第二章 日志采集

一、浏览器的页面日志采集

浏览器的页面型产品/服务的日志采集可分为两大类：

- 页面浏览（展现）日志采集
  - 指一个页面被浏览器加载呈现时采集的日志
  - 此类日志是最基础的互联网日志
  - 此类日志是目前所有互联网产品的两大基本指标（**页面浏览量**（Page View，PV）和**访客数**（Unique Visitors，UV））的统计基础
- 页面交互日志采集
  - 用户操作记录

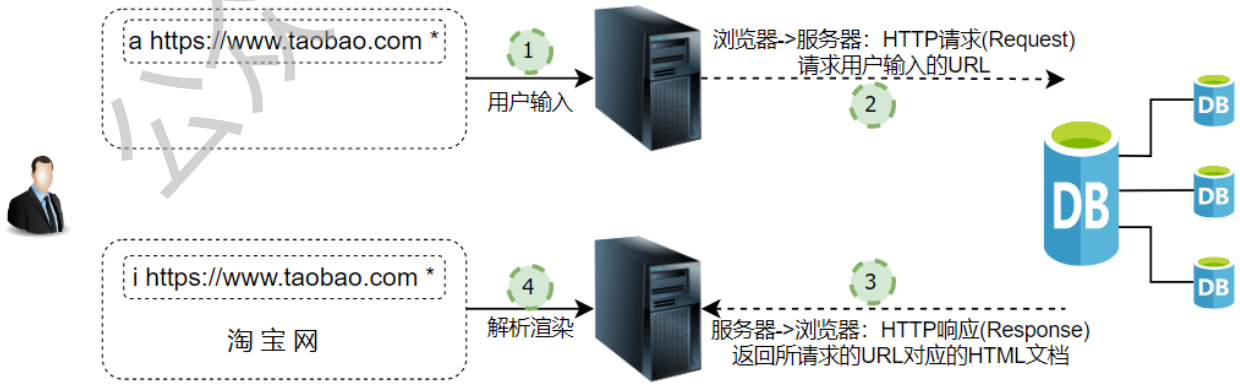
1.1 页面浏览日志采集流程

主要是为了对指标网页浏览量（PV）和访客数（UV）进行分析。

采集页面被浏览器加载展现的记录，这是最原始的互联网日志采集需求，也是一切互联网数据分析得以展开的基 础和前提。

一次典型的页面访问（请求-响应）过程：

以用户访问淘宝首页为例



1. 用户在在地址栏中输入网址并回车
2. 浏览器向服务器发起HTTP请求
3. 服务器接收并解析请求
4. 浏览器接收到服务器的响应内容，并将其按照文档规范展现给用户，从而完成一次请求

在上述网页浏览过程中，如果我们需要记录这次浏览行为，日志采集则是在这四个步骤中的某一环节。

第一二步中，用户的请求尚未抵达服务器。直到第三步完成，我们也只能认为服务器处理了请求，不能保证浏览器能够正确地解析和渲染页面，不能确保用户已确实打开页面，因此在前三步是无法采集用户的浏览日志的。所以，**采集日志的动作是发生在第四步**，也就是浏览器开始解析文档时才能进行。

日志采集思路：

- 在HTML文档内的适当位置增加一个日志采集节点，当浏览器解析到这个节点时，将自动触发一个特定HTTP请求到日志采集服务器。
- 当日志采集服务器接收到这个请求时，就可以确定浏览器已经成功地接收和打开了页面。

页面浏览日志采集的主要过程如下：

采集：客户端日志采集。HTML文档内植入JavaScript脚本进行采集。

- 采集脚本被浏览器加载解析后执行，在执行时采集当前页面参数、浏览行为的上下文信息（如读取用户访问当前页面时的上一步页面）以及一些运行环境信息（如当前的浏览器和分辨率等）
- HTML文档内植入日志采集脚本的动作可以由业务服务器在响应业务请求时动态执行，也可以在开发页面时由开发人员手动植入。

发送：客户端日志发送。日志采集完成后会立即发送或延迟发送。采集到的日志信息一般以URL参数形式放在HTTP日志请求的请求行内。

- 采集脚本执行时，会向日志服务器发起日志请求，以将采集到的数据发送到日志服务器。

收集：服务器端日志收集。由日志服务器完成对日志的收集。

- 日志服务器接收到客户端请求后，会立即向浏览器发回一个请求成功的相应。
- 日志收集模块会将日志请求内容写入一个日志缓冲区内，完成此条浏览日志的收集。

存档：服务器端日志解析存档。日志服务器对浏览日志解析并存档。

1.2 页面交互日志采集

对**用户在访问某个页面时具体的互动行为特征**（交互日志）的采集，比如鼠标或输入焦点的移动变化（代表用户关注内容的变化）、对某些页面交互的反应（可借此判断用户是否对某些页面元素发生认知困难）等。常规的PV日志采集方法无法完成对上述操作日志的采集，因为这些行为往往并不触发浏览器加载新页面。

交互日志的采集是以技术服务的形式呈现的。

采集的交互日志发送到日志服务器需经过以下步骤：

- 注册业务等的具体交互采集点，注册完成后，系统将生成与之对应的交互日志来集代码模板
- 业务方将交互日志采集代码植入目标页面，并将采集代码与需监测的交互行为做绑定
- 当用户在页面上产生指定行为时，采集代码和正常的业务互动代码起被触发和执行
- 采集代码在采集动作完成后将对应的日志通过HTTP协议发送到日志服务器，日志服务对数据进行转储

1.3 日志的清晰和预处理

在大部分场合下，经过上述解析处理之后的日志并不直接提供给下游使用。一般还需要进行相应的离线预处理。具体操作如下：

- 识别流量攻击、网络爬虫和流量作弊（虚假流量）
    - 对所采集的日志进行合法性校验，依托算法识别非正常的流量并归纳出对应的过滤规则集加以滤除。
- 数据缺项补正

    - 对日志中的一些公用且重要的数据项做取值归 、标准化处理或反向补正（即根据新日志对稍早收集的日志中的个别数据项做回补或修订（例如，在用户登录后，对登录前页面日志做身份信息的回补））。
- 无效数据剔除
    - 剔除无意义、已经失效或者冗余的数据项。
- 日志隔离分发
    - 考虑到安全性，某些日志在进入公共数据环境之前需要做隔离。

二、无线客户端的日志采集

采集思路：无线客户端的日志采集采用采集SDK来完成。

移动端的数据采集目的：

- 服务于开发者，协助开发者分析各类设备信息
- 帮助开发者了解用户、用户行为，不断对APP进行优化，提升用户体验

与浏览器日志采集的不同之处：移动端的日志采集根据不同的用户行为分成不同的事件，“事件”为无线客户端日志行为的最小单位。

- 区分不同事件的日志触发时机、日志内容和实现方式的差异
- 更好地完成数据分析

在阿里巴巴内部，多使用名为UserTrack（UT）来进行无线客户端的日志采集。基于常规的分析， UserTrack（UT）把事件分成了几类，常用的包括**页面事件**（同前述的页面浏览）和**控件点击事件**（同前述的页面交互）等。

2.1 页面事件

每条页面事件日志记录三类信息：

- 设备及用户的基本信息。
- 被访问页面的信息。主要是一些业务参数（如商品详情页的商品 ID 、所属的店铺等）
- 访问基本路径。如页面来源、来源的来源，用于还原用户完整的访问行为。

对于页面事件，UT提供了页面事件的无痕埋点， 即无须开发者进行任何编码即可实现。

UT提供了两个接口，分别在页面展现和页面退出时调用。除此之外，还有页面扩展信息的接口，该接口必须在使用页面展现和页面退出方法的前提下使用。

UT还提供了透传参数功能。所谓透传参数，即把当前页面的某些信息，传递到下一个页面甚至下下一个页面的日志中。

2.2 控件点击及其他事件

1、控件点击事件

记录了基本的设备信息、用户信息。

记录了控件所在页面名称、控件名称、控件的业务参数等。

控件点击事件的逻辑比页面事件要简单得多，就是操作页面上的某个控件，因此只需把相关基础信息告诉采集 即可。

2、其他事件

用户可以根据业务场景需求，使用自定义事件来采集相关信息。

UT 提供了一个自定义埋点类，其包括：

- 事件名称
- 事件时长
- 事件所携带的属性
- 事件对应的页面

2.3 特殊场景

为了平衡日志大小，减小流量消耗、采集服务器压力、网络传输压力等，采集 SDK 提供了聚合功能，对某些场景如曝光或一些性能技术类日志，我们提倡在客户端对这类日志进行适当聚合，以减少对日志采集服务器端的请求，适当减小日志大小。

总体思路就是每个曝光的元素一般都属于一个页面，利用页面的生命周期来实现适当的聚合及确定发送时机。

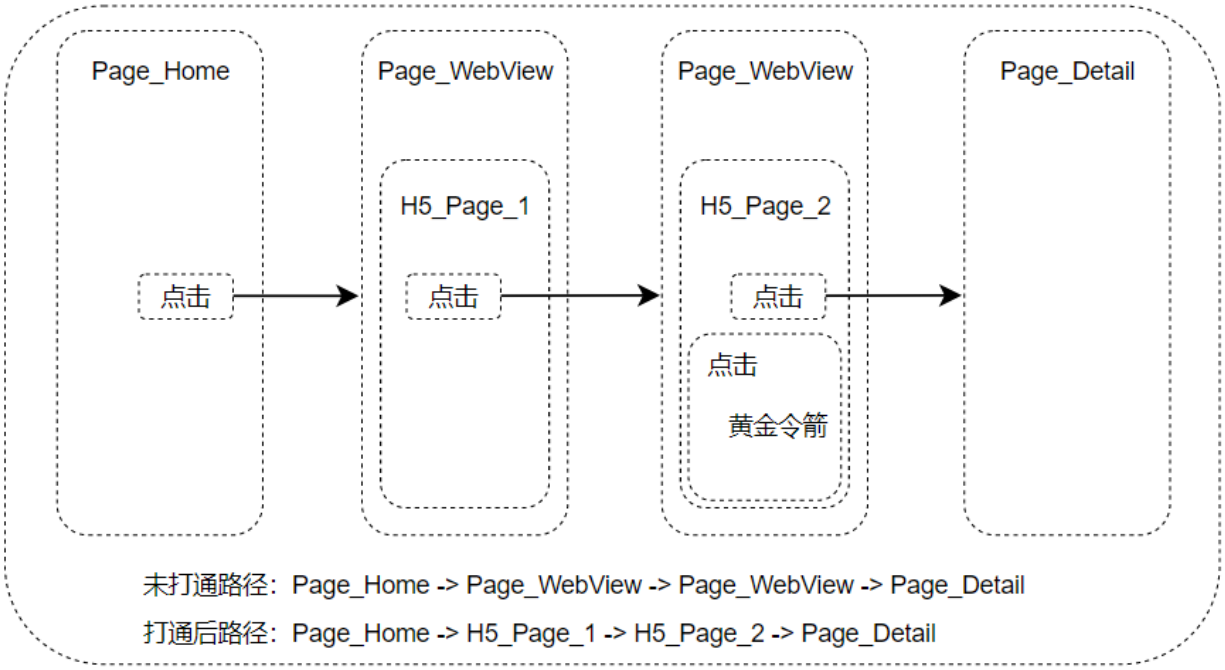
2.4 H5 & Native 日志统一

APP 分为两种：

- 纯Native APP
- 既有Native，又有H5页面嵌入的APP，即Hybrid APP

Native页面采用采集SDK进行日志采集，H5页面一般采用基于浏览器的页面日志采集方式进行采集。

考虑到后续日志数据处理的便捷性、计算成本、数据的合理性及准确性，需要对Native H5日志进行统一处理。



阿里选择Native部署采集SDK的方式，将H5日志归到Native日志。原因如下：

- SDK可以采集到更多的设备相关数据，这在移动端的数据分析中尤为重要
- 采集SDK处理日志，会先在本地缓存，而后借机上传，在网络状况不佳时延迟上报，保证数据不丢失

2.5 设备标识

互联网产品的两大基本指标：页面浏览量（Page View，PV）和访客数（Unique Visitors，UV）。

关于UV，对于登录用户，可以使用用户ID来进行唯一标识。PC一般使用Cookie信息来作为设备的唯一信息。对于APP来说，则需要选的设备唯一信息，不同版本的系统选择的标识不一样。

2.6 日志传输

无线客户端日志上传、压缩及传输。

此时**日志不是产生一条就上传一条，而是日志产生后，先存储在客户端本地，然后再伺机上传。**

伺机：需要有数据分析的支持，如在启动后、使用过程中、切换到后台时这些场景下分别多久触发一次上传动作。

- 日志切分维度为天，当天接收的日志存储到当天的日志文件中。
- 为了后续数据处理，以及特殊时期不同日志的保障级别，还根据应用及事件类型对每日高达数千亿的日志进行了分流。

日志采集完成后，日志采集服务器的日志怎么给到下游呢?

阿里主要使用消息队里（TimeTunel，TT）来实现从日志采集服务器到数据计算的MaxCompute。

TT将消息收集功能部署到日志采集服务器上进行消息的收集，便于后续计算。

- 实时计算：应用实时来订阅TT收集到的消息，进行实时计算。
- 离线计算：离线的应用定时来获取消息，完成离线计算。

三、日志采集的挑战

各类采集方案提供者所面临的主要挑战已不是日志采集技术本身，而是如何实现日志数据的结构化和规范化组织，实现更为高效的下游统计计算，提供符合业务特性的数据展现，以及为算法提供更便捷、灵活的支持等方面。

3.1 阿里典型场景及解决方案

1、日志分流及定制处理

大型互联网网站的日志类型和日志规模都呈现出高速增长的态势，而且往往会出现短时间的流量热点爆发。这一特点使得在日志服务器端采用集中统一的解析处理方案变得不可能，其要求**在日志解析和处理过程中必须考虑业务分流**（相互之间不应存在明显的影响，爆发热点不应干扰定常业务日志的处理）、**日志优先级控制，以及根据业务特点实现定制处理。**

为了避免资源浪费（尽可能多地进行预处理）和需求覆盖不全（仅对最重要的内容进行预处理）两者取舍问题。阿里互联网日志采集体系的基本原则是分治策略。将分类任务前置到客户端，**尽可能靠前地布置路由差异**，就可以**尽可能早地进行分流**，降低日志处理过程中的分支判断消耗，并作为后续的计算资源调配的前提，提高资源利用效率。

阿里的客户端日志采集代码的一个突出特点是**实现了非常高的更新频次**（业界大多以季度乃至年为单位更新代码，阿里则是以周 / 月为单位），**并实现了更新的配置** 。

2、采集与计算一体化设计

以PV日志为例，归类与汇总是页面PV日志采集之后的一个基础性操作。

早期时，是以URL路径，继而以URL（正则）规则集为依托来进行日志分类的。网站规模小时，这种方式没问题，但是当网站规模大了，这种方式就不现实了，失控的大规模正则适配甚至会将日志计算硬件集群彻底榨干。

阿里对采集与计算进行一体化设计，给出了两套日志规范和与之对应的元数据中心。

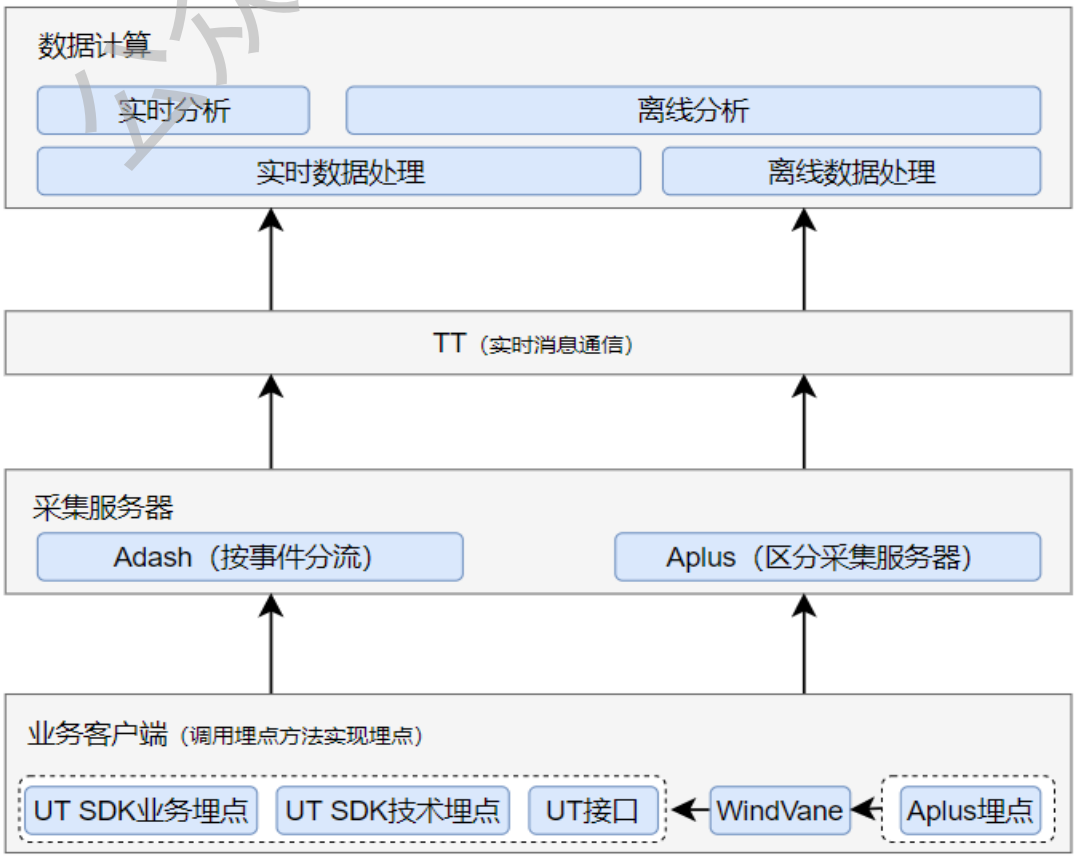
- PV日志的解决方案是目前用户可直观感知的SPM规范（例如，在页面的URL内可以看见spm参数）和SPM元数据中心。
    - 通过SPM的注册和简单部署（仅需要在页面文件内声明一个或多个标签），用户即可将任意的页面流量进行聚类，不需要进行任 多余的配置。
- 自定义日志的解决方案则是黄金令箭（Goldlog）/APP端的点击或其他日志规范及其配置中心。
    - 通过注册一个与所在页面完全独立的令箭实体/控件实体，用户可以一键获得对应的埋点代码，并自动获得实时统计数据和与之对应的可视化视图。

日志本身不是日志采集的目的，服务于基于日志的后续应用，才是日志采集正确的着眼点。

3.2 大促保障

这里以阿里“双11"为例。

数据处理全链路如下图：



1. 端上实现服务器端推送配置到客户端，且做到高到达率；
2. 对日志进行分流；
3. 在实时处理方面，不断优化以提高应用的吞吐量；
4. 实时处理方面，评估峰值数据量，在高峰期通过服务器端推送配置的方式对非重要日志进行适当限流，错峰后逐步恢复。