美图数据统计分析平台架构演进

卢荣斌 Go中国 2017-08-31

内容提要：美图拥有十亿级用户，每天有数千万用户在使用美图的各个产品，从而积累了大量的用户数据。随着App的不断迭代与用户的快速膨胀，产品、运营、市场等越来越依赖于数据来优化产品功能、跟踪运营效果，分析用户行为等，随之而来的有越来越多的数据统计、分析等需求，那么如何应对和满足不断膨胀的数据统计与分析需求？业务的不断发展又怎么推进架构实现的改造？本文将介绍大数据业务与技术的碰撞产物之一：美图大数据统计分析平台的架构演进，希望通过这次分享能给大家带来一些解决数据业务与架构方面的思考。

嘉宾介绍： 卢荣斌，毕业于厦门大学，14年加入美图，主导美图大数据平台架构设计与开发工作，负责美图大数据基础建设、数据服务架构以及数据统计分析等工作，经历过美图大数据平台从无到有的搭建与架构演进，长期关注大数据相关技术体系，积累了多年大数据架构与实践经验。

如果有做过大数据相关开发的同学应该知道数据统计是一个比较尴尬的事情，第一个它可能不是一个非常有技术含量的事情，对于技术人员的成长来说不是非常好。第二它可能是一个比较重复工作的事，需要解决一些简单的需求的重复工作。

美图其实有非常多的App，每个 App 基本上都会有相应的产品运营、销售以及数据分析的同学，这些同学会提各式各样数据统计的需求，比如数据报表或者数据分析的需求。这么多的分析或者说数据统计的需求，在美图是怎么解决的呢？今天主要想介绍下在美图的解决方案，希望大家在平时工作中有遇到类似的问题的话能有一些共鸣之处。
内容主要分三块，第一部分是统计业务与技术碰撞。经过第一部分的演进以后，我们有沉淀下来一些东西，有一些思考知道怎么去解决这些问题，所以第二块主要是介绍美图统计平台架构的实现。第三部分将介绍我们正在做的事情以及未来的一些规划。

统计业务与技术碰撞

这基本上是我自己亲身的经历，刚开始一个人做这一块的业务，会碰到一些有意思的点，可能分三个阶段，第一个阶段是在项目的初期，我们是怎么样去应对一些产品的初期需求的。第二个阶段是当用户量爆发以后，业务数据源上来以后我们又是怎么迭代的。第三是作为一个有一点追求的技术人员来说，怎么让自己去脱离一些业务，得到一些成长。

第一阶段——项目初期

这个阶段特点非常明显：以美拍为例，初期整体的数据体量小；统计需求比较少主要是一些基础的统计指标；产品的迭代非常快，要求数据的统计指标能够快速地跟上产品的迭代速度。

图 1

这一阶段的解决方案在现在看来非常的简陋：一个是业务的服务端可能会有多个节点，保证可用性，每个业务节点的服务会打相应的日志到本地磁盘，然后会通过rsync的方式统一同步日志到一个数据存储节点，在这个节点上写一些简单的shell或者php脚本来实现统计逻辑，配置相应的crontab来定时触发统计任务，最终把数据结果存储到MySQL供展示层调用呈现报表。

第二阶段——快速发展阶段

当用户量突然爆发以后，数据量会不断的增大，产品运营、数据分析的需求越来越多，相应第一个阶段的解决方案会存在比较大的问题，主要有几个：第一个是单点存储的容量非常有限，第二是计算瓶颈很快就会遇上瓶颈，很多时候统计报表因为计算瓶颈导致报表第二天延迟产出，第三是我们用shell或者php脚本来实现统计逻辑，整体后续的维护成本比较大，需要调整一个统计逻辑或者增加一些过滤条件等都比较不方便。

图 2

所以我们做了一些调整：第一是实现了一套数据采集的系统，负责做服务端日志的数据采集工作，将数据最终落地存储到HDFS；第二是前面有提到说存储和计算的单点问题，所以我们自己搭建一个Hadoop集群来解决单点的存储和计算；第三是基于Hive来解决编写过多统计逻辑相关的代码。

第三阶段——有追求的程序员

当需求不断膨胀的时候，作为一个有追求的程序员会考虑到，重复的代码量非常多，即使我们有架一层Hive来写相应的代码，最后做一层数据的过滤或者一些聚合。其实每一个统计需求我们都需要写一个相应的 java 的实现。这个工作量非常的枯燥，也是不断重复。

图 3

一个有追求的程序员的话，可能不会甘于每天做重复的工作。因为在平时接触业务与实现过程中，深有体会统计业务逻辑的流程基本上是一致的，所以考虑抽象出这样一个相对通用的业务处理的流程，基本的流程是从数据源Query出数据，然后做一些业务方面的聚合或者过滤，最终把数据存储到DB。那在代码实现层面做了一层抽象，抽象一个统计的组件，包含Query、Aggregator以及DBStore，然后分别有一些不同Query和Store场景的实现。当做了一层这样的抽象以后，相比于前面的方案，生产力还是得到了比较大的提升。当时我一个人，一天能够做四五个统计需求，而抽象后一天从了解需求开始到实现大概能做七八个统计需求，整体效率有不错的提升。

美图统计平台的架构实现

做了上面的抽象以后还是有不少的痛点，一个是业务的依赖，指的是我们做一个统计需求最花时间成本的是去了解数据业务方的需求背景，了解他们的产品长什么样子或者他们的运营是做了什么活动，业务沟通背景的成本非常高。第二是即使做了抽象还是会有一些相应的重复代码的编码量，还需要做一个统计的组件选择相应的query，相应的业务逻辑的处理以及存储层的DBstore。第三主要是运维成本高，当时上线一个任务需要做一个包到线上，还需要改一些shell等脚本。第四是涉及到个人成长方面，当一个人长时间在做这样事情的话，对个人的技术成长其实是有比较大的瓶颈。

图 4

基于上面的痛点，我们来介绍一下我们是如何解决这些事情的。我们考虑去做一个平台，让业务在我们这个平台去使用，我们提供服务就好。图 4 是我们当时做平台化的大概思路，比如左边这个业务方有非常多的报表数据需求，也可能有App的数据场景、商业广告等的数据需求，我们希望能够提供这样的一个平台，业务的数据需求方在这个平台上面配置他们想要的数据指标，而这个平台负责数据的计算、存储，以及最终吐出相应的数据给数据应用方。

更进一步，在做这个平台时，我们可能需要考虑以下几个比较重要的点：第一，我们可能需要对统计任务有一个比较清晰的元数据描述，可以描述出这些统计任务的计算方式是什么样子，算子是什么。第二是这个统计任务的数据源来自于哪里，以及数据需要存储在什么地方更合适业务查询。第三个是需要有一个调度中心来统一调度所有统计任务的执行。第四要确保任务的最终的正确执行。
基于上面这几个点，考虑需要有一些不同的模块来负责上面的说的几大功能。我们大概有设计三个模块：第一个模块是JobManager，主要是提供平台，供应用方比较方便的配置，能管理任务元数据信息以及其他的数据仓库、app信息的管理等。第二个模块是Scheduler，就是任务的调度中心，负责调度所有的统计任务。第三是任务执行模块JobExecutor，负责任务从查询、聚合到最终的结果落地存储。
接下来详细介绍下这三个模块大概的功能点以及实现的方式。先说第一个模块JobManager，如图5，主要需要提供一个应用方在这个平台上去配置他们想要的数据，另外一个点是我们需要有整合数据仓库，整合数据仓库主要是为了业务方能够查看到他相应业务表的信息。右边这一块主要是对于元数据统计任务的描述，主要包含这几个大块，比如说数据的来源，统计的算子是什么以及存储的介质或者特殊场景的数据过滤器、维度聚合以及任务与任务之间的依赖关系描述。这个模块主要是抽象统计任务，对任务的元数据做统一的配置管理。

图 5

第二个比较大的模块就是任务调度scheduler。当前的实现方式比较简单，是单点的方式。当前有实现的几个点，一是能够根据任务的优先级来调度，二是能够根据任务定时的策略进行调度，三是能够调度工作流，就是依赖关系的调度。

图 6

第三个模块是任务执行的模块JobExecutor（图6）。根据任务的源信息从插件池里组装出具体的query组件，然后到具体的Query层（比如Hive）跑相应的数据，出来后的数据做一些过滤、维度方面的聚合，最终根据任务的信息来组装数据的存储层的组件，把统计数据结果写入到存储层。

图 7

讲完三个模块以后，我们来回顾一下这个统计平台的基础架构。左边有一个JobManager管理元数据，根据元数据去做统计任务的整体标准流程：查询、过滤、维度聚合、存储。有了这样基础的框架以后，可以满足一部分的基础数据统计的场景，但如果是要支持更多的数据统计的业务场景的话，需要做更多的功能的拓展（图8）。

图 8

这里面有四个大方向的功能拓展，第一个是针对临时取数的场景。不一定所有的业务都需要常规例行跑，有非常多的临时跑数场景，比如分析人员需要临时看一个相应app的功能数据或者是说运营需要看一个临时组织活动的数据指标等等，平时会遇到比较多的临时取数相应的需求。对于解决临时取数这一块的需求，我们做的功能有两个，一个是有提供直接填写SQL的功能，支持有SQL基础的用户临时提取数据。这块是扩展

Hive自集成的antlr来做HOL语法解析，解析出来以后，需要校验HOL的合法性，主要是排除一些类似insert、delete操作，以及限定跑数的时间范围以免长时间占用集群计算资源。

第二个功能扩展是尽量丰富数据源，在平时的需求中，会越来越多遇到需要导入业务方的MySQL的数据来做简单的数据统计或者join计算。所以这一块我们是有基于sqoop开发一个插件，来支持导入业务MySQL库表到Hadoop。第三就是Bitmap。这是我们美图自研的一套系统，主要是为了便于多维度去重以及做新增、留存等相应的计算，其原理主要是基于位bit之间的操作。

第三个功能扩展是多存储，当前大部分的数据是存储在MongoDB，介于传统关系型数据库以及NoSQL之间，既能大部分满足业务的查询场景，又能保证分布式的数据存储。第二就是有临时比较大的数据导出情况，业务方需要获取批量比较大的数据，他们能够导入到HDFS上面，然后业用方从HDFS导出数据用于不同的业务应用。第三也支持一些普通文本，比如csv等。第四就是MySQL，这个当前有支持一些分表的策略的存储。最后一块就是要丰富统计算子，当前有实现一些去重、数组、TopN等统计算子。

图 9

另外一个大的功能拓展方面主要是数据可视化（图9）。因为存储层是多样多样，原来的方式是我们的存储层直接暴露给应用的数据后台，他们从我们的存储层查询数据解析，这个方式有一些不太好的地方，第一个是数据台如果不去透明化这个存储层的话，展示层开发需要学习Hbase、MySQL、Mongo等，有比较大的学习成本。第二是数据安全方面的考虑，或者对数据存储的统一的管理，都是有比较不好的地方，所以后面整了一套统一通用的API，有定制一套统一数据的协议，方便展示层统一对接数据做展示。

图 10

但接下来还会有一些问题，我们需要去考虑平台化的数据安全（图10）。比如通常情况下，美拍的后台只能够获取到美拍相关的数据，而不允许美拍后台能获取到其他app商业广告的数据，所以我们有实现一个统一的认证中心CA，就是业务方后台需要先去CA获取相应的授权token，然后去请求统一通用API时都会带上acess token，通用API作为一个通用服务方，会去CA认证这个token是不是合法，合法的话才会在存储层查询相应的数据，最后返回给应用方。

图 11

所以到这边，回顾下美图统计平台的整体架构：有一个统一的任务调度中心调度所有的统计任务，然后是JobExecutor负责从插件池来找相应的插件，做一些查询、过滤等，最终数据存储到DB，会有一个统一的API封装存储层，然后处于一些安全方面考虑有接入CA，最终各个数据后台对接通用API来做数据展示。

当前正在做的事情以及未来小阶段的规划

图 12

当前正在做的事情有两块，还没有正式上线或者接入。第一块是我们自己开发一套分布式调度，这套调度主要偏一套通用调度平台，不仅调度统计的任务，后续可以调度所有的离线计算的任务以及离线统计的任务，接下来所有的统计任务都会迁移到这个通用分布式调度平台上做统一的任务调度，替代当前单点的简单版本的调度中心。后续也会去支持资源方面的隔离和资源方面的调度。第二块就是数据可视化。我们刚刚看到的前面那一块，业务方的所有数据后台需要一遍遍重复地接入我们的统一通用API，是有比较多的重复工作，另外一个痛点是确实是涉及到一些比较大的APP，比如美拍他们整体的统计报表非常多，后台基本上可能是成百上千这样的数据，如果一个数据需求方想看到他们自己的数据，对于他们从成百上千个数据指标去定位到自己的数据指标是非常困难的。数据可视化这么一个平台就是解决这样的问题，我不需要所有的业务方都接入这个通用API，在同一个平台可以选择想要的数据源或者自己可视化的报表，然后呈现自己个性化的数据指标，不需要再去跟所有应用的数据后台去对接我们的API，然后做一些图形方面的展示。所以数据可视化这一块主要是做统计以及提供定制化、个性化的数据报表，有点类似于网易的bdp或者阿里的dataV等平台。

另外两块是接下来一段时间规划要做的事情，第一是数据分析人员经常抱怨说这个数据能不能有更快的查询方式，所以是考虑搭建一个OLAP服务，比如基于kylin等来构建。第二个是实时统计方面，刚刚前面讲的要么是定时的常规的统计需求，要么就是临时的统计需求，没有实时。所以这一块考虑说后面这个平台也能对接实时的统计需求，能更快速地对接满足实时统计需求场景。