Apache Kylin在美团数十亿数据OLAP场景下的实践

本文根据2016年4月北京Apache Kylin Meetup上的分享讲稿整理，略有删节。
美团各业务线存在大量的OLAP分析场景，需要基于Hadoop数十亿级别的数据进行分析，直接响应分析师和城市BD等数千人的交互式访问请求，对OLAP服务的扩展性、稳定性、数据精确性和性能均有很高要求。本文主要介绍美团的具体OLAP需求，如何将Kylin应用到实际场景中，以及目前的使用方式和现状。同时也将Kylin和其它系统（如Presto、Druid等）进行了对比，阐述了Kylin的独特优势。

作为公司的平台部门，需要给各个业务线提供平台的服务，那么如何建设一个满足各种需求的公司平台级OLAP分析服务呢。首先，一个开源项目在公司真正落地会遇到很多障碍，这主要是由各个业务线不同的数据特点和业务特点决定的，所以本文会介绍一下美团的数据场景有什么特点；其次，针对这些数据特点，尤其是和Kylin设计初衷不太相符的部分，有什么样的解决方案；第三，目前OLAP领域还没有所谓事实上的标准，很多引擎都可以做类似事情，比如普通的MPP，Kylin，或者ES等。这些系统之间的对比情况如何，应该如何选择，我们也有部分测试数据可以分享；最后，简单讨论一下未来准备在Kylin上做的工作。

美团的数据场景特点

第一个特点是数据规模和模型特点。一方面从数据规模上来讲，事实表一般在1亿到10亿量级，同时还有千万量级的维表，也就是超高基数的维表。另一方面，数据模型是一开始遇到的最大困难。因为Kylin最初的设计是基于一个星形模型的，但很不幸由于各种原因，很多数据都是雪花的模型，还有其它的模型，比如所谓“星座”模型，也就是中间是两张或者三张事实表，周围关联了其它很多维表。业务逻辑决定了这些数据的关联方式非常复杂，根本无法用经典标准的理论来解释。

第二个是维度。维度最理想的情况是固定的，每天变化的只是事实表。但实际上维度经常会变，这可能和行业特点有关，比如组织架构，相关的维度数据可能每天都会变化。除此之外还可能要用今天的维度去关联所有的历史数据，因此要重刷历史数据，相应的开销也比较大。

第三个是数据回溯的问题。比如发现数据生成有问题，或者上游出错了，此时就需要重跑数据。这也是和经典理论模型有区别的。

从维度的角度来看，一般维度的个数在5-20个之间，相对来说还是比较适合用Kylin的。另一个特点是一般都会有一个日期维度，有可能是当天，也有可能是一个星期，一个月，或者任意一个时间段。另外也会有较多的层次维度，比如组织架构从最上面的大区一直到下面的蜂窝，就是一个典型的层次维度。

从指标的角度来讲，一般情况下指标个数在50个以内，相对来说Kylin在指标上的限制并没有那么严格，都能满足需求。其中有比较多的表达式指标，在Kylin里面聚合函数的参数只能是单独的一列，像sum(if...)这种就不能支持，因此需要一些特别的解决方法。另外一个非常重要的问题是数据的精确性，目前在OLAP领域，各个系统都是用hyperloglog等近似算法做去重计数，这主

要是出于开销上的考虑，但我们的业务场景要求数据必须是精确的。因此这也是要重点解决的问题。

在查询上也有比较高的要求。因为平台的查询服务可能直接向城市BD开放，每次会有几十、上百万次的访问，所以稳定性是首先要保证的。第二要求有很高的性能。因为用Kylin主要是为了实现交互式的分析，让使用者能够很快拿到结果，所以需要秒级响应。

另外经常会有人问到，Kylin有没有可视化的前端，在我们内部更多是由业务方来做，因为原来本身就有这样的系统，以前接的是MySQL等其它的数据源，现在可以直接使用Kylin的JDBC driver对接起来。

以上是美团在OLAP查询方面的一些特点。在用Kylin之前，实际上有一些方案，但效果并不理想。

比如用Hive直接去查，这种情况下，第一个是慢，第二会消耗计算集群的资源。尤其每个月第一天，大家都要出月报，跑的SQL非常多，全提到集群上去，并发度限制导致跑的比平时更慢。

我们原来也做过预聚合的尝试，这个思路跟Kylin很像，只不过是自己做这个事，用Hive先把所有的维度算出来，然后导入MySQL或者HBase。但是这个方案并没有像Kylin这么好的模型定义抽象，也没有从配置到执行，预计算，查询这样整体的框架。现在通过使用Kylin实现了低成本的解决这些问题。

接入Apache Kylin的解决方案

针对上述的问题，经过大量的尝试和验证，目前主要的解决方案有以下几点。最重要的第一点，就是采用宽表。所有非标准星型的数据模型，都可以通过预处理先拉平，做成一个宽表来解决。只要能根据业务逻辑把这些表关联起来，生成一张宽表，然后再基于这张表在Kylin里做数据的聚合就可以了。宽表不只能解决数据模型的问题，还能解决维度变化、或者超高基数的维度等问题。

第二点是表达式指标的问题，也可以通过提前处理解决。把表达式单独转成一列，再基于这列做聚合就可以了。实际上宽表和表达式变换的处理可以用hive的view，也可以生成物理表。

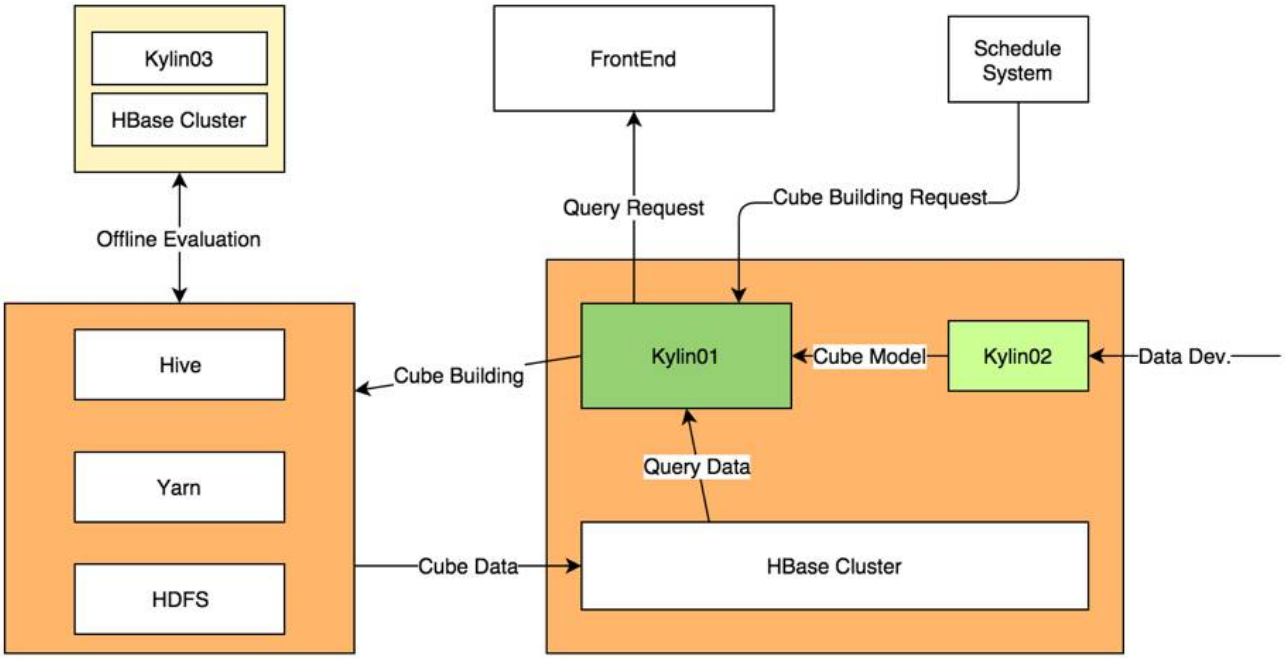
第三个是精确去重的问题，目前的方案是基于Bitmap。由于数据类型的限制，目前只支持int类型，其它包括long、string等类型还不支持。因为需要把每个值都能映射到Bitmap里,如果是long的话开销太大。如果用哈希的话就会冲突，造成结果不准确。另外Bitmap本身开销也是比较大的，尤其跑预计算的时候，如果算出来的基数很大，对应的数据结构就是几十兆，内存会有OOM的风险。这些问题后面我们也会想一些办法解决，也欢迎在社区里一起讨论。（补充说明：目前已在1.5.3版本中实现了全类型精确去重计数的支持。）

从整个系统的部署方式上来说，目前Server采用了分离部署的方式。Kylin Server本质上就是一个客户端，并不需要太多资源，一般情况下使用虚拟机就能够满足需求。

实际的部署情况可以看这张图，左下角的是hadoop主集群，用于执行每天所有hadoop作业

。中间最重要的是Kylin01和02这两个server，是用于线上环境的serve。其中kylin01是生产环境，这个环境一方面要负责从主机群上跑计算，把数据导到HBase，另外也要响应前端的请求，从HBase里读数据。如果想新增一个Cube的话，需要在kylin02上操作，也就是预上线环境。所有业务方人员的cube数据模型定义都是在kylin02上做，没有问题后由管理员切到kylin01上。

这样做的一个好处是kylin01作为一个线上服务能保证稳定性，甚至权限控制能更严格一些；第二，预上线环境下开发完成后，管理员可以在投入生产前进行一次review，保证cube的效率。



右上角是另外的调度系统。整个数据仓库的数据生产都是通过这个调度系统来调度的，其中的任务类型很多，Kylin的cube build任务也是作为其中的一种类型。在上游的数据就绪以后，根据配置的依赖关系，自动触发Cube建立的过程。

左上角这边还有一个完全独立的线下测试集群，这个集群是完全开放的，主要是给用户做一些最开始的可行性调研或者评估的工作，但同时也不保证稳定性。

一个开源的系统从社区拿回来，到真正的落地，再到上生产，这个过程相对还是比较长的，这里并没有太多的技术问题，更多的是一些流程上的经验。就是如何在各个阶段给业务方提供更好的服务，使得接入Kylin的过程更顺畅，沟通成本更低。整个过程主要分为四个阶段。

第一个阶段是方案选型，业务方根据业务需求，选择一些方案进行调研。我们在这个阶段提供了需求的Checklist，从数据模型，维度各个方面列出来比较详细的点，可以让业务方自己对照，确定需求是不是能够被满足。

在确定Kylin能满足需求的基础上，接下来是第二步，线下探查，也就是线下评估或者测试。我们提供了非常详细的接入文档，以及线下测试的环境。

第三步是线上开发，我们也有一些文档支持，基于抽象出来的场景，每个场景怎么配置Cube

，或者做哪些预处理，如何优化等，能够给业务方一个指导性的意见。

最后是开发完成后的切表上线。这个过程目前还是由管理员来操作，一方面是为了避免误操作或者滥操作，另一方面也会对cube进行review，帮助进行优化。

主流OLAP系统对比分析

通过和其它同学交流，有一个感觉就是大家都觉得Kylin还不错，但并不是特别有信心，或者不知道非要用它的理由是什么，或者它和其它系统的对比是什么样的？这里也有部分测试结果可以和大家分享。

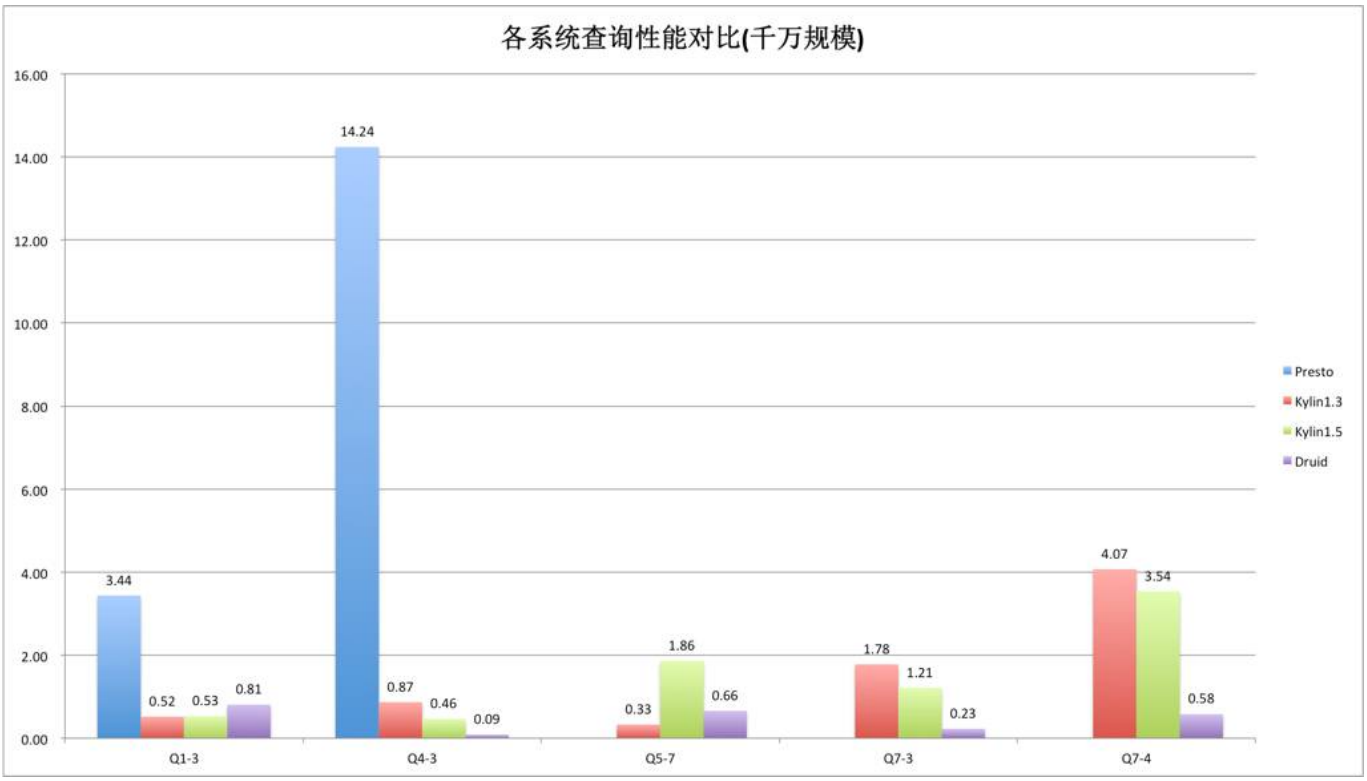
整个测试基于SSB的数据集，也是完全开源的，实际上是专门用于星型模型OLAP场景下的测试。整个测试数据集是非常标准的五张表，可以配置一些参数决定生成的数据集规模，然后在不同的规模下做不同查询场景的测试。现在已经完成的测试的系统包括：Presto，Kylin1.3，Kylin1.5和Druid。数据规模包含千万、亿、十亿三种规模；维度个数为30个；指标个数为50个。典型的测试场景包括：上卷、下钻，和常用的聚合函数。

这里挑选了典型的五个查询场景：一个事实表的过滤和聚合；五张表全关联之后的查询；两个Count Dstinct指标和两个Sum指标；后面两个查询包含8~10个的维度过滤。

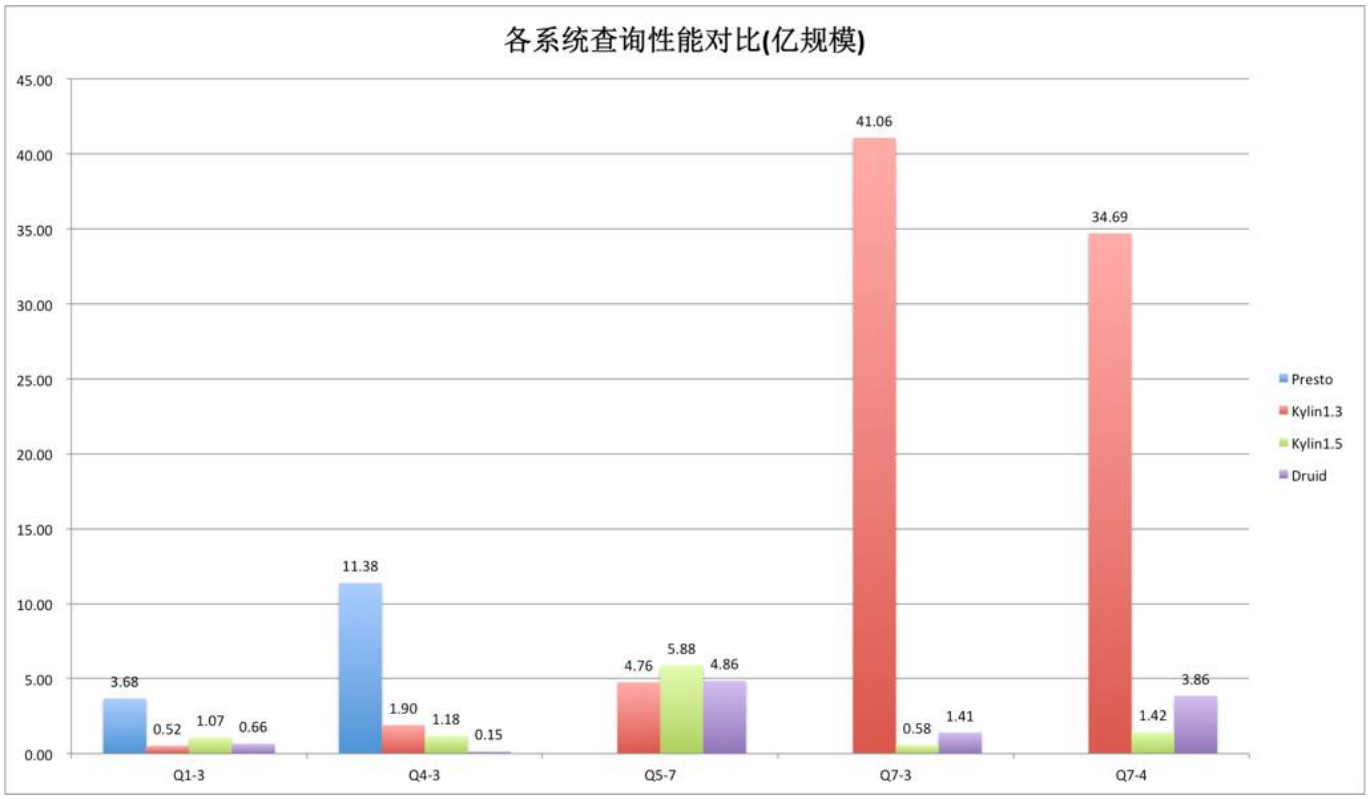
这张图是千万规模下的一个测试结果，包括了四个系统。我们在用Kylin或者其它系统之前没有专门用于OLAP分析的引擎，只能用通用的。Presto是其中表现非常好的引擎，但是在OLAP这种特定的场景下，可以看到不管跟Kylin还是Druid相比差的都比较多，所以前两个测试包含了Presto结果，后面就没有包含了

这里比较有趣的现象是在第三个查询，Kylin1.5反而比Kylin1.3要慢一些。这个地方我们还没有搞清楚是什么原因，后面会详细的看一下。当然这个也可以证明数据没有修改过，是真实的测试数据。

从后面的两个查询上可以看到，在千万规模的级别，和Druid还是有比较大的差距。这主要和它们的实现模式相关，因为Druid会把所有的数据预处理完以后都加载到内存里，在做一些小数据量聚合的时候，可以达到非常快的速度；但是Kylin要到HBase上读，相对来说它的性能要差一些，但也完全能满足需求。



在亿级的规模上情况又有了变化，还是看后面两个查询，Kylin1.3基本上是一个线性的增长，这个数据已经变得比较难看了，这是由于Kylin1.3在扫描HBase的时候是串行方式，但是Kylin1.5反而会有更好的表现，这是因为Kylin1.5引入了HBase并行Scan，大大降低了扫描的时间。Kylin1.5的数据会shard到不同的region上，在千万量级上数据量还比较小，没有明显的体现，但是上亿以后，随着数据量上升，region也变多了，反而能把并发度提上去。所以在这里可以看到Kylin1.5表现会更好。这里也可以看出，在数据量成数量级上升后，Kylin表现的更加稳定，在不同规模数据集上依然可以保持不错的查询性能。而Druid随着数据量的增长性能损失也成倍增长。



刚才是在性能方面做的一些分析，其实对于一个系统来说，性能只是一个方面，除此之外，我们也会去考量其它方面的情况，主要有以下四点。

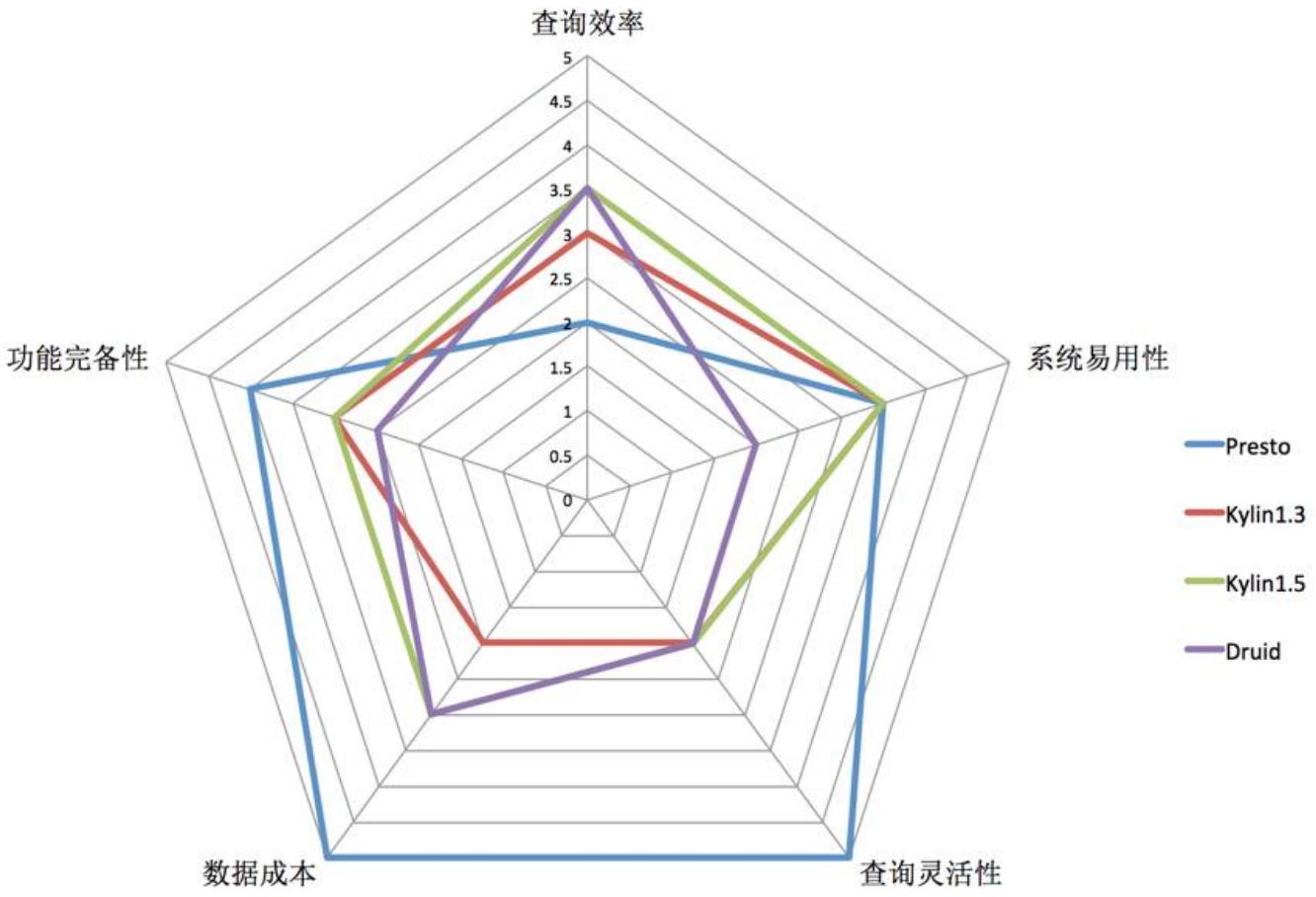
第一，功能的完备性。刚才提到我们所有的数据必须是精确的，但是现在基本上没有哪个系统能完全覆盖我们这个需求。比如Druid性能表现确实更好，但是它去重计数没有办法做到精确。

第二，系统的易用性。作为一个平台服务，不仅要把系统用起来，还要维护它，因此要考虑部署和监控的成本。这方面Kylin相对来说也是比较好的。Druid一个集群的角色是非常多的，如果要把这个系统用起来的话，可能光搭这个环境，起这些服务都要很长的时间。这个对于我们做平台来讲，实际上是一个比较痛的事。不管是在部署，还是加监控的时候，成本都是相对比较高的。另外一个查询接口方面，我们最熟悉或者最标准，最好用的当然是标准SQL的接口。ES、Druid这些系统原来都不支持SQL，当然现在也有一些插件，但是在功能的完备性和数据的效率上都不如原生的支持。

第三，数据成本。刚才提到了有些数据需要做一些预处理，比如表的拉平或者表达式列的变换，除此之外还有一些格式的转化，比如有的系统只能读TEXT格式，这样都会带来数据准备的成本。另一方面是数据导入的效率。从数据进入数据仓库到真正能够被查询，这个时间中间有多长。数据存储和服务的时候需要多少机器资源，这个都可以归为数据成本，就是使用这个数据需要付出的成本。

第四，查询灵活性。经常有业务方问到，如果Cube没定义的话怎么办？现在当然查询只能失败。这个说明有的查询模式不是那么固定的，可能突然要查一个数，但以后都不会再查了。实际上在需要预定义的OLAP引擎上，这种需求普遍来讲支持都不是太好。

这张图是各个系统全方位的一个对比。



从查询效率上看，这里表现最不好的就是Presto，表现最好的应该是Druid和Kylin1.5，两者不相上下。从功能完备性上来讲，确实Presto语法和UDF等等是很完备的，Kylin会稍微差一些，但比Druid好一点。

系统易用性上区别不是太大，这里主要考虑有没有标准的SQL接口或者部署成本高不高，用户上手能不能更快，目前来看Druid接口上确实不够友好，需要去翻它的文档才知道怎么去写查询的语法。

在查询成本上，Presto是最好的，因为几乎不需要做什么特殊的处理，基本上Hive能读的数据Presto也都能读，所以这个成本非常低。Druid和Kylin的成本相对较高，因为都需要提前的预计算，尤其是Kylin如果维度数特别多，而且不做特别优化的话，数据量还是很可观的。

最后从灵活性上来讲， Presto只要SQL写出来怎么查都可以，Druid和Kylin都要做一些预先模型定义的工作。这方面也可以作为大家选型时候的参考。

刚才比较客观的对比了几个系统，接下来再总结一下Kylin的优势。

第一，性能非常稳定。因为Kylin依赖的所有服务，比如Hive、HBase都是非常成熟的，Kylin本身的逻辑并不复杂，所以稳定性有一个很好的保证。目前在我们的生产环境中，稳定性可以保证在99.99%以上。同时查询时延也比较理想。我们现在有一个业务线需求，每天查询量在两万次以上，95%的时延低于1秒，99%在3秒以内。基本上能满足我们交互式分析的需求。

第二，对我们特别重要的一点，就是数据的精确性要求。其实现在能做到的只有Kylin，所以说我们也没有什么太多其他的选择。

第三，从易用性上来讲，Kylin也有非常多的特点。首先是外围的服务，不管是Hive还是HBase，只要大家用Hadoop系统的话基本都有了，不需要额外工作。在部署运维和使用成本上来讲，都是比较低的。其次，有一个公共的Web页面来做模型的配置。相比之下Druid现在还是基于配置文件来做。这里就有一个问题，配置文件一般都是平台方或者管理员来管理的，没办法把这个配置系统开放出去，这样在沟通成本和响应效率上都不够理想。Kylin有一个通用的Web Server开放出来，所有用户都可以去测试和定义，只有上线的时候需要管理员再review一下，这样体验就会好很多。

第四，最后一点就是活跃开放的社区和热心的核心开发者团队，社区里讨论非常开放，大家可以提自己的意见及patch，修复bug以及提交新的功能等，包括我们美团团队也贡献了很多特性，比如写入不同的HBase集群等。这里特别要指出的是核心团队都是中国人，这是Apache所有项目里唯一中国人为主的顶级项目，社区非常活跃和热心，有非常多的中国工程师。特别是当你贡献越来越多的时候，社区会邀请成为committer等，包括我自己及团队成员也已经是Apache Kylin的committer。同时也非常高兴看到以韩卿为首的Apache Kylin核心团队在今年初成立的创业公司Kyligence，相信可以为整个项目及社区的发展带来更大的空间和未来。

未来工作

在未来工作方面，我们认为Kylin还有一些不理想的方面，我们也会着力去做优化和改进。

第一，精确去重计数。刚才提到只支持Int，接下来有一个方案会支持所有的数据类型，能够扩展大家使用精确去重的场景范围（补充说明：目前该功能已在1.5.3版本中实现）。

第二，在查询效率和Build效率上也看到了一些可以优化的部分。比如队列资源拆分，我们所有计算集群的资源都是按照业务线核算成本的，但是现在Kylin本身还不太支持，这个我们也会抓紧去做，相信在很多公司也有类似的需求。还有大结果集和分页。当结果到了上百万的量级时，查询时延会上升到几十秒。同时在查询的时候有可能需要排序并且分页，就是把结果全读出来之后，根据其中的一个指标再order by，这个开销也是比较大的。我们也会想办法进行优化。

最后，Kylin1.5之后有明细数据和Streaming特性出来，后面也会做这方面的尝试。

Q&A

Q1：之前在Build的时候一直提到成本的问题，能给出一个估计值吗，如果一百亿的数据，需要多少时间？

孙业锐：有一个简单数据，大概是两亿行数据，维度的话有十四五个，Build时间不超过两个小时，Build出来的数据是五六百G。

Q2：原始值是多大？

孙业锐：把这个数据抽出来之后，就是只参与Build的数据压缩后只有几个G。

Q3：Kerberos认证失效的问题你们遇到过没有？

孙业锐： Kerberos认证完之后，会在本地临时目录下有一个ticket问题，每天在外部定时刷新一下就可以了，服务是不用停的。

主持人：我补充一下我们为什么选择SQL接口？Kylin解决的是真正的用户面是谁，其实是业务人员和分析人员，他只会SQL，几乎那些人很少说再学个JAVA，所以能给他一个标准的SQL这个是让他上船最快的事情。其实这就是易用性很重要。

刚才看到了Kylin在千万级规模和亿级规模的表现，如果数据规模上到十亿，百亿，千亿的时候，我相信Kylin应该会秒杀所有一切。因为我们现在有另一个案例，生产环境上千亿规模的一张表，可以做到90%查询在1.8秒以内。另外我觉得非常好的一点，像美团、京东这边贡献了很多patch，其实就是把需求提出来，大家可以一起来做。

作者介绍：孙业锐，美团高级工程师，Apache Kylin Committer。2012年毕业于电子科技大学，曾在奇虎360工作，负责Hadoop平台建设，2015年加入美团。目前主要负责数据生产和查询引擎的改进和优化，专注于分布式计算，OLAP分析等领域，对分布式存储系统亦有丰富经验。

本文转载自：
http://mp.weixin.qq.com/s?__biz=MzA5NzkxMzg1Nw==&mid=2653160070&idx=1&sn=08263dc11c67e6137aa8d0a4ee6e2d85