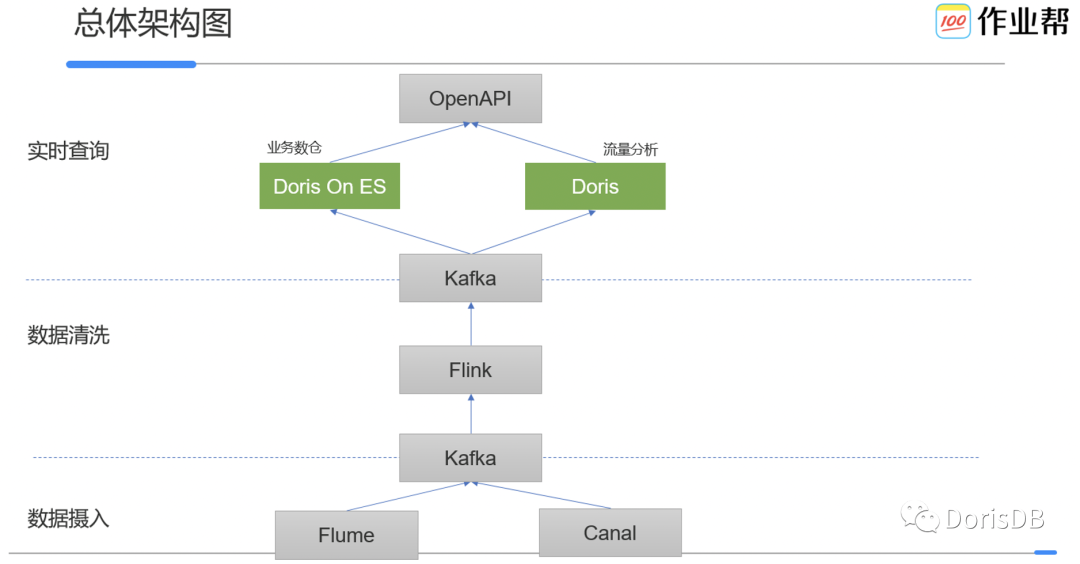
**流量类**：

* 由于数据量大，往往需要预聚合，所以我们引入了Druid。但是Druid只支持聚合数据，无法保留明细数据。导致Druid只能适合流量分析类的场景，对于工作台既需要明细（如查询某个老师的教学数据）也需要聚合（如查询某个部门下所有老师的教学数据）的场景无法满足。

这些“烟囱”式的系统构建方式，导致系统越来越难以维护，且业务接入效率也逐步降低。因此，统一整个查询引擎，对于数仓建设在提高业务支持效率、降低维护成本上都具有非常重大的意义。

***2 总体方案***

经过数月的探索与实践，我们确立了以Doris为基础的数仓实时查询系统，同时也对整个数仓的数据计算系统做了一次大的重构。最终整体的架构图如下：



如图所示（从下到上），原始业务层日志经数据摄入系统进入数仓，在数据清洗计算层，我们将原来的Spark系统升级到了Flink，并且基于Flink-SQL提供了统一数据开发框架，从原有的代码开发升级到Sql开发来极大的提升数据的研发效率。

其后查询系统将Kafka的数据实时同步到查询引擎内，并通过OpenAPI的统一接口对外提供查询服务。接下来，重点讲一下查询系统的工作。

***2.1 查询引擎选型***

实时查询系统的核心在于确定查询引擎。社区的查询引擎较多，如Impala、Presto、Doris、ES（xpack），以及云上的ADB等。这块考虑到调研成本、团队技术生态、维护成本等多种因素，我们最后选择了Doris 作为我们的查询引擎。



注意：上图各性能指标对应的业务查询不完全相同。

在性能调研时，我们也走了一些弯路：第一次使用Doris来做查询引擎，我们发现业务SQL查询延迟比较大，且CPU使用率很高（IDLE < 10%）。后来发现，原因在于我们使用了AGGREGATE模型。如，对于订单数据，我们一般会将用户支付金额等作为指标列（如一个用户从订单预订到支付，状态的改变会修改支付金额值），但是业务端的SQL中有大量的基于支付金额（指标列）的筛选查询，如统计支付金额 > 某个值的用户数。

Doris对于指标列的筛选成本较高。由于Doris底层采用了类LSM-Tree的结构，因此为了确定某一行的数据是否该被筛选，需要读取所有包含该行数据的底层文件，进而聚合计算后才可以决策，结果集是否包含该行（UNIQ模型类似）。

最终我们决定使用Doris on ES，主要考虑点如下：

* 支持任意列检索。基于ES的倒排索引，我们可以对任意列进行检索（筛选）。这个模型大大降低了业务同学的学习理解成本，可以和MySQL一样方便的构建数据模型。
* ES的易用性以及整个技术生态在我们公司内相对成熟，维护成本较低。如，数据修改可以直接覆盖最新值，非常简单。
* Doris on ES在数据Scan上做了大量的优化操作，如列存、local优先、响应内容过滤、顺序扫描、提前终止等，对于数据的扫描性能可以达到~30w/s
* Doris 提供了更强大的SQL语法（如join、多列group by……），且整个查询过程保障了数据的准确度。大大提高了数据使用的效率和数据查询质量。 由于ES缺少分布式计算层，导致ES-SQL需要配置size，否则会导致返回的数据会少于预期的数据。使用Doris on ES可以避免这个问题

当然，对于流量分析的场景，由于指标列一般是PV、UV等，业务上并没有对指标的筛选过滤需求，且Doris自身支持Rollup，因此非常适合流量类的查询分析。

因此，**通过Doris我们统一了整个查询引擎端的实现，这样对于后续整个数仓的进一步建设打下了非常重要的基础。**

***2.2 应用实践***

基于业务场景，我们对需求进行了分类：面向业务工作台的非流量类需求以及流量分析类需求。

对于我们的非流量类场景，在实际的应用中，业务侧的需求主要分两类：

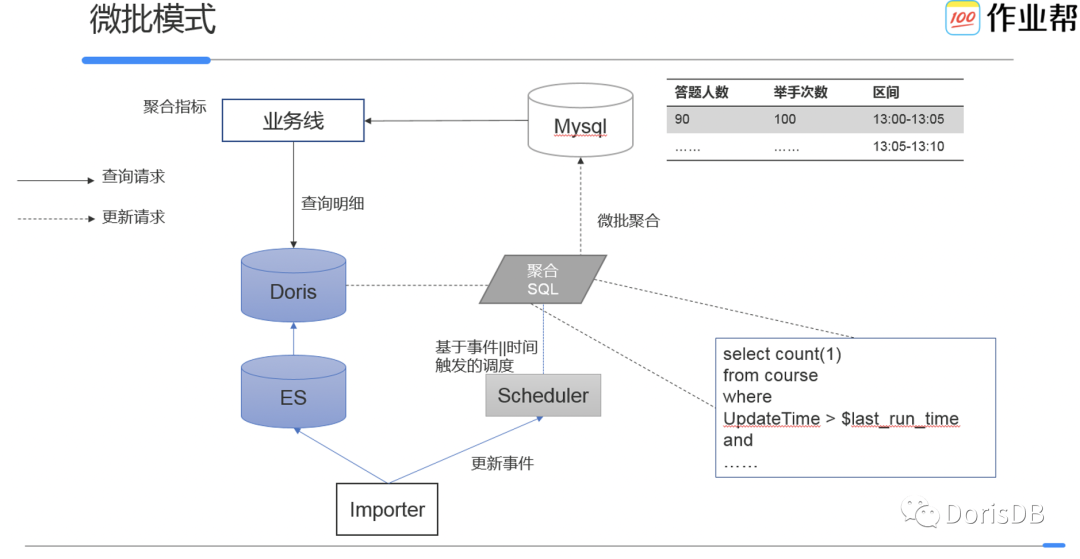
* 明细查询。教研工作台需要关注每个老师的明细信息，如某课程的学生的到课情况、课前预习情况……
* 聚合查询。部门组织上会关注整个部门、小组内的统计信息，如到课率、拉新率等

这些需求在前端查询，均需要保障低延迟。

其中明细查询对于数据的时效性要求更高，因此对于明细类查询，业务侧会直接访问Doris on ES中的数据进行查询，这样基于Doris on ES的任意列检索能力可以保障业务查询模式的灵活性以及数据的时鲜性。

而对于聚合查询，由于不同指标的Sql计算的数据范围不同，且业务侧对于聚合的计算没有明细查询的时效性，因此，我们通过微批（如1min、5mins、10mins……）的调度能力定期计算聚合指标，并存放到ADS层的业务数据库中供前端平台查询。

为了提高数据使用效率，方便业务侧获得特定时间窗口的数据，在数据模型上，我们统一设置了Meta字段如数据更新时间，这样业务可以用来划分每次更新的数据窗口，做增量计算。



这个模式的主要好处：

* 业务端延迟可控、稳定性好。聚合查询的延迟随着具体的SQL不同而不同，定期执行后的数据存放到业务层MySQL中，可以最大化可以保证查询延迟。
* 数据修复成本低、维护方便。一旦数据有异常，可以自动触发对应的数据窗口进行重新计算。原来基于流式计算的数据修复，需要从源头开始修复，且必须驱动主流事件触发，成本非常高。而基于Doris on ES，不同的事件可以更新不同的列或者表，只要在数据查询时join即可
* 高性能。一般业务每次读取部分列，这个模式反而可以发挥ES适合大宽表的场景以及Doris on ES列存读取模式的实现，更保障了这块的高性能。

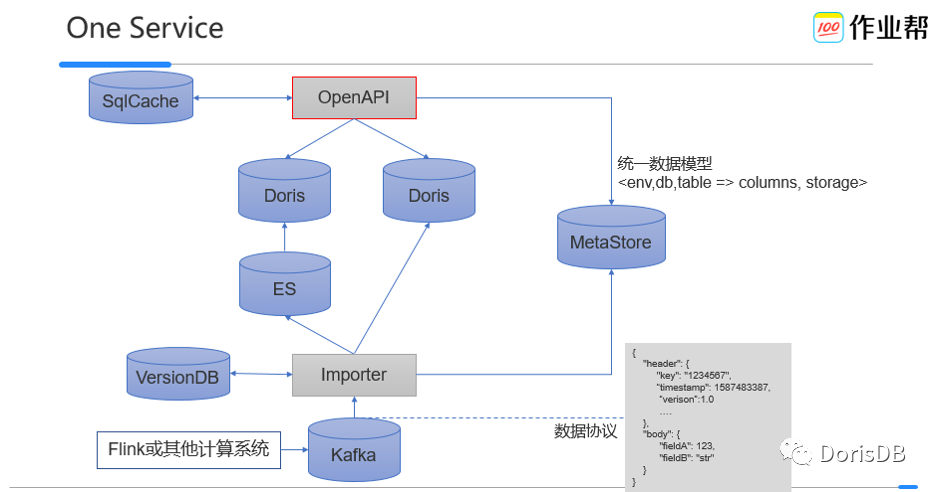
对于流量类场景，在数据清洗后，直接从Kafka导入Doris即可，这块主要是利用Doris Rollup的能力，提供低延迟的数据查询能力。

虽然上述方案可以初步满足业务的需求，但是从最终系统可持续运维的角度来看，还有很多潜在的问题需要解决。

* 如何保障查询稳定性
  + 多个用户同时进行SQL查询，某个查询有可能耗尽整个集群的资源，如何快速止损？
  + 多个场景都在查询某一张表，如何做到可控的降级？
* 如何保障入库的数据质量
  + 避免数据乱序覆盖……
  + 保障数据在多个库之间的无损、低成本迁移……如从Hive迁移到Doris、ES迁移到MySQL……
* 如何提高易用性
  + 数仓内支持SQL的系统很多，如Hive的HQL、Flink-SQL……在部分函数语法上会存在差异，如何透明地打平这些差异，而不是让用户不断的学习异构语法？
  + 数据如果跨云同步，提供多集群数据同步、查询切换，如何对业务透明的完成？
  + 部分表需要自动Rotate的能力，自动删除过期的数据。

上述的这些问题虽然短期内无法一一解决，但是需要提供一个能力：将来解决时成本可控，尽量做到对业务无感知。这些都需要进一步定义出系统的接口边界，否则耦合各个系统，后续使用的用户越多，问题持续时间越久、迁移成本也越高。

因此我们设计了OneModel来统一数据模型，并且构建了OpenAPI来统一服务接口。



目前完成的功能包括

* OpenAPI上
  + SQL缓存
  + 基于业务线的查询条件控制，如query\_timeout
* OneModel
  + 随着Flink系统的引入，结合原离线数仓的表，数据表在不同存储上分布越来越多如Kafka、Redis、Doris、Hive……，因此构建<数据表，Schema，存储>的元数据，支持数据表在不同存储上的映射关系，统一表逻辑视图，提升使用效率
  + 引入了Json-schema，保证入库质量符合数据模型定义
* 其他
  + Rotate Table
  + 规范化数据协议，基于数据版本解决数据写入时乱序问题

基于上述的设计，我们不仅支持了业务功能，更重要的是切分了整个系统的接口，降低了各个系统之间的耦合。有几点具体的好处:

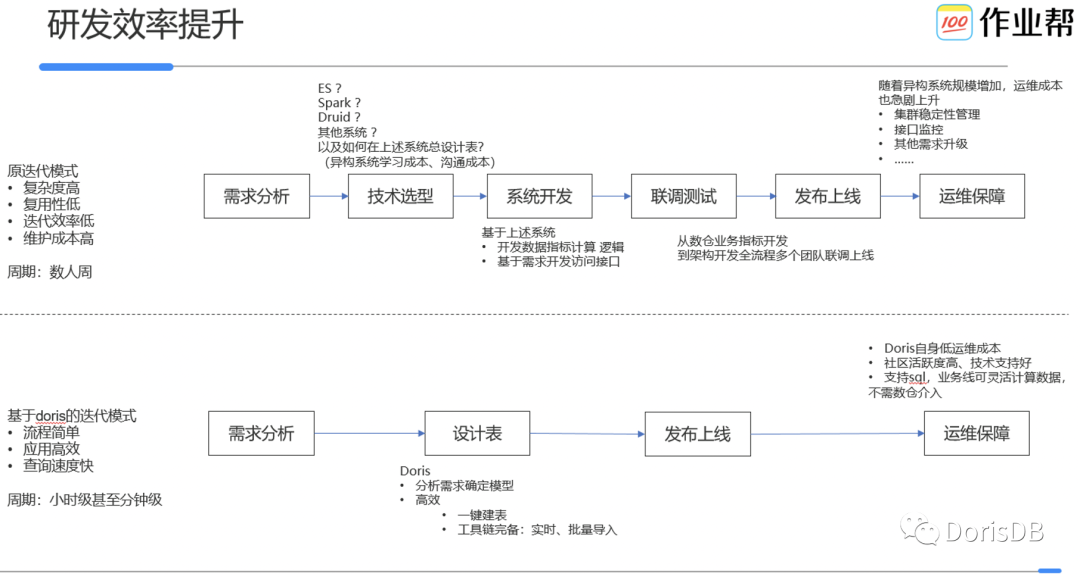
* 数据清洗系统和查询系统基于Kafka解耦，这样当查询系统临时异常时，不会阻塞计算系统。且多个Topic可以天然支持正常数据流&修数数据流的同步入库。
* 业务层通过统一的接口来进行数据访问，在访问入口处可以统一方便的进行流量调度，统一的解决稳定性问题
* 由于整个系统闭包，且接口基于数据协议耦合，稳定性和易用性得到了兼顾。

***3 应用表现***

基于Doris on ES的查询系统上线数月，一直到经历了运营大促的活动，均表现出了非常好的稳定性。每天百万级次调用，99分位延迟~秒级。

当前Doris集群在作业帮已经有5+个集群，~50+表、500GB+数据，用于支持核心业务线如售卖、广告、拍搜、业务数仓等。

**我们的人效也得到了了数十倍的提升**：从过去一个需求“进入查询系统到对外交付数据”需要数人周，提升到当前模式的小时级甚至分钟级。



***4 总结和规划***

通过引入Doris，我们解决了明细&聚合数据查询不统一的问题，奠定了整个数据中台在查询侧的基石。这对于后续数仓向数据中台发展的路径起到了非常关键的作用。未来我们对Doris在作业帮的应用有不少规划：

* 跨集群实时同步。在异地多活等场景下，目前缺少类似mysql-binlog的实时同步能力，需要构建低成本的数据实时同步能力，支持在线业务的稳定性。
* Doris on ES多表Join性能。在长尾的需求下，Join需要扫描两张表的全部数据进行内存计算，尤其是大表Join大表，延迟就会升高。
* Doris on ES表分区能力。如对于ES的Rotate表，目前Doris无法识别新表或者自动删除老的表映射，需要频繁创建Doris表来对应ES.Index。
* Doris on ES表自动同步能力。如ES表Schema修改后，可以自动同步到Doris。
* Doris平台化运维，如建表、修改表、数据导出……

更多Doris on ES的规划，可请参见：https://github.com/apache/incubator-doris/issues/3306

最后，非常感谢百度Doris团队和鼎石科技团队，特别是@wuyunfeng、@imay 等同学热情、给力、靠谱的技术支持！！！我们也希望后续一起参与到Doris的开发建设中来！

**作者：糜利敏，作业帮大数据工程师**