

SACC 第八届中国系统架构师大会  
2016 SYSTEM ARCHITECT CONFERENCE CHINA 2016

架构 创新之路

# 创业公司的大数据平台选型和演进

魔窗CTO张申竣

# 充分了解业务需求和产品所处阶段

创业公司面临的一些挑战和优势：

1. 资源不足
2. 时间压力大
3. 没有技术上的历史包袱，选型相对自由

前两点是搭建大数据平台的挑战，最后一点是优势。对于大部分创业公司而言，这三点挑战和优势始终存在，但是业务特点随着公司的发展会有相应的变化

# 创业公司的发展阶段

产品验证阶段  
产品成熟阶段  
业务增长阶段

## 魔窗的大数据平台监测报告业务需求

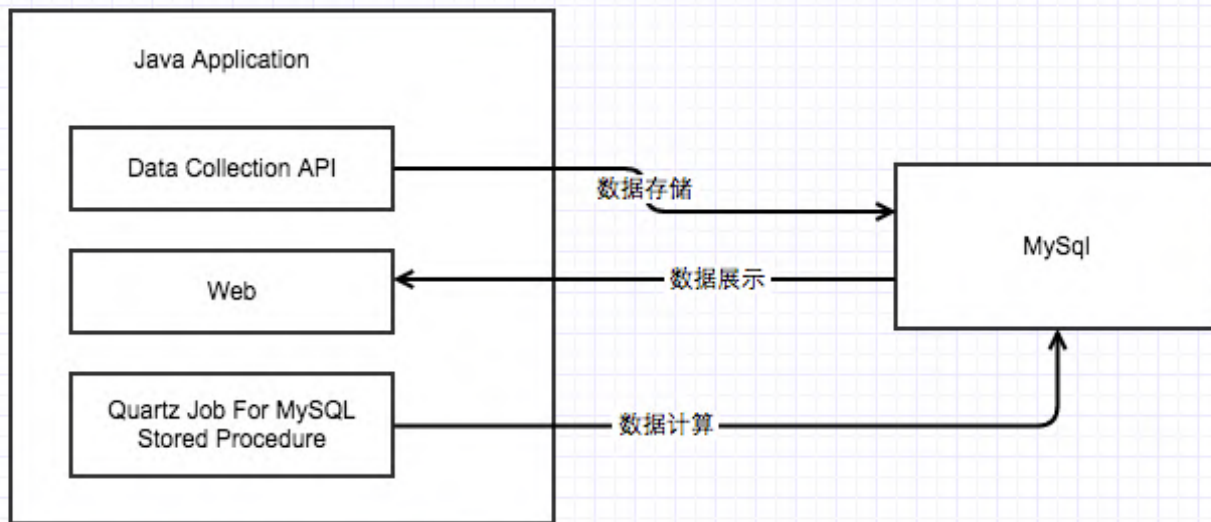
1. 计算由魔窗的移动端SDK采集过来的包括，日活，应用打开次数，流失用户，回流用户等移动端监测的常用指标
2. 因为魔窗是提供基于Deep Link的一系列应用唤醒服务，所以我們还需要监测，从投放在各个渠道的基于Deep Link生成的短链的曝光，安装转化率
3. 魔窗还提供各种营销活动的制作和投放，所以还需要监测营销活动的曝光率

## 业务特点

1. 数据量很小，用户最多只有几十个种子用户，整个监测采集到的数据规模根本不能称作大数据。
2. 魔窗所计算的统计指标也无法确定对用户是否有真正的帮助，很可能整个功能会根据市场反馈最后被砍掉。

这种情况下，魔窗首先考虑的是要尽量缩小产品验证的成本，所以技术选型的原则很简单，端到端跑通功能，设计和实现上越简单粗暴越好，不需要存在技术积累，被砍了也不可惜。所以这个时候架构的总的原则是保证能够最快速迭代，推倒重来也没关系。我们的整个计算平台的架构是这样的

# 架构



# 架构

**缺点：**不能称作是大数据计算平台，只是一个包含了数据采集，数据计算脚本和数据展示的Java应用，拿目前流行的micro service化来说，这个就是一个micro service 的反例，一个monolithic的应用。

**事实上的效果：**非常适合验证产品，利用一些一站式开发框架，修改业务非常简单，MySQL的结构化特点使得计算脚本非常容易。这个架构大约支撑了我们3个月的时间



## 业务特点

1. 计算指标相对稳定，及时加指标也是基于原有的采集点的计算
2. 有一些流量大一点的种子用户进来了，数量也越来越多
3. 计算上分为了实时计算和离线计算这两种需求

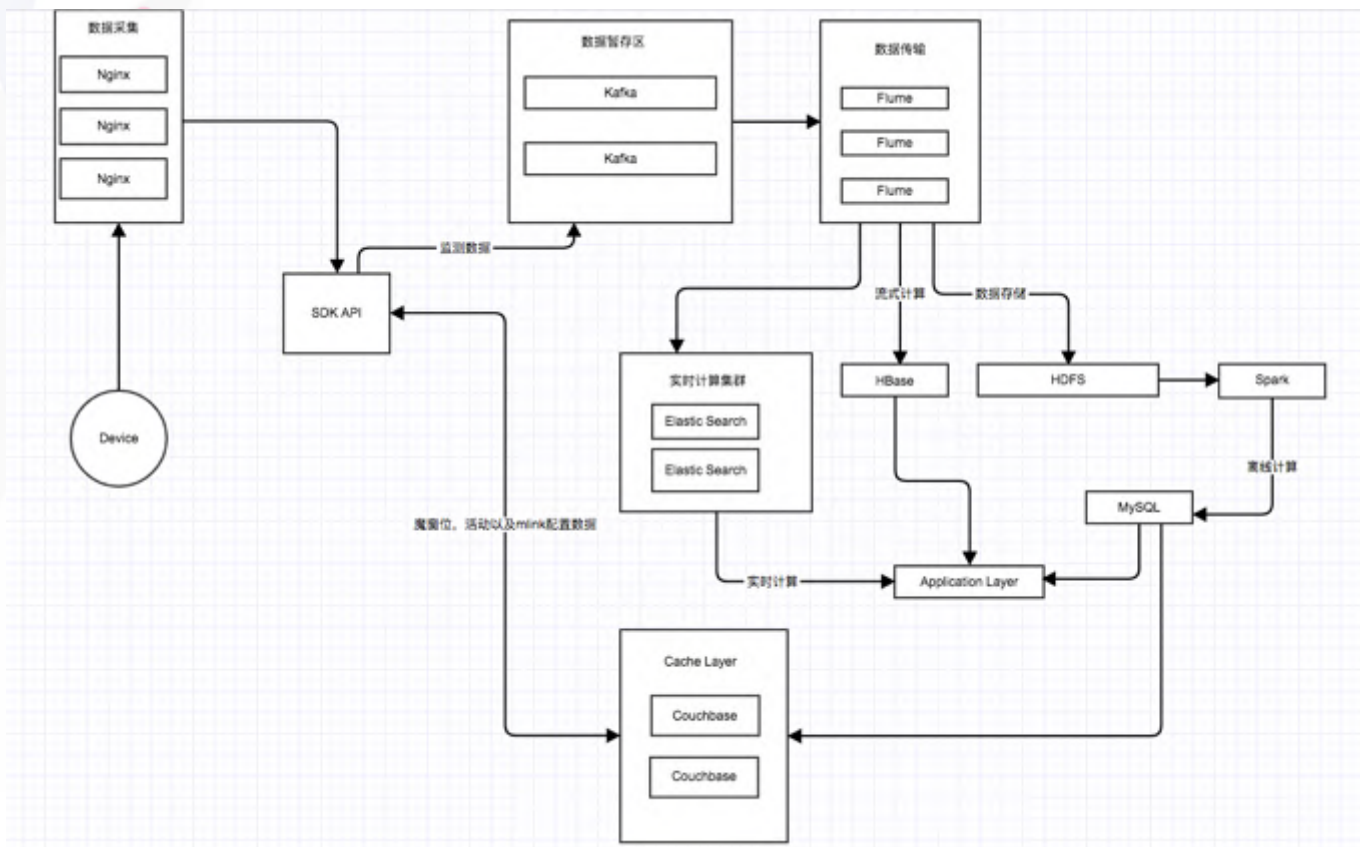
除了MySQL的计算性能问题，这个时候光是采集数据就会经常造成MySQL连接失效，于是我们在不断优化MySQL服务器端和客户端连接参数的同时，开始了真正的大数据平台的架构。这个时期的架构有一个总的原则就是可持续迭代，因为产品一旦稳定成熟，技术上就承受不了推到重来的代价了



# 架构

1. 采集端保证大吞吐量
2. 再存储和计算节点处问题的情况下，保证在一段时间内采集到的数据不会丢失
3. 性能可以通过Scale Out解决，并且易于做Scale Out
4. DevOps简单，能够方便的监测和预警

# 架构



# 数据采集

采用Nginx没有什么太大的争议，异步非阻塞，保证大吞吐量，需要进行参数调优，比如worker，keep\_alive等

## 数据暂存区

这里和一些传统的监测架构有所区别，魔窗并没有采用把Nginx的日志当数据暂存区的办法，而是直接用了Kafka，好处在于：

1. 比起磁盘IO，Kafka的吞吐量更大，并且提供了异步写入的方法，保证Nginx采集到的数据能够最及时的进入数据暂存区
2. 消息队列本身就具有分布式的一些特性，比如支持Failover保证高可用，数据可以存放多份，Partition机制是数据的写入和加载更高效，
3. 消息队列天生能解决不同种类监测数据区分的业务问题（比如Topic）
4. 比起日志，利用Kafka的API能够方便的处理一些数据续传的问题，比如如果存储节点崩溃了，仅仅利用日志是很难知道下次应该从那条记录开始续传的，Kafka就可以利用客户端保存的Offset（实际上我们每个Kafka客户端的Offset是保存在Zookeeper中的）做到

# 数据传输

当时在两种方案里摇摆，一个是Flume 还有一个是Spring XD，最终选择Flume的原因在于

1. 使用简单，有大量的source和sink可以用
2. 能被CDH托管（Spring XD不能被CDH托管，但是可以被Yarn托管）

# 离线计算

Spark + HDFS的模式相信已经被大家所熟悉，下面之谈一下魔窗对于Spark的优化心得：

1. 了解应用中的RDD的partition, 执行中的stage情况，避免过多小任务
2. 尽可能程序中复用RDD, 如果多次使用，考虑做cache, 根据实际情况选择合适的持久化策略
3. 必要时候使用broadcast 和 accumulator
4. 根据自己的作业具体情况结合系统资源监控调整主要资源类参数，例如 num-executors， executor-memory， executor-cores和spark.default.parallelism等
5. 如果允许，建议尝试官方推荐的Kryo
6. 对于jvm，通过打印GC信息了解内存使用情况，调整相应参数

# 流式计算

对于像用户留存这样的指标，根据回溯历史数据去做计算是相当困难的，采用流式计算的话会简单很多，根据魔窗的业务特点也并没有引入Storm或者Spark Stream这样的流式框架，而仅仅是在Flume传输数据的过程中，简单地利用HBase做了流式计算。



## 留存用户举例

	Unique id per device	First access time	Previous access time
	DeviceId1	Ft1	Pt1
	DeviceId2	Ft2	Pt2

	Day	Tenant key	D				W		M	
			D1	D3	D5	D9	W2	W6	M1	M3
	2016-05-11	Tenant1	1		2				2	
	2016-05-11	Tenant2				3		4	1	

## 留存用户举例

为各租户定义需要计算的留存区间，例如5日留存，7日留存，2周留存，1月留存等

例如某租户tenant1，选择配置为计算首日，5日，7日和3月留存

那么该租户所属的某个app（source App）发送的一条类似如以下的event，

tenant1|deviceId1| timeStamp1|action1，

应用会做以下操作：

1. 如果是新deviceId，则上表中新增访问记录
2. 如果不是，例如本例中的deviceId1，计算距离上次访问时间间隔（以天计）， $(timeStamp1-pt1)=2day$ ，更新上表中的previous access time
3. 通过CAS incr. 更新以下留存记录，如果跨天了，表示这个设备的用户就是留存用户。如果跨1天，表示1天留存，跨3天，表示3天留存，依次类推。这是天的留存，周的留存根据Previous access time判断是否跨周，道理相同。

# 实时计算

对于特定时间范围内的全体数据集的实时计算，选用了Elasticsearch作为实时计算的集群，原因如下：

1. 数据结构基于Json，因此是半结构化的数据，易于计算
2. 基于我们的测试，查询的response time基本能够随着节点的增长线性降低
3. 非常容易做Scale Out，非常容易通过参数设置调整数据备份和Partition的策略。
4. 支持查询的模板化，使查询和客户端代码解耦
5. 包括查询，管理在内的所有功能API化，易于运维。
6. 插件丰富支持从其他数据源双向导入数据

# 为什么选择CDH

对下列的几个Hadoop 发行版本进行过调研  
CDH, IDH (Intel) , HAWQ (Pivotal) , Hortonworks

之前已经谈到，在选型里面比较关心的是DevOps，所以需要最大限度的利用已有工具提升运维的效率，在这一方面CDH是最强的，它的管理工具提供了安装，维护，监测，预警等一系列帮助运维的功能，节省了我们维护的很多时间。

## 为什么选择CDH

IDH的特点是在HBase提供了LOB的类型，对二进制存储有帮助，使用特殊的存储类型避免发生频繁的Compacting。同时还优化了Hive计算的性能使相关数据尽量在同一region里。这几点和我们的需求毫无关系，而且Intel已经战略投资Cloudera，之后会把IDH的功能逐步移入CDH。

HAWQ，最为Pivotal HD的基础，HAWQ最大的特点是在于它实际上是一个MPP架构的数据库，提供了基于HDFS之上的SQL支持。3各Data Node的情况下，上亿级别的包含group by聚合以及SQL子查询的复杂查询响应在10秒左右。所以HAWQ非常适合异步的近实时查询，但是我们也没有这个场景。但是用HAWQ开发聚合可以把开发计算任务的成本降到0是非常具有吸引力的。

Hortonworks，各方面和CDH很像，但是管理工具不如CDH强大。

## 业务特点

随着BD的铺开，接入的客户越来越多，随着数据量的增长，产品成熟阶段设计上的许多问题暴露了出来，但是因为先前的架构原则是可持续迭代，所以问题都发生在局部的某些点上

# 问题

1. 没有用到任何序列化技术，数据存储是简单粗暴的文本格式，这样会导致两个问题
  - a) 当数据种类增加时，计算任务会产生大量join，既增加计算的复杂度，又影响性能
  - b) 计算脚本和数据格式严重耦合，脚本任务取字段依赖于该字段在文本文件中的位置，增减字段需要评估所有job的影响。
2. Flume再往HDFS写入时，无法保证一个partition一个文件，往往会被打散成许多小文件，Spark的计算性能和Namenode的性能对小文件的数量严重敏感。
3. 采用Spark Standalone，资源调度不智能，很难充分利用集群资源
4. 被CDH 托管的Flume 一台机器只能使用一个Flume 节点



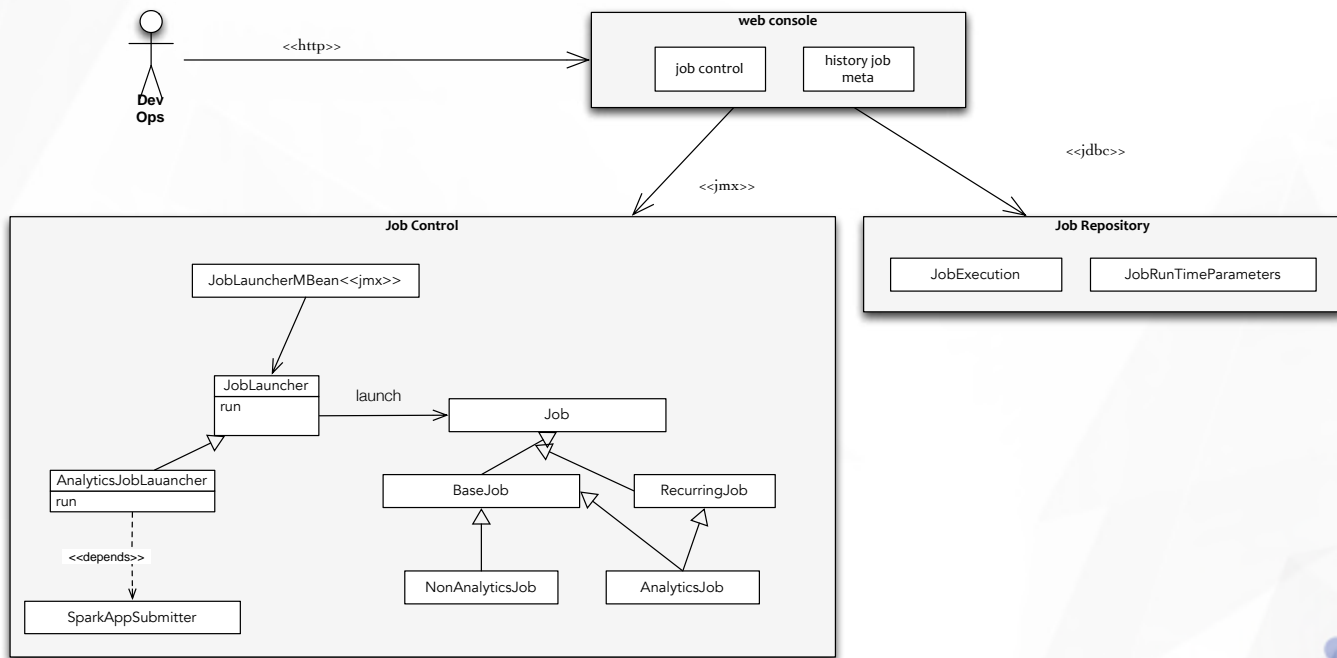
# 架构改进

针对这些问题，我们又逐步进行了一些优化

1. 录入HDFS的文件采用Arvo的格式，保证采集到的一条完整数据可以存储在同一个文件中，不用拆分，摒弃了join。另外基于Schema的数据，使得计算Job的语义更容易理解，可维护性更好。
2. 在离线计算任务之前，我们先会跑一个脚本将同一个partition下产生Flume产生的文件给合并，大大提升计算性能
3. 从Spark standalone 切换到Spark Yarn。这样做的好处在于
  - a) 统一了我们的资源调度平台
  - b) Yarn会自动优化数据的存储和计算发生在同一地域的问题（同一台服务器，同一台机柜）
  - c) 资源调度配置灵活，优化合理
4. Flume 做成了microservice，脱离CDH托管

# 计算任务框架

魔窗针对计算任务扩展开发了web console,通过JMX去控制job的基本操作,同时提供对已执行过的job的信息访问,这些数据存储在mysql中,通过Job Repository服务访问。



# 总结和心得

从我们的平台发展经历来看，在初创公司做大数据平台的选型，最重要的有两点：

1. 产品目标导向，不同的阶段利用有限的资源采集不同的架构策略
2. 无论何种架构策略，DevOps始终是架构选型的一个重要考量，因为它直接影响到你如何评估和调整架构。

# 欢迎关注魔窗



魔窗公众号  
magic-window



您的私人助理  
omic53

Deeplink更多玩法  
<http://t.cn/RcPCnQE>

魔窗详细介绍  
<http://t.cn/RcPC8Og>

官网地址  
[www.magicwindow.cn](http://www.magicwindow.cn)

关注我们公众号和官网了解更多！



增长联盟邀请函

# THANKS

SequeMedia  
盛拓传媒

IT168.com  
中国IT网

ChinaUnix

ITPUB  
www.itpub.net