实时分析平台设计

实时平台介绍

目前 DAU(Daily Activation User)/DNU(Daily New User) 等核心指标的实时统计数据可以通过两种方式查看, 1. 友盟统计 2. 自研统计。无论友盟还是自研统计都无法对维度进行细粒度拆分分析，对 DAU/DNU 的上升或者下降除了靠直觉或经验猜测外别无它法。本文描述了一个实时统计框架来统计实时数据，使用户可以从多维度、细粒度的查看实时数据。但都无法满足我们的多维度实时分析需求，本文我们会阐明目前头条实时分析平台: thor.bytedance.net 的核心设计思路，希望对大家在系统设计和实现上有所助益。该平台可以实时查看，各App 新增和活跃用户 10分钟、小时力度在各维度信息。该平台可以用于：

- 指导产品投放。产品投放尤其关注特定渠道的新增量，尤其在做广告的时候，查看实时的新增尤为重要，如果效果符合预期则可以直接加大投放而不用等到第二天
- 热点事件分析。通过该平台可以很容易查看热点事件对活跃或新增的影响
- 提前暴露问题。比如新版本发布的时候，可以实时查看活跃和新增的用户变化，提早发现问题，及时止损
- 其他多维度实时分析需求

技术选型

数据量预估

在做数据平台的时候，第一个需要考虑的就是数据量，这有助于我们技术的决策。在估算数据量的时候，我们可以从6个角度出发，我们以 活跃用户10分钟级别为例：

估算角度	说明	数据量
app 数	目前接入10 个	10
可能维度组合	DLU 需要统计10 个维度 假设最多只能同时选择三个维度	10 * 9 * 8
每个组合可能值	假设每个维度有100可能值	100 * 100 * 100
每天采点数	10分钟维度的话，每天采点144次	144
保存天数	假设保存3个月	90
类型	累积值、分时段值	2

这样可以估算出：**DLU 10 分钟级别数据，并只能同时选三个维度**的存储量 10 * 10 * 9 * 8 * 100 * 100 * 100 * 144 * 90 * 2 = 186 万亿

我们对数据集进行了压缩，做了如下规定：

- 只支持 两个维度的交叉筛选 + Os

- 每个维度只可查询Top 100 的值
- 10 分钟数据只保存8 天

这样，我们就可以将数据量保持在300 亿

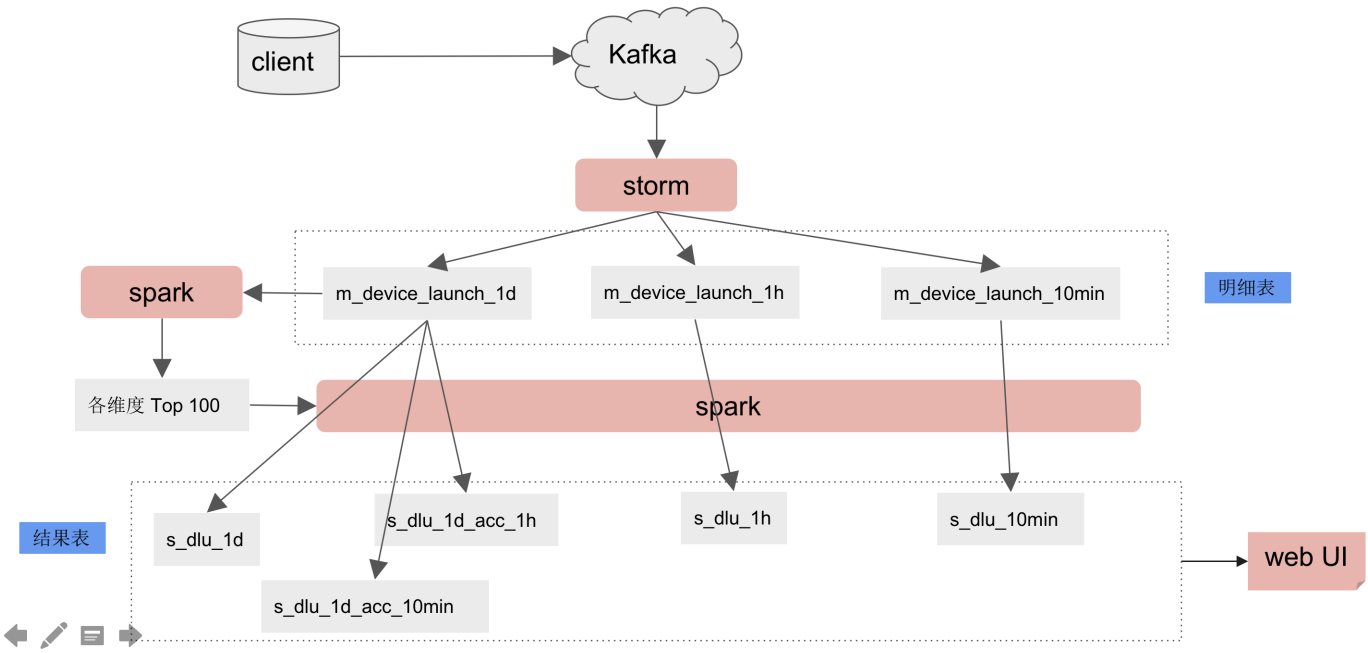
方案选择

- Druid。Druid是一款优秀的面向海量数据的、用于实时查询与分析的OLAP存储系统。通过各种基数估算算法进行大数据的快速查询与分析，但经校验数据误差较大，不予采用。具体可以参考调研报告：druid 调研报告
- CounterService 计数方式。基于每一种维度组合可能性进行设备的计数统计，但是有两个问题：1，维度组合可能太多（>=1 亿种），CounterService更适合一些简单的场景 2，数据异常也不容易排查问题
- 全量数据聚合计算。实时聚合当前全量数据，这能满足我们的系统需求。

1-min 数据流架构10-min AND 1-hour 数据流架构设计

整体设计

整体设计思路为：将设备上报信息通过storm 写入hbase 明细表，然后通过spark 进行聚合预计算出结果存储到 hbase结果表，供web 系统使用。具体系统架构图如下



具体步骤为：

- client 将数据上报到kafka实时流。
  数据的实时上报是实时系统的保证，已验证99%的数据都会在3分钟内上报到服务器
- storm 消费kafka数据，经处理后写入hbase 明细表。
  主要做了两件事情：1、维度值获取以及处理 2、将设备维度的数据明细信息依次写入到 10分钟明细表、小时明细表、天级明细表。

明细表格式example (下面会细讲)	
23#20170101#13#298789	os:android, os_version:0.98, brand:xiaomi, ...

- spark 轮询读取明细表，预计算出所有可能查询结果，并写入hbase 相关结果表

结果表格式example(下面会细讲)	
23#20170101#13#os:android#brand:xiaomi#-	dlu:19023

- web 读取hbase 结果表，并可视化展现

Q & A
Q: 为啥要建三张明细表？
A: 明细表会在特定时间窗口内对设备进行去重处理用于计算分时段数据。比如 m_device_launch_10min 会在每个10分钟区间内对device_id 进行去重处理，但是两个不同的10分钟区间可能会有重复的device_id，因此我们需要对不同的时间窗口进行不同存储
Q: 为啥要建五张结果表？
A: 五张结果表，其实区分了不同时间力度（分钟、小时、天）的累积值、分时段值。但因为天级的累计值和分时段值其实是一个意思，所以天级别只有一张表，一共5张表
Q: 为啥不只存储分钟力度的累积结果表，小时级别和天级别的累积表都从分钟级别累积结果表中聚合？
A: 是的。我们可以这么做，但是我们区分不同的时间力度，就是使得程序更加的简单，易懂，数据的获取也更加的方便和高效

hbase 表设计

hbase 明细表:

- row_key pattern 为：{salt}#{date_format}#{app_id}#{device_id}。比如：23#20170101#13#1232239823
  salt = device_id % 1000 。这样做有两个好处，1. 数据尽可能分散到hbase 的不同region server上 2. 一个device数据只会分到一个salt分区上面，便于spark 聚合计算
  date_format 对于不同的时间力度格式不同，具体如下

day	%Y%m%d
hour	%Y%m%d_%H
10min	%Y%m%d_%H%M note: M 取值只有 00, 10 到 50

- value 为：dimension_key:dimension_value, dimension_key:dimension_value。比如：brand:小米, os:Android, os_version:0.12

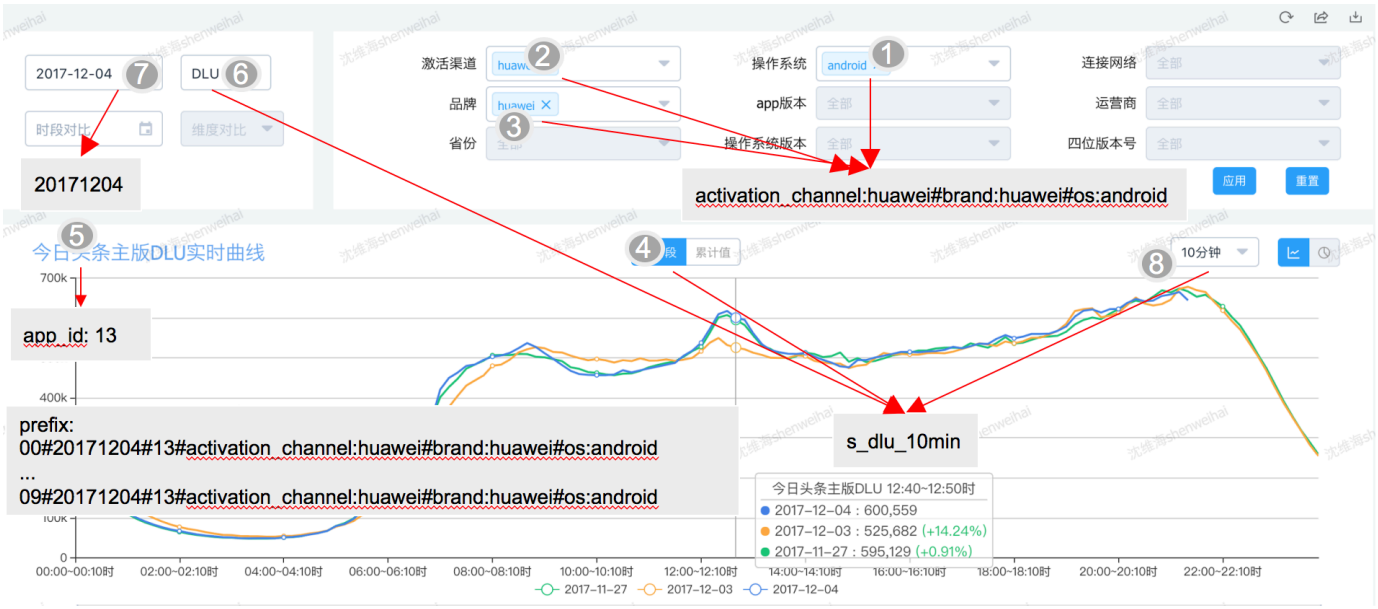
hbase 结果表：

- row_key pattern 为：{salt}#{date_format}#{app_id}#{dimension_whence_str}#{optional time}
  salt = hash(date_format + app_id + dimension_whence_str} % 10。这样可以优化查询，根据查询条件我们可以直接定位salt分区
  optional time 对于不同的时间力度格式是不同的，具体如下

day	None
hour	%H
10min	%H%M note: M 取值只有 00, 10 到 50

  dimension_whence_str 是按筛选条件进行的字符串拼接，比如: brand:xiaomi#os:iOS#-。维度的顺序是有序的
- value 为：dlu:{dlu} or dnu:{dnu}

举个例子：

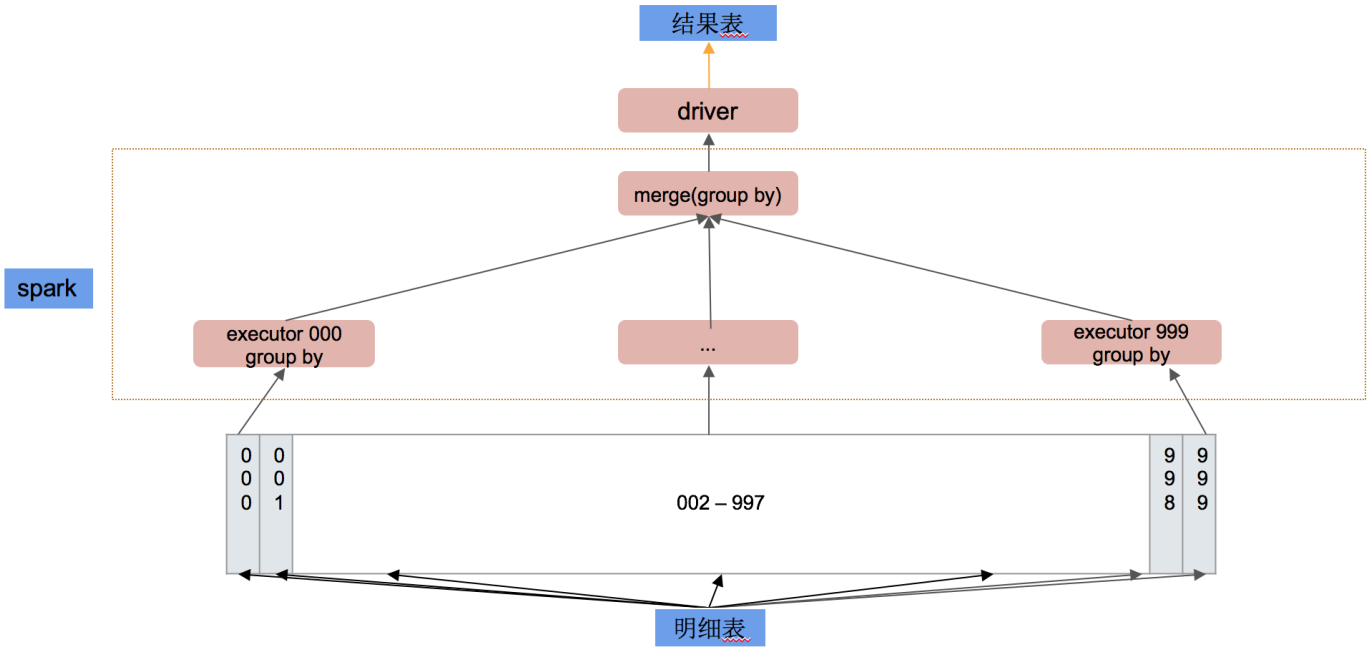


如上图通过 4、6、8 即可以确定hbase 表为 s_dlu_10min；通过1，2，3，5， 6， 7 可以拼出来row_key 的prefix 为 {salt}#00#20171204#13#activation_channel:huawei#brand:huawei#os:android。这样就可以直接获取想要的数据了

数据聚合

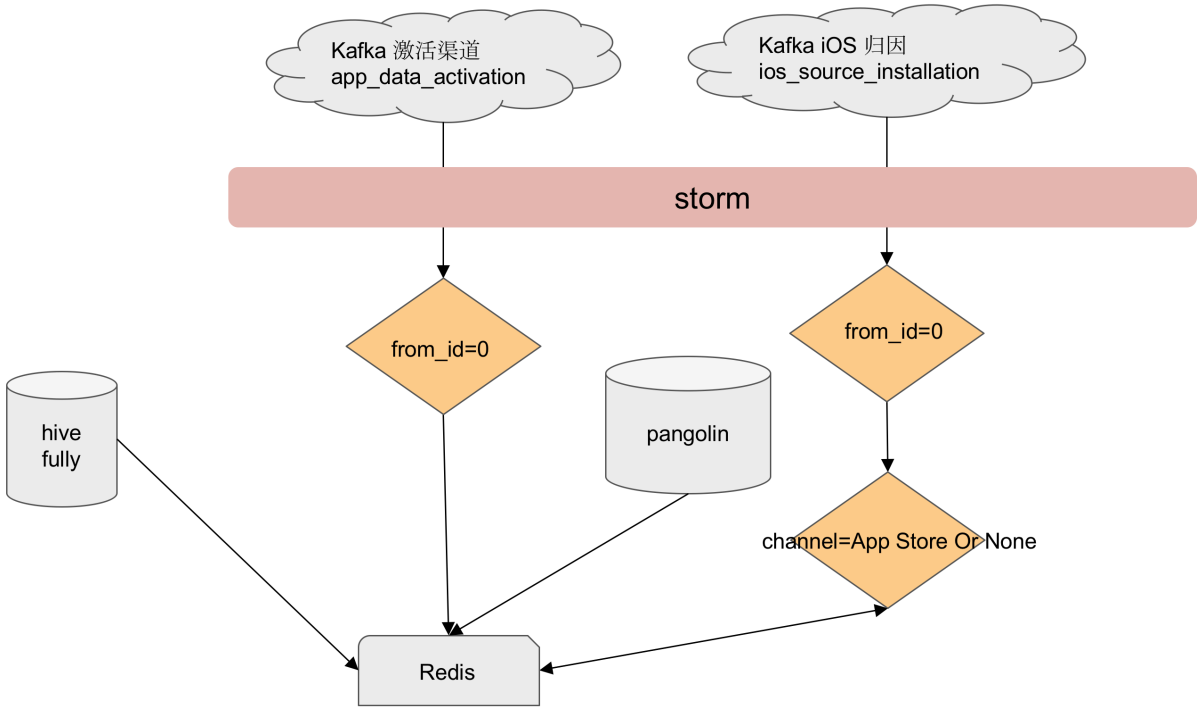
我们通过spark 进行数据的聚合统计，将明细表的数据预处理以后写入结果表，具体步骤如下图：

- spark 启动1000 executor 分别获取明细表的1000 个分区数据，在每一个executor 进行聚合
- 将所有execotor 的聚合结果进行二次聚合出最终结果，并传递给driver
- driver 将聚合数据写入到结果表



字段获取 - 激活渠道获取

首次激活渠道获取，用户上报信息中仅仅包含device_id，没有激活渠道。目前策略，维护一个全量的device_id + app_id 和 activation_channel 的redis 映射。其中，数据有两部分来源，一个是hive 里面的查出来的历史数据，一个是实时storm 数据流处理的新增的映射，具体流程如下



步骤

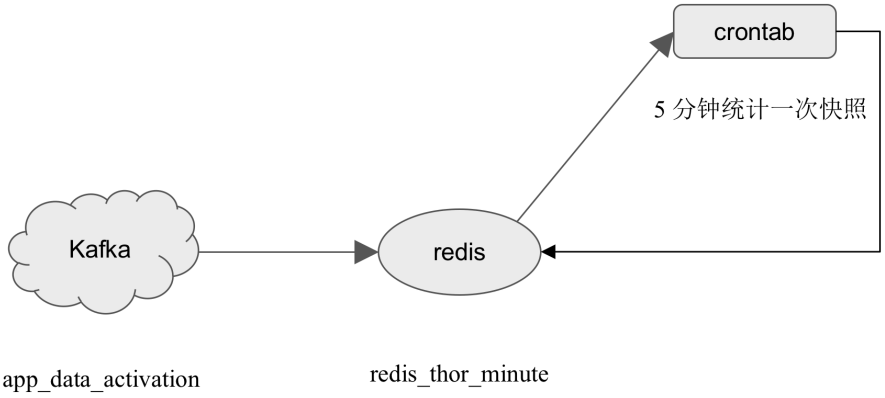
- 将 hive 历史全量数据导入redis
- 从激活渠道kafka中获取实时的映射关系也写入redis
- 从iOS 归因对channel 进行补充（因为从激活渠道kafka中来的所有iOS渠道的channel 都是App Store）

redis 使用hset 进行存储，如下

HSET		
device_id	app_id	channel_id
12232323232	13	129
	1221	192
	45	89
	...	...
12232434323	12	192
	1231	34
	23	91
	...	...

1 分钟DUN 数据流架构

我们需要对抖音和火山的一分钟DNU数据进行统计、展现，因为模型简单，因此我们直接全部通过redis 解决问题



步骤：

- 用户上报新增信息到kafka
- 通过storm 消费kafka将新增用户按 app_id + device_id = time index（将一天按一分钟力度分成1440） 进行存储，redis 存储格式如下：

HSET		
app_name#p_date	device_id	time_index
aweme#20170101	102932323	0
	102932324	132
	102932325	12
	...	...
live_stream#20170101	102932323	0
	102932324	132
	102932325	12
	...	...

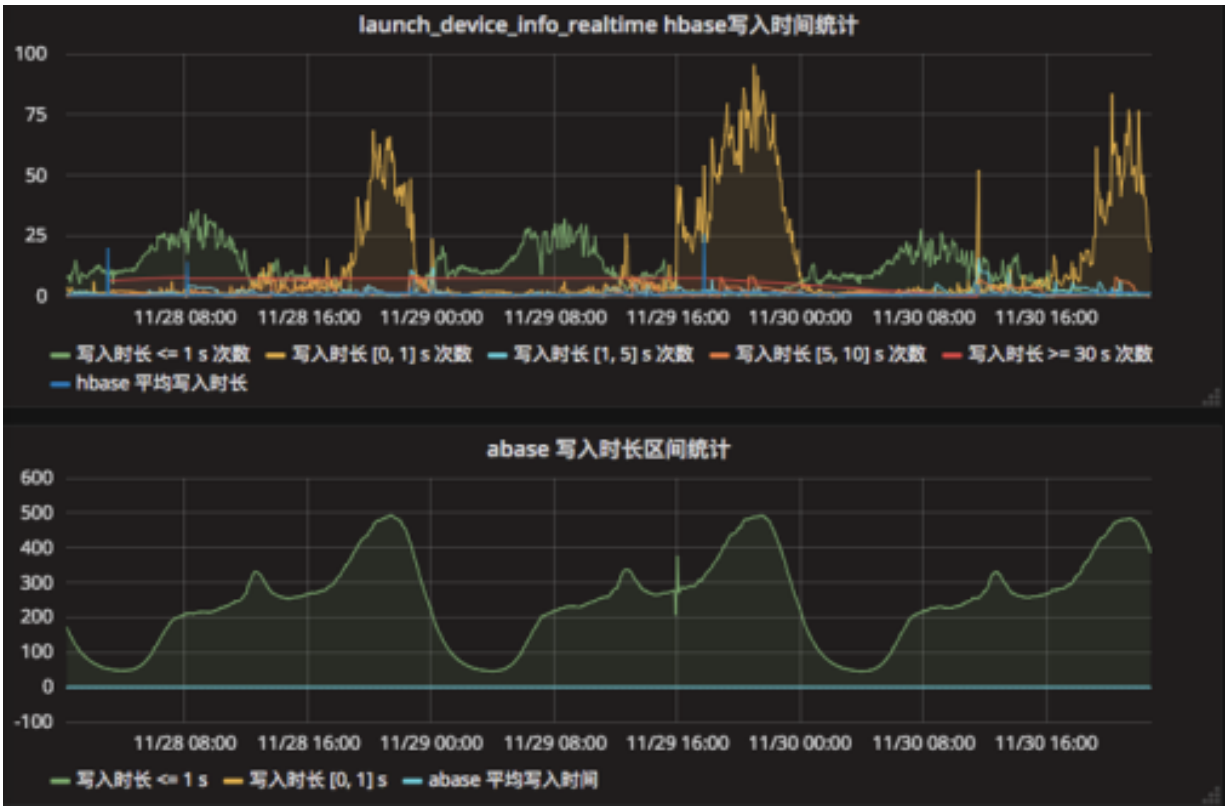
- 使用crontab 每5分钟统计统计一次数据，重新写入redis，redis 存储格式如下：

HSET		
app_name#p_date	device_id	time_index
aweme#20170101	102932323	0
	102932324	132
	102932325	12
	...	...
live_stream#20170101	102932323	0
	102932324	132
	102932325	12
	...	...

优化 / 问题

hbase 迁移到abase

由于写入hbase 是通过thrift 接口，在晚高峰的时候写入性能大大降低。使用abase 有两个好处：1. 平均每1000条写入时间超级快，对比可以参看下图 2. 资源消耗小（目前看来hbase + thrift 的机器消耗 是abase 的两倍以上）



hbase 读写集群分离

现在hbase 写的带宽可以达到 200 MBps，spark 读的带宽可以达到 1.5GBps，如下图。因此导致web service 读取数据的时候特别慢，因此我们进行了集群拆分，hbase 明细表写（带宽大） + hbase 明细表 读（带宽大） => hbase-growth 集群；hbase 结果表写 + web service 读 => hbase-growth-service 集群；效果明显

数据延迟怎么办？

因为我们的数据存在上报延迟，所以会出现一种情况就是我们现在统计的数据相比一个小时以后统计的数据可能会偏低一点。为了解决该问题，我们在统计数据的时候，除了统计最近10分钟 or 1 小时的数据以外还会重新统计已经计算的数据，以10分钟分时段数据为例，我们除了会统计最近10分钟的数据以外，还会统计最近20min和最近60分钟的数据，如下图

60 min	50 min	40 min	30 min	20 min	10 min

其他一些优化

- 提供白名单服务，可以直接添加需要特别关注但不在top 100 的维度值
- 维度值top 100 对 app_version 进行了特殊处理，区分操作系统
- 通过redis 实现参数的可配置