

24 | 如何利用监控保障发布质量？

2018-08-28 王潇俊

持续交付36讲

[进入课程 >](#)



讲述：王潇俊

时长 14:19 大小 6.56M



你好，我是王潇俊，今天我和你分析的主题是：如何利用监控保障发布质量。

在前几次的分享中，我详细介绍了发布在持续交付过程中的重要地位，以及如何去思考和设计一套灰度发布系统。作为发布及监控系列的最后一篇文章，今天我就和你聊聊灰度发布的最后一个过程：监控，以及如何做好发布后的监控。

之所以有今天这次分享，最重要的原因是要告诉你：**千万不要认为发布结束，就万事大吉了。特别是生产发布，发布结束时才是最危险的时刻。** 因为，故障都是伴随着发布变更而来的。所以，我们需要有一套监控系统，及时发现问题、定位问题，帮助我们减少因故障带来的损失。

同时，随着分布式系统的普及，以及 APM（Application Performance Management，系统性能管理）概念的兴起，针对分布式系统的全链路监控系统也逐步发展起来，为持续交付提供了有力的支持。可以说，一套性能优良的监控系统，可以为持续交付保驾护航。

当然，这个专栏的主要内容是帮你解决持续交付的问题，所以我不会去分享监控系统如何设计这种需要一整个专栏才能解决的问题。

因此，我今天分享的重点是，帮助你理解监控的常规概念，和你聊一些技术选型方案，并一起讨论一些与持续交付有关的问题。

监控的分类

从一般意义上来讲，我们会把监控分为以下几类：

1. 用户侧监控，关注的是用户真正感受到的访问速度和结果；
2. 网络监控，即 CDN 与核心网络的监控；
3. 业务监控，关注的是核心业务指标的波动；
4. 应用监控，即服务调用链的监控；
5. 系统监控，即基础设施、虚拟机及操作系统的监控。

因此，我们要做好一个监控系统，可以从这五个层面去考虑，将这五个层面整合就可以做成一个完整的、端到端的全链路监控系统。当然，监控系统的这 5 个层次的目标和实现方法有所不同，接下来我将分别进行介绍。

第一，用户侧监控

随着移动互联网的兴起，用户对 Mobile App 的体验成了衡量一个系统的重要指标，所以对用户侧的监控也就变得尤为重要。因为，它能够第一时间向我们反馈用户使用系统的直观感受。

用户侧的监控通常从以下几个维度进行，这些监控数据既可以通过打点的方式，也可以通过定期回收日志的方式收集。

1. 端到端的监控，主要包括包括一些访问量、访问成功率、响应时间、发包回包时间等等监控指标。同时，我们可以从不同维度定义这些指标，比如：地区、运营商、App 版

本、返回码、网络类型等等。因此，通过这些指标，我们就可以获得用户全方位的感受。

2. 移动端的日志。我们除了关注系统运行的日志外，还会关注系统崩溃或系统异常类的日志，以求第一时间监控到系统故障。
3. 设备表现监控，主要指对 CPU、内存、温度等的监控，以及一些页面级的卡顿或白屏现象；或者是直接的堆栈分析等。
4. 唯一用户 ID 的监控。除了以上三种全局的监控维度外，用户侧的监控一定要具备针对唯一用户 ID 的监控能力，能够获取某一个独立用户的具体情况。

第二，网络监控

网络是整个系统通路的保障。因为大型生产网络配置的复杂度通常比较高，以及系统网络架构的约束，所以网络监控一般比较难做。

一般情况下，从持续交付的角度来说，网络监控并不需要做到太细致和太深入，因为大多数网络问题最终也会表现为其他应用层面的故障问题。但是，如果你的诉求是要快速定位 root cause，那就需要花费比较大的精力去做好网络监控了。

网络监控，大致可以分为两大部分：

1. 公网监控。这部分监控，可以利用模拟请求的手段（比如，CDN 节点模拟、用户端模拟），获取对 CDN、DNS 等公网资源，以及网络延时等监控的数据。当然，你也可以通过采样的方式获取这部分数据。
2. 内网监控。这部分监控，主要是对机房内部核心交换机数据和路由数据的监控。如果你能打造全局的视图，形成直观的路由拓扑，可以大幅提升监控效率。

第三，业务监控

如果你的业务具有连续性，业务量达到一定数量后呈现比较稳定的变化趋势，那么你就可以利用业务指标来进行监控了。一般情况下，单位时间内的订单预测线，是最好的业务监控指标。

任何的系统故障或问题，影响最大的就是业务指标，而一般企业最重要的业务指标就是订单和支付。因此，**监控企业的核心业务指标，能够以最快的速度反应系统是否稳定。**反之，

如果系统故障或问题并不影响核心业务指标，那么也就不太会造成特别严重的后果，监控的优先级和力度也就没有那么重要。

当然，核心业务指标是需要经验去细心挑选的。不同业务的指标不同，而即使定义了指标，如何准确、高效地收集这些指标也是一个很重要的技术问题。比如，能不能做到实时，能不能做到预测。这些问题都需要获得技术的有力支持。

第四，应用监控

分布式系统下，应用监控除了要解决常规的单个应用本身的监控问题外，还需要解决分布式系统，特别是微服务架构下，服务与服务之间的调用关系、速度和结果等监控问题。因此，应用监控一般也被叫作调用链监控。

调用链监控一般需要收集应用层全量的数据进行分析，要分析的内容包括：调用量、响应时长、错误量等；面向的系统包括：应用、中间件、缓存、数据库、存储等；同时也支持对 JVM 等的监控。

调用链监控系统，一般采用在框架层面统一定义的方式，以做到数据采集对业务开发透明，但同时也需要允许开发人员自定义埋点监控某些代码片段。

另外，除了调用链监控，不要忘了最传统的应用日志监控。将应用日志有效地联合，并进行分析，也可以起到同样的应用监控作用，但其粒度和精准度比中间件采集方式要弱得多。

所以，我的建议是利用中间件作为调用链监控的基础，如果不具备中间件的能力，则可以采用日志监控的方式。

第五，系统监控

系统监控，指的是对基础设施的监控。我们通常会收集 CPU、内存、I/O、磁盘、网络连接等作为监控指标。

对于系统监控的指标，我们通常采用定期采样的方式进行采集，一般选取 1 分钟、3 分钟或 5 分钟的时间间隔，但一般不会超过 5 分钟，否则监控效果会因为间隔时间过长而大打折扣。

发布监控的常见问题

持续交付，或者发布系统，对监控的诉求又是什么呢？其实简单来说只有一句话，即：**快速发现发布带来的系统异常**。

对于这样的诉求，优先观察业务监控显然是最直接、有效的方式。但是只观察业务监控并不能完全满足这样的需求，因为有两种情况是业务监控无能为力的：

第一种情况是我们所谓的累积效应，即系统异常需要累积到一定量后才会表现为业务异常；

另外一种情况就是业务的阴跌，这种小幅度的变化也无法在业务监控上得到体现。

因此，我们还需要配合应用监控，关注被发布应用的异常表现。

但是，在分布式系统，或者微服务架构下，有时被发布应用本身并没有异常表现，却影响了与之相关联的其他应用。所以，除了关注被发布应用本身以外，我们还要关注它所在的调用链的整体情况。

在持续交付体系中，还有一些关于监控的其他问题，主要包括测试环境是否也需要监控、发布后要监控多久，以及如何确定异常是不是由你刚刚发布的应用引起的。接下来，我们一起看看如何解决这三个问题。

第一，测试环境也要监控吗？

首先，我们需要认识到一个问题，即：部署一套完整的监控系统的代价非常昂贵。而且，监控作为底层服务，你还要保证它的稳定性和扩展性。

因此，测试环境是否需要监控，确实是一个好问题。

我来说说我建议的做法：

如果你的监控系统只能做到系统监控或日志级别的系统监控，那么对于一些对系统性能压榨比较厉害、对稳定性也没太多要求的测试环境来说，无需配备完整的监控系统。

如果你的监控系统已经做到了调用链甚至全链路的监控，那么监控系统无疑就是你的“鹰眼”，除了发现异常，它还可以在定位异常等方面给你帮助（比如，对测试环境的 Bug

定位、性能测试等都有极大帮助)。在这样的情况下，你就一定要为测试环境配备监控系统。

你可能还会问，测试环境有很多套，是不是每套测试环境都要对应一套监控系统呢？这倒未必。你可以对监控系统做一些改造，通过数据结构等方式去兼容多套测试环境。

第二个问题，发布后需要监控多久？

一般来说，需要延时监控的情况都是针对生产发布来说的。

如果生产发布过程本身就是一个灰度发布过程的话，那么你基本就没有必要进行延时监控了。

但是，如果整个灰度过程本身耗时并不长的话，我的建议是要进行一定时间的延时监控。我们通常认为，发布完成 30 分钟以后的异常，都属于运行时异常。所以，**我建议的发布后监控时间为 30 分钟。**

第三个问题，如何确定异常是由我的发布引起的？

具备了持续部署能力之后，你最直观的感受就是发布频次变高了。

以携程为例，我们每天的生产发布频次超过 2000 次，如果算上测试环境的发布，则要达到 1 万次左右。如此高频率的发布，我怎么确定某个异常是由我这次的发布引起的呢？而且除了发布，还同时进行着各类运维变更操作，我怎么确定某个异常是发布造成的，而不是变更造成的呢？

解决这个问题，**你需要建立一套完整的运维事件记录体系，并将发布纳入其中，记录所有的运维事件。当有异常情况时，你可以根据时间线进行相关性分析。**

那么，如何构建一套完整的运维事件记录体系呢？很简单，你可以通过消息总线的形式去解决这个问题。

总结

今天，我围绕着灰度发布的最后一个过程：监控，展开了这次的分享。因为我们这个专栏要解决的主要问题是持续交付，所以我并没有过于详细地阐述如何设计一个监控系统，而只是

为你介绍了监控体系的一些基本概念，以及一些与持续交付、持续部署相关的问题。

首先，我介绍了监控的几种分类，以及分别可以采用什么方式去采集数据：

1. 用户侧监控，可以通过打点收集，或者定期采集日志的方式进行数据收集；
2. 网络监控，通过模拟手段或定期采样进行收集；
3. 业务监控，需要定义正确的指标以及相匹配的采集技术，务必注意实时性；
4. 应用监控，可以通过中间件打点采集，也可以通过日志联合分析进行数据采集；
5. 系统监控，通常采用定期采样的方式收集数据。

其次，我和你分享了三个对发布来说特别重要的监控问题：

1. 测试环境的监控需要视作用而定，如果不能帮助分析和定位问题，则不需要很全面的监控；
2. 一般发布后，我建议继续坚持监控 30 分钟，把这个流程纳入发布流程中；
3. 完整的运维事件记录体系，可以帮你定位某次故障是否是由发布引起的。

通过今天的分享，我们可以明白，只有拥有了强大的监控系统，我们才能放手持续交付，即监控可以为持续交付保驾护航。

思考题

你所在的公司是如何构建监控体系的呢，达到持续交付的需求了么？


欢迎你给我留言。

持续交付36讲

量身定制你的持续交付体系

王潇俊 携程系统研发部总监



新版升级：点击「 请朋友读」，10位好友免费读，邀请订阅更有**现金**奖励。

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 23 | 业务及系统架构对发布的影响

下一篇 25 | 代码静态检查实践

精选留言 (1)

写留言



丹丹兒

2018-10-09

老师好像大部分说的都是app的？

展开

1