


# 端体验优化之全链路监控

技术漫谈 2022-03-11 00:05

以下文章来源于GPE大前端，作者白卿



GPE大前端

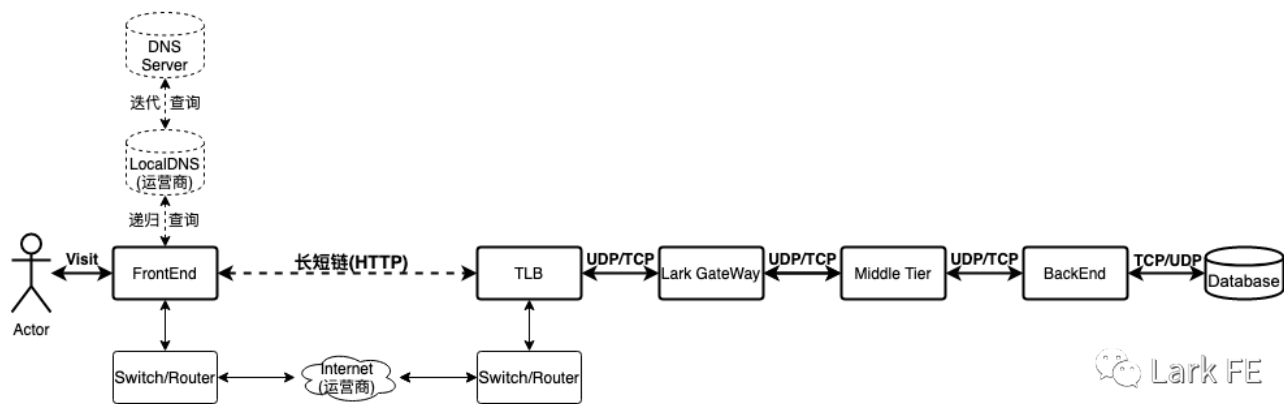
暂无描述

>

全链路监控近年来一直是一个比较热门的议题，纵观行业里面的一些分享议题以及各大云厂商在这里的建设和投入也可以看到，全链路监控是必争之地。也间接的说明全链路监控对业务的重要性，而各Saas能力又无法真正的完全占领这个市场，也说明这里存在的不少挑战，特别是如何围绕业务场景打造高效、便捷的全链路监控尤为重要。行业中全链路监控平台也比较多，有涵盖比较全面的监控平台，也有聚焦端监控的平台，也有做单一场景性能监控的。其全链路监控的思路出发点来自《Dapper，大规模分布式系统的跟踪系统——Google生产环境下的分布式跟踪系统》，我们把这个系统的概念进一步扩大到涵盖端的整个链路，更多的从全链路的视角来看，基于整个链路把监控链接起来，从而提供更为便捷、高效的监控、问题定位平台。

## 概念

那么，全链路监控涵盖哪些内容呢？我们关注的点有哪些？首先，从狭义上来讲，全链路监控就是我们常提到的APM(Application Performance Management)，对应用程序性能和可用性进行监控并告警，聚焦在性能和可用性上。从广义上来看全链路监控包含的内容会比较多，涵盖如下的整个链路。



从整个链路图可以看到，我们监控的维度需要涵盖整个链路，链路的各个环节的监控能够有效的串联起来，就是我们期望达到的全链路监控的目标。其中包含端监控、网络监控、接入层监控、中间层调用监控、服务监控、数据库监控等。

## 监控涵盖

### 端监控

端的监控内容会比较多，首先这里会去细分不同的端（区分的原因是不同端的上报和监控也有所差异），基于细分的不同端的情况，最终的目标是一致的，提升端应用的可用性和易用性（这一点在其他系列文章《端体验优化之保证多端体验-AC原则》（即将发布，敬请期待）等中也有提到）。这里我们从另外一个技术栈的角度去看端监控应该包含哪些内容。

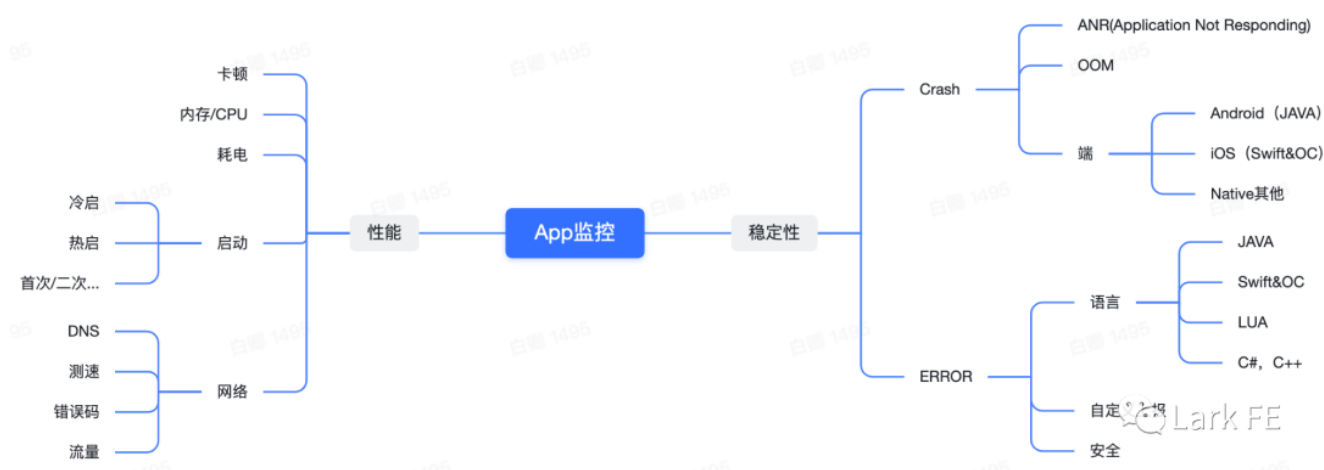
### 前端监控

错误	性能	首屏渲染	首屏加载&交互	Performance(API)
错误监控	请求监控 & 错误码	FP (First Paint) 首次绘制	DCL (DomContentLoaded)	Performance Timeline API
用户日志	资源监控 & 测速	FCP (First Contentful Paint) 首次内容绘制	L (onLoad)	Navigation Timing API
自定义上报	数据请求测速	LCP (Largest Contentful Paint) 最大内容渲染	TTI (Time to Interactive) 可交互时间	User Timing API
离线日志	首屏 & 白屏	FMP(First Meaningful Paint) 首次有效绘制	TBT (Total Blocking Time) 页面阻塞总时长	Resource Timing API
广告				

自定义测试	卡顿监控	CLS (Cumulative Layout Shift) 累积布局偏移	FID (First Input Delay) 首次输入延迟	Paint Timing API
			SI (Speed Index)	Lark FE

上面枚举到的是我们在做前端监控的时候需要关注到的监控点。随着本身对体验的要求，这里的监控的点也会迭代更新，我们在做监控的时候，也需要关注到技术本身的迭代，如何更好做到一些指标/维度的监控，都需要涵盖进来。

## 客户端（App）监控



客户端监控主要会从**性能和稳定性**去监控（欢迎补充），详细分类来看，我们会比较关注在几个点的监控，卡顿、CUP、内存、耗电、启动、测速、crash。这些是直接关注到用户使用App的体验以及本身使用过程中对设备的影响。

## 链路监控

### 网络监控

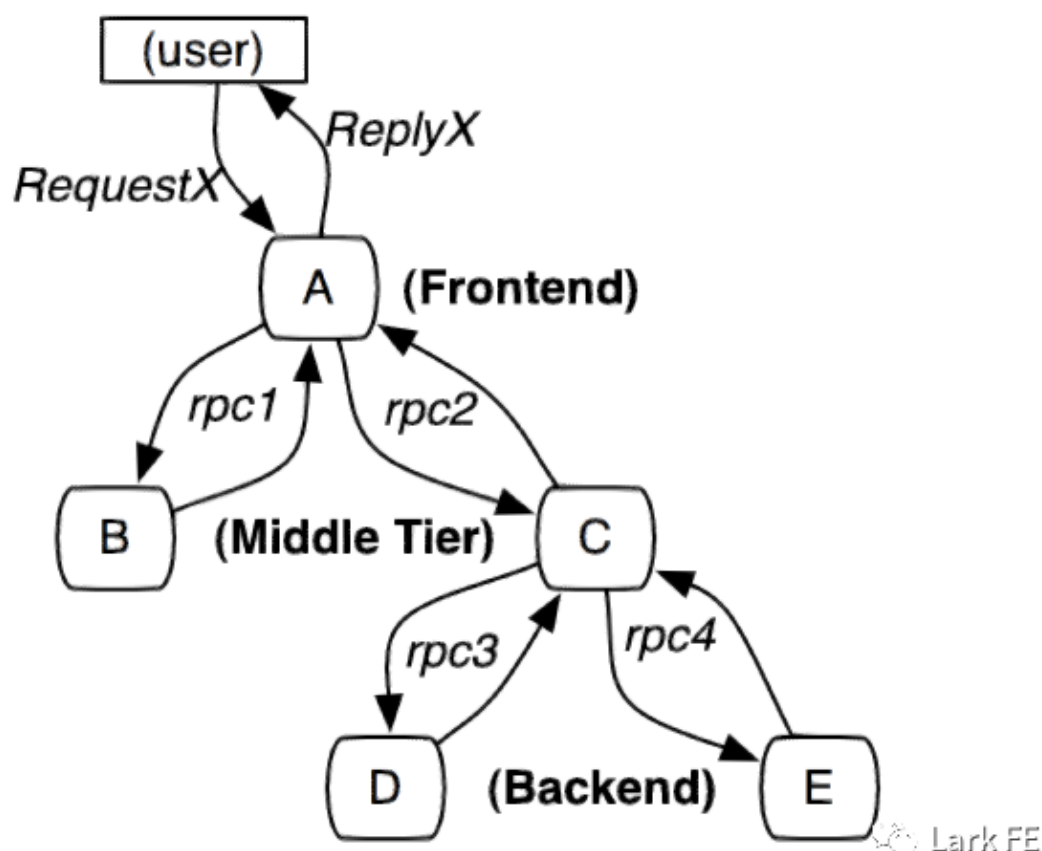
网络监控一般基于**端的访问结果**来监控来体现，因此在端上的监控中网络的监控占比很重。从另外的角度，我们也可以及时感知到我们的网络状况，对于一些网络产生的事故业务能够第一时间获悉。

### 接入层监控

接入层即网关，从上面的链路调用流程来看，接入层更多的走UDP/TCP调用，对于这一层的监控更多我们会在**网关进行流量/数据包、机器稳定性的指标（SLA）监控**。基于SLA的指标，我们可以更好的看到所有网络流量以及接入层机器负载情况。

## 链路调用监控

（直接复用前面引言中提到的分布式系统中的链路调用模型图）



在分布式微服务架构中，一个客户端的请求，需要由多个微服务应用协同处理并返回。而一个用户请求产生的链路上的多个微服务应用之间的顺序调用关系，就产生了整个链路的概念。对于链路的监控必要性也比较明显

- 全局调用关系，明确上下游依赖
- 每个应用的SLA指标和状态
- 故障/异常的及时发现、定位、解决

而在引言中，提到的Google分布式链路跟踪中（Dapper），如何在超大规模系统上建设低损耗、应用透明的、大范围部署这几个核心需求成为链路追踪中关注的核心内容。同时，Dapper也进一步描述了链路追踪的技术细节，包含表示、埋点、传递、收集、存储、展示等。同时，跟踪树和Span、Trace、Annotation(标注)等一些链路追踪的核心概念也被用于各全链路监控系统中。

目前业内也有比较多的链路追踪的开源组件，这些开源组件基本都遵循一个统一的标准，就是基于Dapper启发下而抽象出来的OpenTracing标准(<https://opentracing.io/>)。目前链路调用监控的实现上也有两种方式。

- SDK：引入链路追踪SDK自动生成链路数据，并自动上报。缺点：需要做好框架适配。如：Jaeger。
- 探针：应用运行的过程中，通过一个Agent动态的拦截底层框架的行为，从而自动注入监控逻辑。缺点：不是每个语言都具备动态拦截能力。如：Skywalking。

## 数据库监控

数据库的监控的本质目的是要保证数据的可用性，数据本身是比较核心的资产，因此本身数据库监控在其他服务监控要求的基础上需要更为完善和严谨的监控。**从软件到硬件，涵盖服务、CPU、内存、CPU温度、磁盘等**。业内也有一些比较好用的数据库监控工具，可以参照看看。

## 关键功能

前面重点阐述了监控内容的涵盖，那么除了本身要监控的内容之外，作为全链路监控还需要包含些关键的功能呢？从监控的目标来看，我们需要**更智能的发现问题，更便捷的定位问题，快速辅助解决问题，同时，能够让端上的监控上报更为自动化减少侵入式监控打点**。

## 自动上报

监控在端上需要做到尽可能的自动化上报，而不是需要侵入式的上报。目前端上很多API也支持自动监控，因此一个优秀的全链路监控，在上报这里需要释放开发，引入SDK或者探针的方式在Runtime阶段实现监控上报。

## 更多维度

监控维度的丰富度，直接影响是否可以快速定位分析问题，同时更丰富的维度可以让我们的监控粒度更细粒度，这样便于我们发现诸如地域、运营商、租户、设备(诸如DeviceID、IMEI、IDFA、UDID和UUID)等非常垂直场景的质量情况，可以针对性的分析和定位。

## 实时告警

监控最重要的一个内容就是能够实时告警给业务，这样能及时感知到业务的一些异常状态，便于主动发现问题。实时告警分级别可以进行不同程度的告警IM、短信、电话，对于一些高优

的问题，实时加急。

## 智能分析

智能分析是配合前面的实时告警，所谓智能分析是能够真正发现问题，而不是一些正常的波动被误报，误报会极大的影响有效信息的获取。智能分析除了能够更准确的识别问题之外，也需要基于前面提到的维度可以把更细维度的问题发现出来。

## 宏观统计

平台提供基于维度的完善的报表能力，可以在多个维度上进行宏观观测。这些报表和统计也是作为业务进行优化的标准对齐。基于完善的报表统计，可以进一步和行业同类产品对齐标准和口径。

## 详细链路日志

除了宏观的统计报表之外，详细错误定位上能给予更快速、便捷的查找能力，能够快速搜索到需要的日志。同时，日志上是基于全链路的完整日志，而不是阶段性的日志，这是评判全链路监控平台区别于普通监控平台的一个重要功能点。基于全链路的日志可以知道用户在整个链路访问过程中，具体在哪一个环节miss掉了，直击问题环节。

## 总结

全链路监控作为我们主动发现、定位分析、辅助解决的重要工具，需要被重点建设起来。我们在一些关键场合的性能优化分析都是基于具体的数据来去优化，而这些数据都需要监控工具来辅助输出。因此做好全链路监控对于业务的迭代演进、优化至关重要，这也是未来我们需要重点建设和关注的点（性能实验室）。

喜欢此内容的人还喜欢

ChatGPT当然会迅速消失，但一切才刚刚开始 1/2

加州AI高学长



关于主页解封

小魏的奇妙物语



## 第二节：传感器的认识

爱橙创客编程

