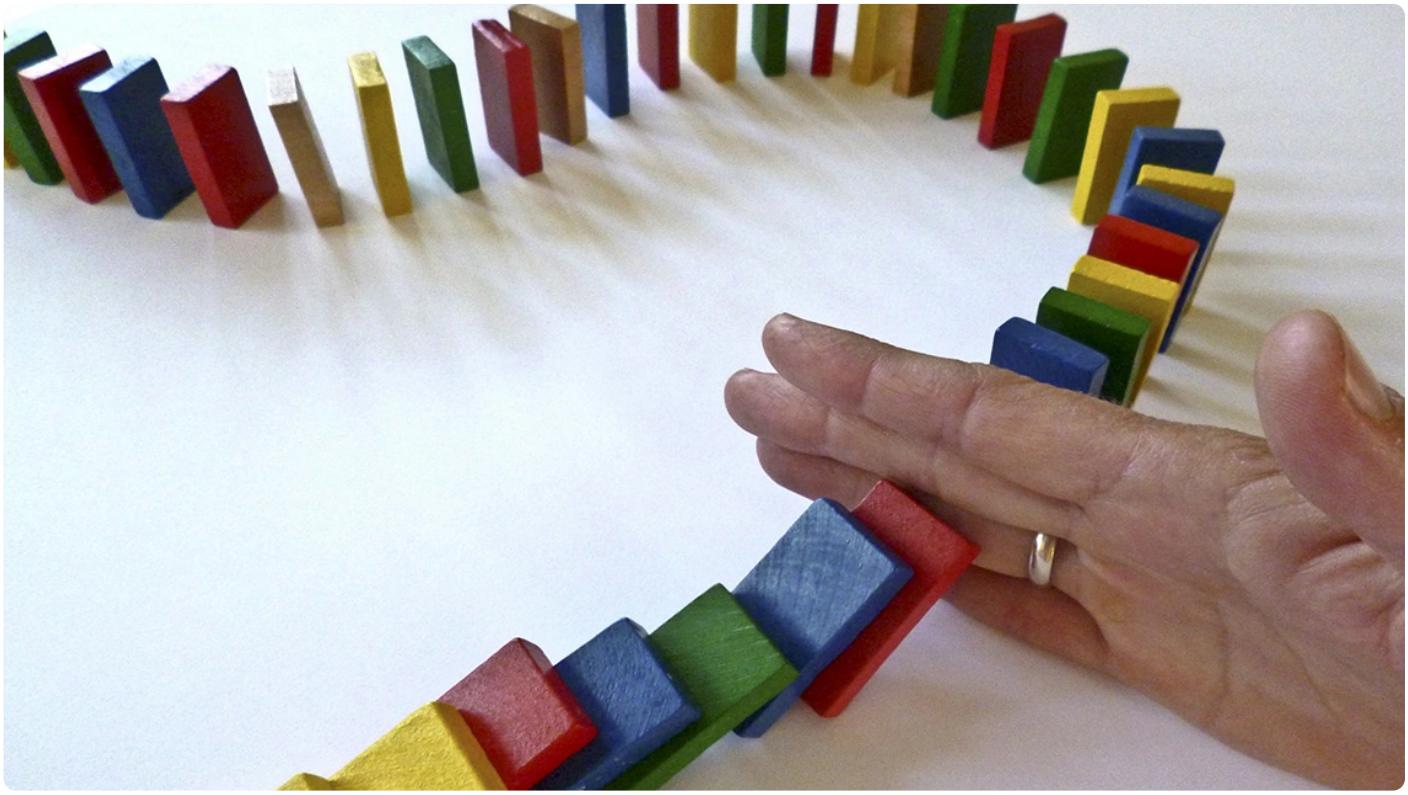
37 | 遇到线上故障，你和高手的差距在哪里？

2019-05-25 宝玉 来自北京

《软件工程之美》



你好，我是宝玉。在软件上线后，发生线上故障是一个常见的问题，但怎样对线上的故障进行处理，却很能反映出新手和高手程序员的差距。对于团队来说，如何应对线上故障，也同样能反映出线上运维水平的高低。

今天，我将带你一起分析一下，新手和高手在应对故障时有什么不同？大厂在处理线上故障时，有哪些可以学习借鉴的地方。

遇到线上故障，新手和高手的差距在哪里？

在这里，我把新手在处理线上故障遇到的一些常见问题列一下，同时，我们也一起分析下，高手是怎么处理这些问题的。

新手遇到复杂的线上故障，不知道该怎么下手

对于线上故障，有的很简单，从界面或者错误日志上可以直观地看到问题在哪，从而也好找到方法去解决。但有的故障，却没办法直观地看出原因，比如说内存一直在涨，CPU 居高不下，遇到这种复杂的故障，通常新手就不知道该怎么下手了。

而对高手来说，会在实践中总结一套自己解决问题的步骤，遇到问题，会按照解决问题的步骤有条不紊地去分析和解决。（比如说 caoz 的这篇《出了 bug 怎么办》，就充分体现了一个高手的水平。）通常通过下面这样的步骤：

第一步，评估影响范围；

第二步，试图重现问题；

第三步，临时方案和终极方案；

第四步，风险评估及持续优化。

如果你还记得专栏文章《28 | 软件工程师的核心竞争力是什么？（下）》里的内容，就会发现，其实这本质就是一种解决问题的能力。一步步分析、解决和预防问题。

从新手到高手，可以从借鉴像这样的方法开始，然后在实践中不断总结经验，形成一套自己的分析问题、解决问题的方法。

新手遇到线上故障，会想着马上修复 Bug

当发现 Bug 后，尤其是自己的 Bug，很多开发人员马上就想到了 Bug 的修复方案，迫不及待就要去写代码打补丁了。然而这样做的问题就是，匆忙之间打补丁，如果没有经过充分的测试，可能会引入新的 Bug，甚至是更严重的 Bug。如果要充分测试，那么意味着时间会比较长，而线上故障的时间越长，可能意味着损失也越大。

而对于高手来说，会首先对故障进行评级，看对用户的影响范围，如果是核心业务，大面积影响用户，那么当务之急是恢复生产，然后再考虑如何去修复 Bug。

恢复生产并不一定需要修复 Bug，可以用一些临时性的方案，比如说回滚系统到上一个稳定的版本；重启服务看是否能恢复正常。当然在恢复之前，还要尽可能保留当时的日志、故障场

景的截图、内存的 Dump（把当前内存数据保存的静态文件）等信息，用来后续查找故障原因使用。

遇到线上故障，新手需要时刻牢记：恢复生产、降低损失是第一要务，修复 Bug 是其次的。

新手遇到线上故障，不知道如何快速定位到 Bug 在哪

在临时恢复业务后，还是需要找到 Bug 在哪，后续才能从根本上解决。对于比较复杂的线上故障，新手通常不知道从哪里下手，看日志看代码都看不出所以然，而高手却总能快速地定位到 Bug 在哪。

高手快速定位 Bug 在哪，关键在于通过有效的手段，逐步缩小问题范围，直到找到 Bug 在哪里。

比如说，一种常见手段就是先重现 Bug，因为有了重现的步骤，就等于将问题的范围，缩小到重现 Bug 的这几步操作相关的代码上，就很容易发现问题在哪。

还有一种手段就是分析错误日志，通过错误日志，可以马上定位到错误在哪里。所以对于平时写程序，无论是客户端还是服务端，注意收集错误日志是非常重要的，可以帮助你在排查问题的时候节约不少时间。

还有一些不能重现的 Bug，则不是那么容易发现，其实也可以按照缩小问题范围的思路来定位。

比如说，Bug 是在最近一次部署后发现的，并且回滚部署后就恢复了正常，那么就说明问题很可能是由于这一次部署和上一次部署之间的代码变更导致的，如果代码变更不多，就可以通过分析变更的代码来定位。

像内存泄漏或者 CPU 高的问题，一般就可以通过分析内存 Dump 文件，分析当前是哪些线程占用资源多，线程运行的代码是什么；哪些变量占用资源多。从而可以缩小范围，快速发现问题在哪。

排除法也是一种缩小范围的方法，尤其是在架构比较复杂的情况，一次用户请求的操作可能经过多个服务，如果配合日志，那么可以对一个请求经过的每一个服务都进行日志分析，对于正常的服务可以逐一排除，直到找到出问题的环节，从而可以缩小问题范围。

所以下一次你在遇到难以定位的 Bug 的时候，也可以想想怎么样可以逐步缩小问题的范围，直到发现问题。

新手解决完线上故障后，下次可能还会发生类似故障

新手在遇到线上故障后，采用一些临时解决方案，比如说重启服务，发现恢复了，然后就把这事忘记了。对于线上的故障，如果不找到产生的原因，那么下一次还会发生类似的故障，甚至比以前还更严重。

高手对于线上故障，会仔细分析 Bug 产生的原因，从根本上解决，避免类似的故障再次发生。

比如说，我以前所在的项目组，采用敏捷开发，每周一个 Sprint，每周会部署上线。但上线后经常会出现一些故障，导致在部署后就要回滚，或者再打补丁。

虽然每一次上线后的故障可能都不一样，但是如果仔细分析背后的深层次原因，还是因为上线前没有充分测试导致的。每周 3～4 天时间开发，1～2 天测试，来不及对程序进行充分的测试。所以我们后来从流程上改进，将一个 Sprint 内开发好的程序，在测试环境测试一周后再上线，这样调整后，极少出现线上故障。

有关上面案例中流程改进结果，可以参考专栏文章《07 | 大厂都在用哪些敏捷方法？（下）》中“一周一个迭代怎么保证质量？”的说明。

对于新手来说，每一次解决线上故障，同时也是一次学习和总结的机会，不仅是学习如何解决一个线上故障，还要学习解决一类的线上故障，避免类似的故障再次发生。

大厂都是怎么处理线上故障的？

在处理故障方面，可以看到新手和高手的差距。同样，如果你留心观察，会发现各个大厂，也都有一套自己线上故障处理的流程。（比如：《SRE：Google 运维解密》《阿里如何应对电商故障？》《滴滴是如何高效率处理线上故障的？》）

左耳朵耗子 V

+关注

2012-11-13 23:29 来自 iPhone客户端

出现线上故障，recovery 大于 find root cause。另，对于严重问题，Amazon还要写分析报告，要问自己至少5个Why，如：为什么会发生？为什么测试到？为什么没在发生的第一时间检测到?为什么花这么长时间解决？等等。

通过看这些大厂的故障处理流程，你会发现，大厂其实是把高手解决故障的方式，变成故障处理的流程和操作手册，并且通过反复地故障演习。不断练习和强化对故障处理的流程，让系统更健壮，让新手也可以快速上手，做到高效处理线上故障。

至于具体的处理流程，其实大同小异。

首先，对故障进行评级。

根据故障影响的范围，对故障进行评级，从而决定后续的处理方案。比如说 P0 是最严重最紧急的，可能是大面积服务瘫痪，影响大量用户，需要紧急处理；如果是 P5，可能只是用户体验相关的，晚一点处理也没关系。

其次，要马上恢复生产，避免进一步损失。

使用临时方案，恢复生产减少损失是第一位的。可以采用部署回滚、服务降级等处理手段。

另外，要分析故障原因，修复故障。

最后，记录故障发生处理全过程，分析故障原因，提出后续改进方案。

大厂处理线上故障处理机制有哪些值得借鉴的地方？

从流程看，大厂处理线上故障的机制似乎并没有什么特别的，那么有没有值得学习借鉴的地方呢？答案是肯定有的。

故障报警和轮值机制

你可以先思考一个问题：如果你所在项目组的系统出现线上故障，要多长时间可以恢复正常？怎么样可以做到最快的速度恢复？

要做到最快速度处理线上故障，关键就是要让正确的人第一时间就可以去响应。正确的人就是对故障服务最熟悉的人，通常就是这个服务的开发人员。

但让所有开发人员 7x24 小时随时待命也不现实，所以一般大厂会采用轮值的机制，比如说对于每个服务，每周要安排两个人值班，一个是主要的，出现故障第一时间响应；另一个人准备着，以防万一联系不上主要值班人员时可以顶替值班。

大厂都有一个报警系统，值班的那一周，值班人员手机要 24 小时开机，笔记本要随身携带，如果负责的服务出现故障，那么会在第一时间被报警系统呼叫。如果 15 分钟没有人响应，就会层层往上传递，值班开发人员没响应就呼叫经理，再是总监，VP，直到 CEO。

这套机制虽然被很多开发人员诟病良多，毕竟值班期间要随时待命，但确实是一套非常简单有效的机制，让最熟悉服务的开发人员第一时间去处理，可以帮助线上系统以最快的速度恢复服务。

实战演习

在我工作经历中，不止一次出现过数据丢失的情况，其实丢失数据前，都有完善的备份恢复方案和日常备份，然而这些备份恢复方案却从来没执行过，等到真正出问题，才发现这个方案完全是不可行的，日常备份也早已被破坏无法恢复，最终导致数据丢失。

如果日常对这些方案有演习，去实际测试一下，就不至于这么狼狈。实战演习就是频繁地对故障进行演练，来测试平时做的这些方案是不是真的可行，这样遇到真正的故障，才不至于手忙脚乱不知道如何应对。

其中最有名的就是 Netflix 的混乱猴子军团，Netflix 在亚马逊云上建立了一个叫做 Chaos Monkey（混乱猴子）的系统，这些猴子会在工作日期间随机杀死一些服务，制造混乱，来测试生产环境下的稳定性。

也有人把这样的实战演习叫“混沌工程”。

> 混沌工程就像“疫苗”：注射少量潜在有害的异物以预防疾病，这种人为的“破坏”其实是有帮助的。混沌工程通过在技术系统中注入危害 (如延迟、CPU 故障或网络黑洞) 来建立这种免疫力，从而发现和修正潜在的弱点。《以毒攻毒： Google、Amazon、Netflix 如何用混沌工程控制系统风险》

日志记录和分析工具

对于软件来说，线上出现问题，分析日志记录是最简单有效的定位问题方式。这就要求平时在开发的时候，就要注意对关键日志信息的记录，同时还要搭建像 ELK 或 Splunk 这样的日志分析系统，方便查询日志。

举个例子：一个 API 请求，出现了随机无法访问的故障，而这个 API 可能会经过 5-10 个服务，怎么快速定位是哪一个服务出现问题?

一个好的实践是这样的：

对于每一个请求，都会分配一个唯一的请求编号（requestId），在经过每一个服务的时候，都带上这个请求编号，每个服务都把这个请求的输入和输出记录下来，输入的 url 参数是什么？http 的 header 是什么？输出的状态码是什么，输出内容的大小是什么？如果出错，异常信息包括错误堆栈是什么?

当出现故障的时候，找到一个有问题的 requestId，根据这个 requestId 去日志分析系统查询相关的所有服务的日志，这样马上就可以看出来哪一个服务返回的结果是有问题的。

当然还有一些其他好的实践，例如说新功能上线时，灰度发布的策略。通过开关控制，先让一小部分用户使用，如果出现故障，马上关闭开关，避免影响。

大厂的这些线上故障处理预防的实践都是公开的，通过网上的一些文章或者他们技术人员在技术大会上的分享，你也可以从中了解和学习到很多。重要的是看这些实践的好处是什么，哪些是可借鉴到你的项目中的。

总结

今天带你一起学习了线上故障的处理。对于线上故障的处理，基本原则就是要先尽快恢复生产减少损失，然后再去查找原因，最后不要忘记总结复盘。

要做到最快速度处理线上故障，关键就是要让正确的人第一时间就可以去响应。正确的人就是对故障服务最熟悉的人，通常就是这个服务的开发人员。

要让你的故障响应流程在真正遇到故障时能起到作用，需要经常做故障演习，测试你的故障响应流程，测试你的系统在故障下的稳健性。

线上故障的分析，少不了对日志的记录和分析，平时在开发阶段就应该要注意对日志的记录，同时也可以搭建一套适合你项目的日志分析系统，在遇到故障时，能及时的通过日志定位到问题所在。

最后，保持学习大厂对这些线上故障处理的好的实践，应用到你的项目中。

课后思考

你平时对于线上故障是如何处理的？有哪些可以改进的地方？你的团队对于线上故障是怎么样的处理流程？有哪些可以改进的地方？欢迎在留言区与我分享讨论。

感谢阅读，如果你觉得这篇文章对你有一些启发，也欢迎把它分享给你的朋友。

精选留言 (11)

邢爱明

2019-05-30

我在传统的制造业甲方公司，系统大部分是内部管理系统，对于线上故障处理，个人一直有几个疑问，希望老师能给一些指导意见:
1. 谁来主导线上故障处理的过程？我们现在定的是产品经理，这个岗位用户安抚、信息同步方面做的比较好，但由于完全不懂技术，对排查原因、制定解决方案基本帮不上忙。
2. 故障排查是不是应该有一个标准的分析过程，让运维、开发、安全各方能更好的协作？如判断影响范围、分析系统资源是否有瓶颈、查看系统日志报错信息、分析最近发布等等。目前一旦有线上事故，由于分工不同，大家在排查的时候容易出现扯皮的现象，运维说系统资源没问题，开发做最近没做发布，很影响故障的恢复进度。
3. 便利性和安全如何平衡？目前处于安全考虑，开发人员不能登录生产的应用服务器，一些核心系统生产数据库也不能查询数据，只能查看tomcat日志；而运维人员完全不懂系统的功能，tomcat的日志也不看。这样的安全限制在故障排查的时候造成了信息的割裂，难以快速对故障进行定位。

作者回复: 1. 谁主导线上故障，我觉得有两个指标要考虑：

一个是这个人或者角色要懂技术懂业务，这样出现故障，能对故障进行评级；
另一个是要能调动开发和运维去协调处理，这样出现故障能找到合适的人去处理，不然也只能干着急。

2. 故障排查上：

如果是操作系统、数据库、网络等非应用程序故障，应该是运维负责；

如果是应用服务故障，应该是开发去负责，即使开发最近没有去做发布也应该是开发去查。因为只有开发对于应用程序的结构才清楚，才能找出来问题。排查过程中，运维要给予配合。

3. 应该搭建起来像ELK这样的日志管理系统（可参考《38 | 日志管理：如何借助工具快速发现和定位产品问题？》），将应用程序日志也放上去，这样正常情况下就不需要去登录服务器了，直接就可以通过日志工具查看到异常信息。另外，一些特殊情况应该允许开发人员登录服务器排查定位。

共 2 条评论 >

8

纯洁的憎恶

2019-05-25

新手用野路子解决问题，高手用模型解决问题：
1.给问题评级。紧迫的调动优势资源先解决，一般的往后放放。

2.尽快恢复生产。生产是企业的首要职责，遇到问题优先恢复生产，减少直接损失，然后再正式的解决问题。
3.找到问题出现的位置。“搜集证据”，通过“粗调”在时空上缩小包围圈，再用“精调”明确问题点，运用排除法最终锁定问题。
4.分析原因，修复问题。
5.提出解决方案。钥匙不一定插在锁眼里，要沿着问题的线索不停“倒带”找到根源。再针对根源，站在系统和流程的高度制定解决方案，避免问题复现。

重点：
1.通过故障报警+业务骨干轮值机制，让正确的人第一时间响应问题。
2.通过实战演习，确保应急预案稳定可行。
3.通过使用日志记录和分析工具，积累、整理日常生产信息，出现问题才有得分析，否则重现问题也无济于事。

作者回复: 👍感谢分享补充!

6

鲍勃
2019-05-25

我是做物联网(嵌入式)Linux相关的开发的，感觉有一点就是：解决问题后，总结做的不够。需要不断学习和实践

作者回复: 总结还是蛮重要的。因为出了故障，多少反映出整个开发流程可能是存在问题的，比如开发时没有写自动化测试和代码审查，测试时没有充分测试各种路径。总结后，就可能会找出来这些潜在问题，比如说增加针对这类Bug的自动化测试代码，增加代码审查，测试时增加对这类Bug的测试用例，从根源上改进这些问题，从而做的更好。

4

calvins
2020-04-09

Bug故障，我觉得从三个方面考虑，第一，预防，包括应急方案，多维度评审，充分测试等，第二，排查，怎么快速定位，分析问题，这块高手和新手最大的差别是经验和知识面，有一套完整分析的流程和工具是非常重要的。第三，处理，怎么形成一套完整的处理流程，常见的是先预警，建立故障问题，itil系统流程跟踪，最终解决问题，后续就是反思总结，经验分享，但是从目前接触得大多系统来看，三个方面都有涉及，过程还不是特别理想，特别涉及多系统接入，扯皮还是不少的。

作者回复：👍赞总结！

帮补充一点：遇到线上故障，第一时间恢复生产非常重要！

3

hua168

2019-05-27

老师，像大点的公司一般都会有业务监控系统吧，直接用业务监控会不会有帮助？
比如比较著名的CAT（地址：https://github.com/dianping/cat）
像日志监控系统也运维写的吧，是不是开发给一个监控的API就行了？

作者回复: 有关这部分内容，可以参考下一篇的内容，会有更详细的介绍。

CAT是很好的业务监控系统，用好了一样可以很有帮助。

日志监控系统不需要自己从头写，开发或者运维搭都可以，搭建好了做一些配置或者二次工作就好了。

3

alva_xu

2019-05-27

这是ITIL要解决的问题。我觉得最主要还是从三个方面来看。一是从流程上，对于事件管理、问题管理、变更管理、服务等级管理等，要有明确的流程。二是要有合适的工具，比如ticket系统，CMDB，监控工具、日志平台等。三是从人员组织来看，要有一线、二线和三线团队的支持，根据所创建的ticket的严重性和紧急性，给予不同level的支持。当然这也是目前流行的devops要解决的问题。

作者回复：🙏谢谢补充：在对故障评级后，还应该要提交ticket用来跟踪整个故障解决的过程。

3

梁中华

2019-05-26

最好的方式是不出现问题，事情做在前面，多做设计评审和测试用例评审，大点的项目要做上线方案评审和应急回滚方案，当然灰度发布几乎是必须，staging环境验证也是必须的。

作者回复: 👍对，预防是最好的！只是再怎么预防还是有发生故障的可能，所以各方面准备措施都需要准备齐全。

2

六维
2019-06-27

目前线上故障，是直接通过工作群反馈的。 目前大小问题都要立即（包括春节期间）响应处理。一看到有消息，所有人员都绷紧了一条线。 故障评级和对应的处理策略是可以借鉴引入的，轮班制度也需要建立，不至于出现联系不到人的情况（要让所有的研发人员7*24小时待命是不现实的）。

作者回复: PagerDuty这个产品可以了解下看看，很适合用来做故障报警工具，不知道国内是否有同类产品。

1

ifelse
2022-07-06

今天带你一起学习了线上故障的处理。对于线上故障的处理，基本原则就是要先尽快恢复生产减少损失，然后再去查找原因，最后不要忘记总结复盘。--记下来

巫山老妖
2020-02-28

稍微总结了下：
新手处理线上故障

- 遇到复杂的线上故障，不知道怎么下手
- 遇到线上故障，会想着马上修复Bug，匆忙打补丁，可能会引入新的Bug，造成更严重的损失
- 不知道如何快速定位Bug
- 解决完线上故障，可能还会重犯

高手处理线上故障

- 会有一套解决问题的步骤

    - 第一步，评估影响范围
    - 第二步，试图重现问题
    - 第三步，临时方案和终极方案
    - 第四步，风险评估及持续优化
- 遇到故障，会先评级、评估影响范围，优先保证业务可用，恢复生产，再考虑修复Bug
- 通过有效手段重现Bug，逐步缩小问题范围，定位具体的错误位置
- 会仔细分析Bug产生的原因，从根本上解决，避免类似的故障再次发生

大厂处理线上故障值得借鉴的地方

> 大厂其实是把高手解决故障的方式，变成故障处理的流程和操作手册，并且通过反复地故障演习。不断练习和强化对故障处理的流程，让系统更健壮，让新手也可以快速上手，做到高效处理线上故障。

- 故障报警和轮值机制
    - 找对故障服务最熟悉的人
    - 轮值on call，报警响应
- 实战演习（混沌工程）
- 日志记录和分析工具（搭建ELK或Splunk这样的日志分析系统）
- 其他好的实践
    - 灰度发布策略
    - 开关控制灰度

这节课让我更深刻的了解处理线上故障的实践，前后端解决具体问题的方法可能会有所不同，但总体解决策略和思路是类似的。关于工程师解决问题的和分析问题的能力其实也是我们的核心竞争力，如何更好的解决问题，提升业务价值，是我们在整个成长过程中需要不停去思考并践行的。


小老鼠

2019-11-25

可以用精准测试工具