Apache Doris在美团外卖数仓中的应用实践

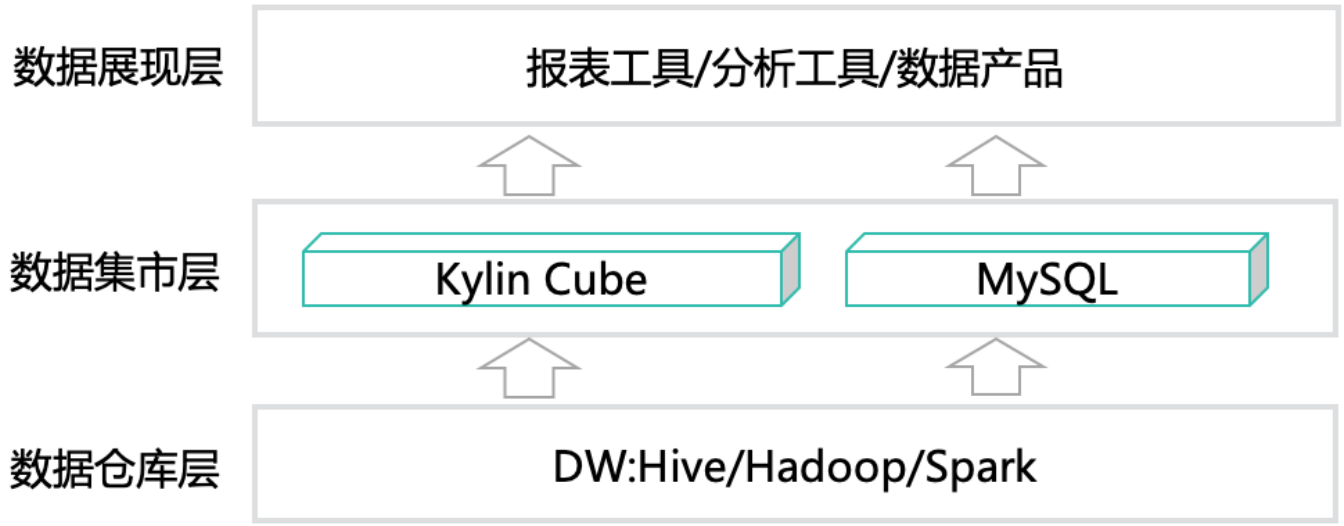
序言

美团外卖数据仓库技术团队负责支撑日常业务运营及分析师的日常分析，由于外卖业务特点带来的数据生产成本较高和查询效率偏低的问题，他们通过引入Apache Doris引擎优化生产方案，实现了低成本生产与高效查询的平衡。并以此分析不同业务场景下，基于Kylin的MOLAP模式与基于Doris引擎的ROLAP模式的适用性问题。希望能对大家有所启发或者帮助。

本文侧重于以Doris引擎为“发动机”的数仓生产架构的改进与思考。在开源的大环境下，各种数据引擎百花齐放，但由于业务的复杂性与多样性，目前并没有哪个引擎能够适配所有业务场景，因此希望通过我们的业务实践与思考为大家提供一些经验参考。美团外卖数仓技术团队致力于将数据应用效率最大化，同时兼顾研发、生产与运维成本的最小化，建设持续进步的数仓能力，也欢迎大家多给我们提出建议。

数仓交互层引擎的应用现状

目前，互联网业务规模变得越来越大，不论是业务生产系统还是日志系统，基本上都是基于Hadoop/Spark分布式大数据技术生态来构建数据仓库，然后对数据进行适当的分层、加工、管理。而在数据应用交互层面，由于时效性的要求，数据最终的展现查询还是需要通过DBMS（MySQL）、MOLAP（Kylin）引擎来进行支撑。如下图所示：



如果想及时了
解Spark、Hadoop或者Hbase相关的文章，欢迎关注微信公共帐号：iteblog_hadoop

汇总数据的交互

业务团队日常经营分析最典型的场景就是各种维度下的自定义查询，面对如此灵活可变、所见即

所得的应用场景，美团平台使用Kylin作为公司的主要MOLAP引擎。MOLAP是预计算生产，在增量业务，预设维度分析场景下表现良好，但在变化维的场景下生产成本巨大。例如，如果使用最新商家类型回溯商家近三个月的表现，需要重新计算三个月的Cube，需花费几个小时，来计算近TB的历史数据。另外，应对非预设维度分析，MOLAP模型需要重新进行适配计算，也需要一定的迭代工作。

明细数据的交互

业务分析除了宏观数据之外，对明细数据查询也是一种刚需。通常大家会选择MySQL等关系型DB作为明细数据的快速检索查询，但当业务成长较快时，很快就会遇到性能瓶颈，并且运维成本也很高。例如，大数据量的同步、新增字段、历史数据更新等操作，它们的维护成本都非常高。

外卖运营业务特点

美团的使命是“帮大家吃得更好，生活更好”。外卖业务为大家提供送餐服务，连接商家与用户，这是一个劳动密集型的业务，外卖业务有上万人的运营团队来服务全国几百万的商家，并以“商圈”为单元，服务于“商圈”内的商家。“商圈”是一个组织机构维度中的最小层级，源于外卖组织的特点，“商圈”及其上层组织机构是一个变化维度，当“商圈”边界发生变化时，就导致在往常日增量的业务生产方式中，历史数据的回溯失去了参考意义。在所有展现组织机构数据的业务场景中，组织机构的变化是一个绕不开的技术问题。此外，商家品类、类型等其它维度也存在变化维的问题。如下图所示：

数据生产面临的挑战

数据爆炸，每日使用最新维度对历史数据进行回溯计算。在Kylin的MOLAP模式下存在如下问题：

- 历史数据每日刷新，失去了增量的意义。
- 每日回溯历史数据量大，10亿+的历史数据回溯。
- 数据计算耗时3小时+，存储1TB+，消耗大量计算存储资源，同时严重影响SLA的稳定性。
- 预计算的大量历史数据实际使用率低下，实际工作中对历史的回溯80%集中在近1个月左右，但为了应对所有需求场景，业务要求计算近半年以上的历史。
- 不支持明细数据的查询。

解决方案：引入MPP引擎，数据现用现算

既然变化维的历史数据预计算成本巨大，最好的办法就是现用现算，但现用现算需要强大的并行计算能力。OLAP的实现有MOLAP、ROLAP、HOLAP三种形式，MOLAP以Cube为表现形式，但计算与管理成本较高。ROLAP需要强大的关系型DB引擎支撑。长期以来，由于传统关系型DBMS的数据处理能力有限，所以ROLAP模式受到很大的局限性。随着分布式、并行化技术成熟应用，MPP引擎逐渐表现出强大的高吞吐、低时延计算能力，号称“亿级秒开”的引擎不在少数，ROLAP模式可以得到更好的延伸。单从业务实际应用考虑，性能在千万量级关联查询现场计算秒开的情况下，已经可以覆盖到很多应用场景，具备应用的可能性。例如：日数据量的ROLAP现场计算，周、月趋势的计算，以及明细数据的浏览都可以较好的应对。

下图是MOLAP模式与ROLAP模式下应用方案的比较：

MOLAP模式的劣势

1. 应用层模型复杂，根据业务需要以及Kylin生产需要，还要做较多模型预处理。这样在不同的业务场景中，模型的利用率也比较低。
2. Kylin配置过程繁琐，需要配置模型设计，并配合适当的“剪枝”策略，以实现计算成本与查询效率的平衡。
3. 由于MOLAP不支持明细数据的查询，在“汇总+明细”的应用场景中，明细数据需要同步到DBMS引擎来响应交互，增加了生产的运维成本。
4. 较多的预处理伴随着较高的生产成本。

ROLAP模式的优势

1. 应用层模型设计简化，将数据固定在一个稳定的数据粒度即可。比如商家粒度的星形模型，同时复用率也比较高。
2. App层的业务表达可以通过视图进行封装，减少了数据冗余，同时提高了应用的灵活性，降低了运维成本。
3. 同时支持“汇总+明细”。
4. 模型轻量标准化，极大的降低了生产成本。

综上所述，在变化维、非预设维、细粒度统计的应用场景下，使用MPP引擎驱动的ROLAP模式，可以简化模型设计，减少预计算的代价，并通过强大的实时计算能力，可以支撑良好的实时交互体验。

双引擎下的应用场景适配问题

架构上通过MOLAP+ROLAP双引擎模式来适配不同应用场景，如下图所示：

技术权衡

MOLAP
：通过预计算，提供稳定的切片数据，实现多次查询一次计算，减轻了查询时的计算压力，保证了查询的稳定性，是“空间换时间”的最佳路径。实现了基于Bitmap的去重算法，支持在不同维度下去重指标的实时统计，效率较高。
ROLAP
：基于实时的大规模并行计算，对集群的要求较高。MPP引擎的核心是通过将数据分散，以实现CPU、IO、内存资源的分布，来提升并行计算能力。在当前数据存储以磁盘为主的情况下，数据Scan需要的较大的磁盘IO，以及并行导致的高CPU，仍然是资源的短板。因此，高频的大规模汇总统计，并发能力将面临较大挑战，这取决于集群硬件方面的并行计算能力。传统去重算法需要大量计算资源，实时的大规模去重指标对CPU、内存都是一个巨大挑战。目前Doris最新版本已经支持Bitmap算法，配合预计算可以很好地解决去重应用场景。

业务模型适配

MOLAP： 当业务分析维度相对固化，并在可以使用历史状态时，按照时间进行增量生产，加工成本呈线性增长状态，数据加工到更粗的粒度（如组织单元），减少结果数据量，提高交互效率。如上图所示，由A模型预计算到B模型，使用Kylin是一个不错的选择。

ROLAP： 当业务分析维度灵活多变或者特定到最新的状态时（如上图A模型中，始终使用最新的商家组织归属查看历史），预计算回溯历史数据成本巨大。在这种场景下，将数据稳定在商家的粒度，通过现场计算进行历史数据的回溯分析，实现现用现算，可以节省掉预计算的巨大成本，并带来较大的应用灵活性。这种情况下适合MPP引擎支撑下的ROLAP生产模式。

MPP引擎的选型

目前开源的比较受关注的OLAP引擎很多，比如Greenplum、Apache Impala、Presto、Doris、ClickHouse、Druid、TiDB等等，但缺乏实践案例的介绍，所以我们也没有太多的经验可以借鉴。于是，我们就结合自身业务的需求，从引擎建设成本出发，并立足于公司技术生态融合、集成、易用性等维度进行综合考虑，作为选型依据，最终我们平台部门选择了2018年刚进入Apache社区的Doris。

Doris简介及特点

Doris是基于MPP架构的OLAP引擎，主要整合了Google Mesa（数据模型）、Apache Impala（MPP Query Engine）和Apache ORCFile（存储格式，编码和压缩）的技术。

Doris的系统架构如下，主要分为FE和BE两个组件，FE主要负责查询的解析、编译、优化、调度和元数
据管理；BE主要负责查询的执行和数据存储。关于Doris的更多技术细节，可参考其官方文档。

Doris的特点：

- 同时支持高并发点查询和高吞吐的Ad-hoc查询。
- 同时支持离线批量导入和实时数据导入。
- 同时支持明细和聚合查询。
- 兼容MySQL协议和标准SQL。
- 支持Rollup Table和Rollup Table的智能查询路由。
- 支持较好的多表Join策略和灵活的表达式查询。
- 支持Schema在线变更。
- 支持Range和Hash二级分区。

Doris在外卖数仓中的应用效率

上图是我们在一个分析项目改造中的评估项目收益，整体在查询效率不变的情况下，生产耗能及存储成本都有较大收益。

以20台BE+3FE的Doris环境，效率、性能表现情况如下：

- 支撑数据分析产品数十个以上，整体响应达到ms级。
- 支持百万、千万级大表关联查询，同时进行维表关联的雪花模型，经过Colocate Join特性优化，可以实现秒级响应。
- 日级别，基于商家明细现场计算，同时满足汇总及下钻明细查询，查询时效基本都可以控制在秒级。
- 7日趋势分析，2~3秒。由于数据量较大，根据集群规模不同查询性能有所区别，但数据量较大时，调动的集群资源较多，因此MPP的并发性能受限于集群的性能。一般原则是并发较高的业务，需要严格控制查询时效（基本在毫秒级），对于并发不高的业务，允许进行较大的查询，但也要考虑集群的承受能力。
- 通过一年来的应用以及Doris的不断改进升级，Doris的高可靠、高可用、高可扩展性也得到进一步验证，服务稳定可靠。

准实时场景下的应用

离线业务分析大多基于T+1的离线数据，但在营销活动场景下，外卖团队往往需要当日的实时数据进行业务变化的监控与分析，通常情况下会采用实时流计算来实现。

外卖实时业务监控有如下特点：

- 避免分钟级的生产波动影响，业务上10、15分钟准实时数据可以满足分析需要。
- 实时数据需要与离线数据进行日环比与周同比的比对。
- 订单业务需要事件时间，体验业务需要生产时间，业务对齐逻辑复杂。
- 不同业务线需求差异大，指标需要良好扩展性。

由于业务上的复杂性，实时流计算中，需要考虑诸多业务口径的对齐，业务ER模型在合流处理中开发成本较高，资源占用较大，通过设计基于Doris的准实时生产数仓，可以灵活地实现业务微批处理，且开发生产成本都比较低。以下为基于Doris的准实时数仓架构设计，是典型的实时Lambda生产架构：

实现准实时计算方案，需要以下能力的支撑：

实时的写入能力：目前支持Kafka To
Doris秒级延迟。在可靠性、稳定性建设方面仍需进一步提升。
引擎建设
：短平快的计算+高效的存储性能。目前Doris引擎性能仍有进步空间，2020年将有较大改进提升

，随着后续Page Cache，内存表等能力的上线，IO将不再拖后腿，并发能力将有较大提升。
可靠的调度能力：提供5、10、15、30分钟的调度保障能力。
Lambda架构简化：实时数据与离线数据更好的在Doris中进行融合，灵活支撑应用。
高效的OLAP交互
：支撑业务的灵活查询访问，业务层通过视图进行逻辑封装直接复用汇总层多维模型，提高了开发效率，减少了运维成本。

相比Storm、Flink中的窗口计算，准实时DB微批的优势：

Doris引擎在美团的重要改进

Join 谓词下推的传递性优化

如上图所示，对于下面的 SQL：

select * from t1 join t2 on t1.id = t2.id where t1.id = 1

Doris开源版本默认会对t2表进行全表Scan，这样会导致上面的查询超时，进而导致外卖业务在Doris上的第一批应用无法上线。

于是我们在Doris中实现了第一个优化：Join谓词下推的传递性优化（MySQL和TiDB中称之为Constant Propagation）。Join谓词下推的传递性优化是指：基于谓词t1.id = t2.id和t1.id = 1,
我们可以推断出新的谓词t2.id = 1，并将谓词t2.id = 1下推到t2的Scan节点。 这样假如t2表有数百个分区的话，查询性能就会有数十倍甚至上百倍的提升，因为t2表参与Scan和Join的数据量会显著减少。

查询执行多实例并发优化

如上图所示，Doris默认在每个节点上为每个算子只会生成1个执行实例。这样的话，如果数据量很大，每个执行实例的算子就需要处理大量的数据，而且无法充分利用集群的CPU、IO、内存等资源。

一个比较容易想到的优化手段是，我们可以在每个节点上为每个算子生成多个执行实例。这样每个算子只需要处理少量数据，而且多个执行实例可以并行执行。

下图是并发度设置为5的优化效果，可以看到对于多种类型的查询，会有3到5倍的查询性能提升：

Colocate Join

Colocate Join（Local Join）是和Shuffle Join、Broadcast Join相对的概念，即将两表的数据提前按照Join Key Shard，这样在Join执行时就没有数据网络传输的开销，两表可以直接在本地进行Join。

整个Colocate Join在Doris中实现的关键点如下：

- 数据导入时保证数据本地性。
- 查询调度时保证数据本地性。
- 数据Balance后保证数据本地性。
- 查询Plan的修改。
- Colocate Table元数据的持久化和一致性。
- Hash Join的粒度从Server粒度变为Bucket粒度。
- Colocate Join的条件判定。

关于Doris Colocate Join的更多实现细节，可以参考《Apache Doris Colocate Join 原理与实践》。

对于下面的SQL，Doris Colocate Join和Shuffle Join在不同数据量下的性能对比如下：

```
select count(*) FROM A t1 INNER JOIN [shuffle] B t5    ON ((t1.dt = t5.dt) AND (t1.id = t5.id)) IN
NER JOIN [shuffle] C t6    ON ((t1.dt = t6.dt) AND (t1.id = t6.id)) where t1.dt in (xxx days);
```

Bitmap 精确去重

Doris之前实现精确去重的方式是现场计算的，实现方法和Spark、MapReduce类似：

对于上图计算PV的SQL，Doris在计算时，会按照下图的方式进行计算，先根据page列和user_id列group by，最后再Count：

显然，上面的计算方式，当数据量越来越大，到几十亿几百亿时，使用的IO资源、CPU资源、内存资源、网络资源会变得越来越多，查询也会变得越来越慢。

于是我们在Doris中新增了一种Bitmap聚合指标，数据导入时，相同维度列的数据会使用Bitmap聚合。有了Bitmap后，Doris中计算精确去重的方式如下：

可以看到，当使用Bitmap之后，之前的PV计算过程会大幅简化，现场查询时的IO、CPU、内存，网络资源也会显著减少，并且不再会随着数据规模而线性增加。

总结与思考

在外卖运营分析的业务实践中，由于业务的复杂及应用场景的不同，没有哪一种数据生产方案能够解决所有业务问题。数据库引擎技术的发展，为我们提供更多手段提升数据建设方案。实践证明，以Doris引擎为驱动的ROLAP模式可以较好地处理汇总与明细、变化维的历史回溯、非预设维的灵活应用、准实时的批处理等场景。而以Kylin为基础的MOLAP模式在处理增量业务分析，固化维度场景，通过预计算以空间换时间方面依然重要。

业务方面，通过外卖数仓Doris的成功实践以及跨BG的交流，美团已经有更多的团队了解并尝试使用Doris方案。而且在平台同学的共同努力下，引擎性能还有较大提升空间，相信以Doris引擎为驱动的ROLAP模式会为美团的业务团队带来更大的收益。从目前实践效果看，其完全有替代Kylin、Druid、ES等引擎的趋势。

目前，数据库技术进步飞速，近期柏睿数据发布全内存分布式数据库RapidsDB v4.0支持TB级毫秒响应（处理千亿数据可实现毫秒级响应）。可以预见，数据库技术的进步将大大改善数仓的分层管理与应用支撑效率，业务将变得“定义即可见”，也将极大地提升数据的价值。

参考资料

- Doris文档和源码
- Apache Kylin VS Apache Doris

作者简介

- 朱良，美团外卖数据仓库工程师。
- 凯森，美团大数据工程师，Apache Kylin Committer。

本文原文：Apache Doris在美团外卖数仓中的应用实践