21 | 事件管理（上）：事件降噪的几个典型手段

2023-02-24 秦晓辉 来自北京

《运维监控系统实战笔记》

课程介绍 >

讲述：秦晓辉

时长 16:32　大小 15.11M

你好，我是秦晓辉。

前面一章我们介绍了各个部分的监控实战，偏重如何采集数据、如何构建仪表盘。有了这些监控数据之后，下一步就是告警了，几乎所有的监控系统都具备生成告警事件的能力，但通常都不具有完备的事件后续处理能力。这里说的后续处理主要包括：多渠道分级通知、告警静默、抑制、收敛聚合、降噪、排班、认领升级、协同闭环处理等等。监控系统或多或少都有一些这方面的能力，但是通常都不完备，而这，正是 PagerDuty 这种产品存在的价值。

在事件处理方面，一般我们会遇到两个痛点，一个是告警事件太多，被过度打扰，另一个是重要告警疏漏，无法闭环处理。这个部分我会用两讲内容来介绍这两个痛点的解法。下面我们先来聊一聊告警事件太多的问题，看看通常是什么原因导致的。

告警太多的常见原因

最常见的原因，是**告警规则设置得不合理**。比如很多规则触发了告警之后，实际没有后续动作，只是起到常态化通知的效果，不需要排查，也不需要止损，甚至连个长线的 TODO 都没有。这类告警多了人就疲了，当重要的告警来临的时候，也容易忽略。这样的规则如果不经过治理，日积月累，就会产生很多无用的告警。

第二个常见的原因是**底层出问题导致所有的上层依赖都告警**，越是底层影响越大，比如基础网络如果出问题，发出几万条告警都是正常的。

第三个原因是**渠道错配**。一些不重要的告警也使用打扰性很高的渠道发出，用户可能会觉得单一渠道不可靠，想用多个渠道同时发送的方式来保障告警触达率，这也属于告警规则配置不合理的范畴。

第四个原因是**预期内的维护动作**导致的。比如程序升级变更，如果进程重启时间过长，可能会导致关联的服务告警，或者某个机器重启，忘记提前屏蔽了，也会产生一堆关联告警。

了解了常见原因，下面我们来看一下有哪些常见解法。

优化告警规则

类似 PagerDuty 这样的告警事件中心的产品，一定程度上是可以解决一些告警过多的问题，但如果能从告警规则的源头做好优化，自然是事半功倍。很多公司的告警规则配置没有原则可循，每次故障复盘先看告警是否漏报，一线工程师为了不背锅，自然是尽量多地提高告警覆盖面，但这么做的后果，就是告警过多，无效告警占多数，长此以往，工程师疲惫不堪。

那么告警规则的配置应该遵照一个什么原则呢？虽然每个公司业务不同，总有一些通用的原则可循吧？的确如此，这里我分享一下我个人的做法，希望对你有所启发。

每个规则都应该对应具体的 Runbook

Runbook 就是告警处理手册，也就是告警触发之后，应该细化排查哪些方面，按照一个什么方式执行动作，应该有一个手册参考。如果告警发生之后没有后续动作，那这个告警的意义就不大了。在 Nightingale 的告警规则配置页面，可以看到一个专门的 Runbook 配置，Grafana 的告警配置页面，也有一个 Runbook 的选项，就能看出他们对它的重视程度。

这个原则看起来是不是很合理？但是真要落地的时候，又会发现紧急需要处理的告警事件通常容易对应 Runbook，但是有些告警规则产生的告警确实没有那么紧急，有些只是想作为一个

通知，好像又确实难以对应一个固定的 Runbook。

针对这两种情况，我的做法是：不紧急的告警，也必须要有动作，虽然这个动作可能不是立马执行处理，但至少要创建个低优先级的工单之类的，或者提高告警阈值，等问题严重一些再告警。对于只是想通知一下的告警，其实都不算告警，只能看做是一种另类的报表和巡检手段，这样的“告警”就按照报表和巡检的逻辑来处理，比如把这类“告警”发到一个单独的邮件组或者单独的聊天群组，平时都不用关注，只要每天早上上班或晚上下班之前稍微看一眼就行，这样就可以减少打扰。

制定了这个原则之后，如果大家不遵守怎么办呢？还是有很多告警没有对应的 Runbook，作为管理人员，我们应该怎么处理？我的建议是分产品线统计一个指标：“Runbook 预置率”，就是各个产品线有多少告警规则配置了 Runbook，有多少没有配置，这个比例要统计出来，然后做成红黑榜，让大家去治理，治理一段时间之后有经验了，知道预置率大概在一个什么范围是合理的，然后就可以要求大家至少达到预置率下限的值。否则，就一定是有问题的。

Runbook 这个配置原则，是我最为推荐的原则，效果非常明显，其次就是告警分级原则。

每个告警都应该合理分级

基本每个监控系统都支持为告警规则配置不同的级别，基本上每个监控系统的用户也都知道应该做分级告警。但是具体怎么分级，却没有一个行业共识，大家各做各的。这里我也分享一下我的理解，你可以参考借鉴。

首先，不同级别的告警应该对应不同的处理逻辑，这样分级才有意义，比如通知渠道不同，通知范围不同，或者介入处理的人的范围不同，处理时效不同 ，如果某两个级别对应完全一样的处理逻辑，就可以合并成一个级别。

我的做法是把告警分成 3 个级别。

级别	通知渠道	说明
Critical	电话、短信、即时消息、邮件	影响收入的、影响客户的，必须立刻处理
Warning	短信、即时消息、邮件	无需立刻处理，但是如果不处理，时间久了就会演化为 Critical 的问题，可以先放入TODO列表，手头上的紧急事务搞定之后就去处理
Info	邮件	每天下班前稍微看一眼，偶尔一两天忘了看也无伤大雅

另外，如果 Critical 的告警规则很多，大概率也有问题，说明系统架构不够鲁棒，出点什么事都要立刻介入，系统没有自愈能力。这样的系统，需要配备更多运维人员，而且还很难跟老板讲清楚价值。怎么办？这就需要制定运维准入规则，哪个系统要交给运维人员来运维，首先要提供一些信息。

- 相关联系人，出了问题能够及时找到人，联系不上的话得能直接联系研发领导。
- 服务相关信息，比如代码仓库、系统架构、依赖哪些服务、依赖哪些系统参数、哪些 JVM 参数、常见问题还有处理办法等等。

然后进行准入评审，如果系统架构有明显问题，就没办法通过准入要求，不接受运维，如果老板要求必须接，那就只能加人了，或者明确说明在架构调整好之前，不负责 SLA。如果老板不接受沟通，那就跳槽吧，老板根本不懂运维、不懂稳定性还不信任你。

上面介绍的两个告警规则优化原则，是最重要的两个原则。照做的话，可以搞定大部分无效告警，下面我们再从告警产品的角度来看，有哪些手段可以解决告警太多的问题。

告警规则支持生效时间的配置

不同公司的业务相差很大，比如券商，交易时间段内需要高优保障稳定性，但是非交易时段，有些进程直接停掉也无所谓。但如果是监控系统，数据是时时刻刻都在上报的，没有高峰低谷，需要时刻保证高可用。你可以看下典型的生效时间配置，可以配置一周当中哪些天生效、哪些时间段生效。当然，整体是否启用的开关也必不可少。

图片中的配置其实可以继续优化，现在是只能配置一个时间段，如果能够配置多个时间段就更好了。

告警规则支持分级和不同的触达渠道

有的系统会把触达渠道和告警级别强绑定，用户只需要选择级别，就自动使用对应的通知媒介，有的系统会把二者拆开，我个人对这个没有倾向性，Open-Falcon 是绑定的，Nightingale 是分别配置的，分别配置的方式会相对灵活一点。

一些低优先级的告警就用打扰性低的通知媒介来通知，比如邮件，不要什么都打电话发短信，如果某个通知渠道经常用来通知一些低优先级的告警，时间久了这个通知媒介发出的告警就不被重视了。

重复告警支持最大次数和发送频率

有些告警短时间没法恢复，可能会重复发送。比如一分钟检查一下某个指标，如果超过阈值就告警，从某个时刻开始，触发了阈值，一分钟之后发出第一条告警，但是短时间没有恢复，持续了 10 分钟，监控系统在第二分钟检查的时候发现还是告警状态，可能还会发出一条告警，但是这个告警的重要性就远不如第一条，完全可以不用通知。通过设置发送频率，可以做到比如 1 小时之后再检查，如果 1 小时之后还是没有恢复，再发出第二条告警，这样告警数量就大幅减少了。当然，如果好几天没有恢复，也没有必要每个小时发一次，应该支持最大发送次数，限制一下未恢复之前的通知总次数。比如 Nightingale 对这两项的配置。

你看图片里还有一个启用恢复通知和留观时长，这里我也解释一下，这也是减少通知次数的典型手段。告警之后一般会发一条“Triggered”的消息，如果恢复了，有些用户也希望得到通知，这个时候会收到一条“Resolved”消息。但是重要的告警只要收到了，一般我们就会立马到平台上去排查，既然用户已经登录平台了，告警是否恢复其实已经可以在平台上看到了，不用再通知，所以有些用户就把“启用恢复通知”给关闭了，这种方式也可以减少消息通知次数。

留观时长有点儿像是去看病，告警了就相当于得病了，比如高烧 39 度，触发了告警，这个时候我们像医生一样进行一系列诊断和修复的动作，发现体温退到 37 度，这个时候应该出院吗？有的时候是不行的，需要观察一段时间，一段时间内没有再次高烧再出院，这就是留观时长的设计逻辑。

告警事件支持屏蔽配置

告警屏蔽我们比较熟悉了，一般就是在做一个预期的维护动作之前，提前把相关告警屏蔽掉，免得在维护期间又收到告警。

告警屏蔽一般有两种配置方式，一个是配置成未来的一个时间段，一个是配置成周期性时间段。未来的一个时间段，就是用来应对刚才介绍的预期内维护行为的，周期性时间段比较特殊，比如每天凌晨 1 点到 5 点不需要告警，或者周末不需要告警，就可以配置成周期性屏蔽规则。

告警事件支持抑制配置

告警抑制的典型使用场景是一个指标配置两条策略，不同的优先级和阈值，如果高优先级的告警触发了，低优先级的告警就被抑制，不再重复发送了。

比如磁盘使用率大于 88% 就发一个 Warning 级别的告警，大于 99% 则发一个 Critical 级别的告警，现在磁盘使用率在 85%，突然一下子写到了 99.1%，理论上这个时候两个告警都会触发，但是我们只希望收到 Critical 级别的告警。这就需要告警抑制规则的支持了。

有些监控用户会表达这种需求：某个核心数据库告警了，依赖这个核心数据库的告警都会发出来，希望这个场景也做个抑制，只收到数据库那条告警，上游服务告警就不要再发出来了。这个需求合理吗？

我觉得有待商榷。所有的上游服务都发出告警，虽然告警消息变多了，但是可以通过聚合发送的方式减少打扰，众多服务告警，也正好可以让我们知道影响范围。另外，服务之间的依赖关系错综复杂，依赖这个数据库的服务出了问题，确实可能是数据库导致的，但也可能是其他依赖导致的，如果这个时候通过抑制规则只发出数据库告警，可能就会忽略一些问题。

告警事件聚合发送逻辑

事件降噪发送最有效的技术手段，其实就是聚合发送，能够起到立竿见影的效果。用户的告警规则配置得太乱，系统是没办法左右的，但是短时间触发很多告警，系统是可以通过技术手段聚合之后再发送的。

事件聚合一般是根据两三个维度的信息进行聚合运算，比如告警接收者的维度、时间的维度、指定的某个标签的维度（比如产品线），也可能不指定聚合标签，只使用接收者维度和时间维

度，这样聚合率会更高。什么意思呢，我举个例子。

张三同学，在一分钟内，收到 10 条 Kafka 告警，10 条 Elasticsearch 告警。如果只按照接收者维度和时间维度聚合，就可以把这 20 条告警聚合成 1 条通知张三，通知消息可能是这么写的。

> 张三您好，最近一分钟您有 20 条告警，最高级别是 Warning，相关告警举例：
> 10.2.3.4 Kafka 有目录 Offline
> 10.2.3.5 Kafka 有目录 Offline
> 10.2.3.6 ElasticSearch 流量过大
> 更多详细告警信息，请点击这里查看

如果把产品维度也加上，这 20 条告警就会聚合成 2 条，1 条报的是 Kafka，1 条报的是 Elasticsearch。所有聚合发送的事件中心大都是类似的逻辑，当然也可以根据文本相似性之类的做聚合。

另外，我们也可以增加更多维度来聚合。那怎么做才是最佳实践呢？

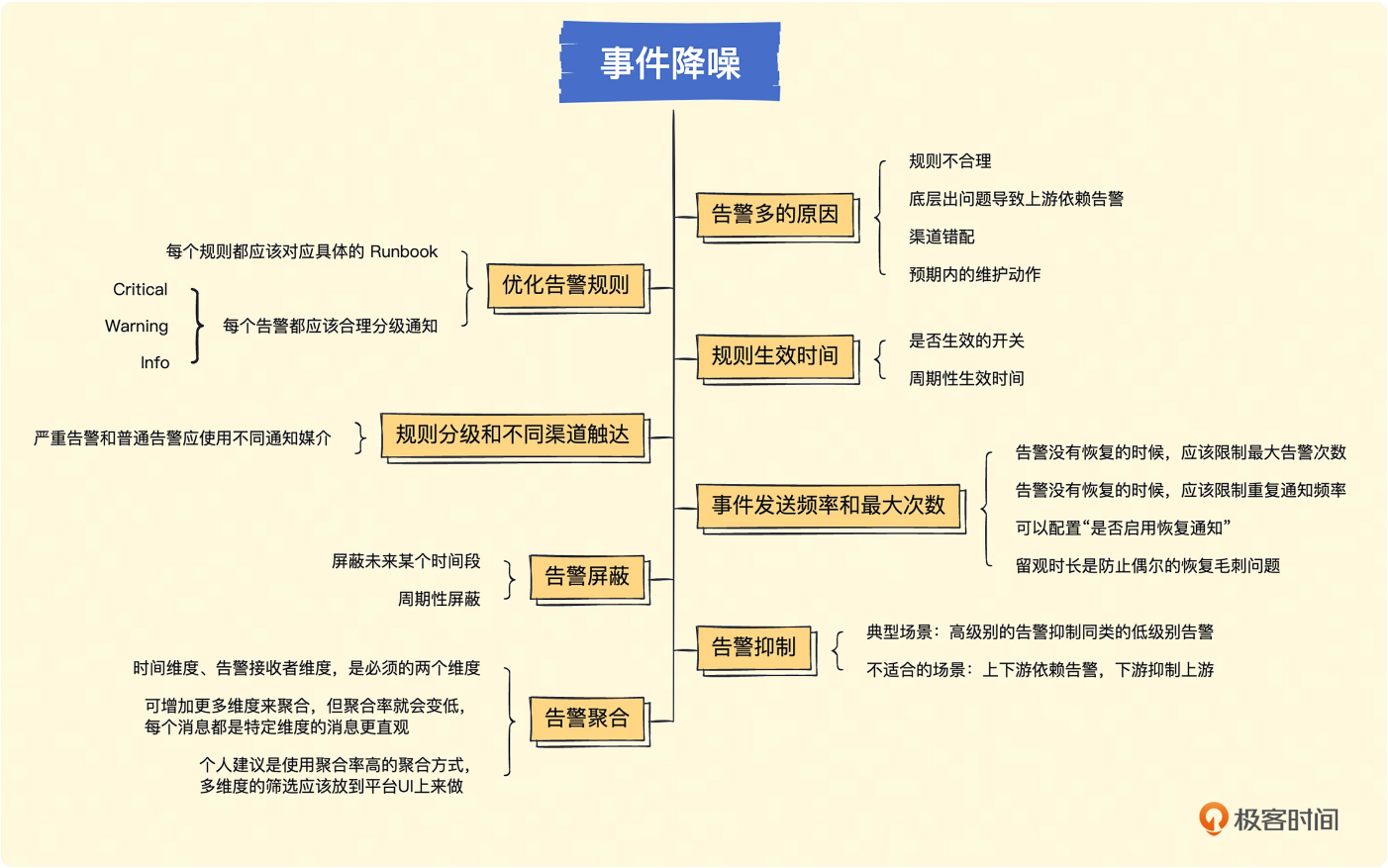
我分享一下个人看法：从告警聚合通知角度来看，只根据接收人和时间两个维度做聚合就够了，这样的聚合率最高，被打扰的次数最少。虽然会把多种告警消息混杂在一起通知给用户，显得有些混乱，但是用户一般也不会在短信、邮件、即时通信软件里期待分门别类的事件查看效果。想要更好的查看效果，可以去页面上看，页面上有更大的操作区，想怎么聚合就怎么聚合，想怎么看就怎么看。告警都已经发生了，大概率是要打开电脑处理的，都已经打开电脑了，在页面上查看一下也是顺其自然的。

针对告警事件太多的问题，主要就是上面这些解决办法。市面上的开源产品，只有 Prometheus 生态的 Alertmanager 做得相对完备一点，但是也不具备我上面说的所有的能力。主要是监控系统的重心，大都放到了数据采集、数据存储、告警触发引擎上面了，对于告警发送这块，关注相对比较少。而且，告警发送是个通用需求，Zabbix 需要、Prometheus 需要、Open-Falcon、Nightingale、各类云监控其实都需要，做一个统一的产品对接所有这些事件源，是个更合理的做法。下面我们对这一讲的内容做一个小结。

小结

告警事件太多的常见原因，包括告警规则设置不合理、底层服务告警导致上游大量告警、渠道错配、预期内维护等几类。对于怎么改进这些问题，我最推荐的方式是优化告警规则，搞定问题源头，这需要依赖一些原则，典型的原则有两个，一是要求所有的告警都有 Runbook，二是要求分级合理。对于告警规则，还要注意生效时间的配置、发送频率、最大发送次数的配置。对于告警事件，要做好预先屏蔽、抑制，对于最后产生的告警，要做到聚合发送，减少打扰。

另外所有的告警事件，建议都要持久化保存，至少保存几个月。通过这些历史资料，我们可以分析出很多信息，比如哪个团队接收告警最多、哪个告警规则发出的告警最多、哪些告警长时间都没有恢复、平均告警恢复时长是多少，我们可以根据这些数据做告警优化。



互动时刻

对于告警事件，如果一直没有恢复，我们有一些手段可以减少打扰，但是告警事件如果恢复了，就应该发送恢复通知（如果你设置了接收恢复通知的话），有的时候会遇到一些非常难受的场景，就是告警一会触发一会恢复，一会又触发一会又恢复，循环往复，烦得很，针对这个场景不知道你有没有什么解法，技术上的或者非技术上的都可以分享到评论区，同时也欢迎你把今天的内容分享给你身边的朋友，邀他一起学习。我们下一讲再见！

上一篇　20丨应用监控：如何使用日志来监控应用？

下一篇　22丨事件管理（下）：如何保证事件的闭环处理？

精选留言 (6)

写留言

SICUN

2023-02-25 来自北京

1.课后题：推送的报警信息可以带链接，链接跳转的页面添加人工介入的按钮，人工介入后对应报警就只记录不推送了，人工处理完之后把对应事件置为已处理，然后接着走监控告警规则，建议在加一个最大的单次人工介入处理时间，防止人工处理完忘记点已处理导致后续监控不推送问题。

延伸：问题解决后可以复盘问题反复出现的原因，然后对恢复脚本或是告警规则做改进。

2.想请问一下老师水平触发（Level Triggered）工作模式和边缘触发（Edge Triggered）工作模式的适用场景有哪些

作者回复: 这俩词我第一次听说😄

共 2 条评论 >　2

那时刻

2023-03-10 来自北京

思考题：如何处理报警重复的问题，我觉得可以通过滑动平均值的方式来聚合报警，比如，报警第一次触发的时候发出报警，然后在一定的时间之内如果有相同的报警发出/解除，进行滑动平均计算，如果达到再次报警或者是报警升级的阈值，再次报警；如果达到解除报警的阈值，则解除报警。以此来减少重复报警的次数

1

晴空万里

2023-04-11 来自广东

老师课程里面好像没有介绍：告警触发引擎的设计逻辑吧？还是我没看见，最近公司在做监控告警平台，不知道告警触发引擎怎么设计实现

作者回复: https://github.com/ccfos/nightingale/wiki/faq 007号问题


wayne

2023-02-28 来自浙江

之前我也一直在尝试引入AI来做告警收敛，效果不明显；最后还是通过时间线+告警层级两个维度来做收敛，效果更好些。不过大部分收敛规则是要运维同学花时间去配置的，他们也提出能否减少配置的功能，自动去聚合，老师提到的从告警接收人+告警时间维度来聚合，确实是个不错的解决方案，可以减少运维同学的配置工作。


Geek_1a3949

2023-02-24 来自上海

课后题：配置或增大告警规则的留观时长，观察一段时间后再恢复。


peter

2023-02-24 来自北京

请教老师几个问题：

Q1：告警处理这一块目前是否引入了AI和大数据？

告警事件需要几个月的保存，通常会积累大量的数据。请问，针对这些历史数据，公司是否会引入大数据和AI进行处理？或者目前是否有一些比较先进的公司采用了这些手段？比如阿里、京东等。

Q2：需要配置多少运维人员？

公司一般需要配置多少运维人员？可以结合具体的例子。比如极客这种规模需要多少？如果不了解极客的情况，不好回答，这时候可以根据作者自己公司的规模回答。

Q3：能否以作者自己公司作为例子讲解？即作者自己公司是怎么做监控的，比如，公司大致有什么业务，流量多大，用户量多大，机器数量等，基于这些信息，采用了什么框架进行监控、运维。老师可以以自己目前所在的公司为例子进行讲解，也可以以以前的单位为例子。如果方便的话，建议以加餐形式用一节课来讲；如果不方便，就在留言中回复即可。

作者回复: 1，我了解的公司没有在事件这块引入ai的，即使有，也是实验性质的，效果一般，因为数据量太小

2，case by case 来看，我们公司没有运维

3，很多数据是不能对外讲的，至于用什么方案，指标层面我觉得categraf+nightingale可以解决绝大部分公司的问题

共 2 条评论 >