网站可靠性工程中的算法

NOI2022 冬令营
钟诚

目录

简介

什么是网站可靠性工程（SRE）？

- 在Google，网站可靠性工程起源于2003年。
- 它是一个框架、一套理念，目的是可靠地运行大型系统。
- 当你请软件工程师组建运维团队的时候，SRE就诞生了。
- SRE在运维层面负责生产系统的运行。

讲者介绍

- 钟诚
- 曾任NOI科学委员会学生委员
- 在Google SRE的岗位上工作了5年，在慕尼黑和苏黎世的办公室工作
- 《Google SRE工作手册》（The Site Reliability Workbook）中文版译者

SRE的基本原理

- SRE需要有执行力的服务质量目标（SLO）。
- SRE需要时间来做工程工作，致力于让明天更美好。
- SRE团队要有能力来控制他们的工作量。

拥抱风险

如何确定最佳的可靠性目标?

- 你是否认为，那些耳熟能详的网络服务都试图把100%作为自己的可靠性目标?

~~100%~~

对几乎**任何服务**来说，100%都是 **错误** 的可靠性目标

*— **Benjamin Treynor Sloss**, Google副总裁，Google SRE的创始人*

为什么100%这个目标是错误的?

- 没有一个用户可以感知到一个系统100%可靠性和99.999%可靠性之间的差别。系统的设计者如果要提升0.001%的可靠性，得千辛万苦地花费巨大的努力，但却得不到任何好处。
- 在用户和你的服务之间，有许多其它系统（他们的笔记本电脑、家里的WiFi、移动/联通/电信、国家电网）。这些系统合在一起，可靠性远低于99.999%。
- 极高的可靠性是有代价的：为了保证最高的稳定性，会牺牲开发新功能的速度，用户也会更晚地见到新产品，还会显著地提高开发的代价，并最终严重地降低一个团队能开发的功能数量。

但是……我们在NOI好像可以拿100%的分数?

- NOI的题都有特定的限制，是固定的、有限的、清楚地写在题面上的，而现实世界中的服务都将面临许多更复杂、更难以预料的实际情况。
- 解决NOI的题目是一次性的，做完了就大功告成了。现实世界中，我们会不断地定期发布新的功能，可能每周甚至每天都要发布。
- NOI的题目都是独立的，而现实世界中的服务都会和其它系统有各种交互，有数不清的依赖关系。

服务质量目标（SLO）

什么是服务质量目标（SLO）？

- “服务质量目标（SLO）”的确定，为系统的运行设定了一个目标。
- 它专注于从用户的视角检视系统的表现。
- 如果用户满意，就意味着SLO已经达到了。

如果不是100%，那是多少？

- 例子：公共澡堂
- 假设在过去的几年里，你发现学校的澡堂经常不太好用。
- 有时候，洗着洗着水温突然变冷，有时候彻底没水了，又有时候只有冷水了。
- 所以我们该怎么定义澡堂的可靠性目标呢？

定义澡堂的可靠性目标

- 假设热水管的设计温度为75°C。
- 那么，下面的目标是否合适：

  热水管里出来的水总是 ≥ 75°C

定义澡堂的可靠性目标

- 假设热水管的设计温度为75°C。
- 那么，下面的目标是否合适：

  热水管里出来的水总是 ≥ 75°C

- 如果98%的时间里，热水管出来的水是75°C，只有1%的时间是74°C、1%的时间是76°C呢？
- OK，这么看起来问题也不大啊。那下面的目标是否合适：

  热水管里出来的水总是 ≥ 70°C

定义澡堂的可靠性目标

- 假设热水管的设计温度为75°C。
- 那么，下面的目标是否合适：

  热水管里出来的水总是 ≥ 75°C

- 如果98%的时间里，热水管出来的水是75°C，只有1%的时间是74°C、1%的时间是76°C呢?
- OK，这么看起来问题也不大啊。那下面的目标是否合适：

  热水管里出来的水总是 ≥ 70°C

- 但如果热水总是70°C呢?

定义澡堂的可靠性目标

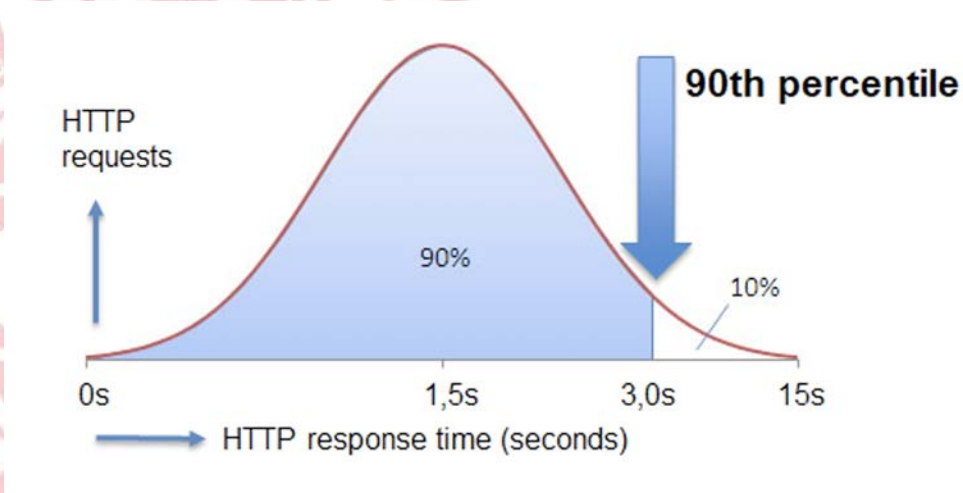
- 打开热水15秒后，热水管出水温度：
  - 95%的时间 ≥ 60°C
  - 90%的时间 ≥ 70°C
  - 50%的时间 ≥ 75°C
- 在连续的20秒中，前10秒的平均温度和后10秒的平均温度之差：
  - 95%的时间 ≤ 10°C
  - 50%的时间 ≤ 3°C
- 在95%的时间里，出水量 ≥ 6升/分钟

典型的互联网服务的SLO

- 每月正常运行时间占99.9%（每月可能有43分钟的时间不可用）。
- 每月99.99%的HTTP请求能正常返回200（成功）。
- 50%的HTTP请求在300毫秒（0.3秒）内返回结果。
- 99%的日志条目在5分钟内处理完毕。

关于统计性谬误

- 我们通常更喜欢使用“百分位数”，而不是一组值的平均值。
- 这样做可以考虑数据点的长尾效应，这些数据点通常具有与平均值显着不同（并且更有趣）的特征。
- 所以，我们不能假设“中位数”和“平均数”是一回事，它们可能差的很远。



错误预算策略

错误预算

- 错误预算，是指我们的可靠性目标与100%之间的差距。
- 这是一个用来消耗的预算。
- 如果我们的目标是99.9%的正常运行时间，那么当发生了一个20分钟的事故之后，这个月里还有23分钟的错误预算。

错误预算策略

- 错误预算策略，是指当你的服务用尽了错误预算之后，你答应一定会去做地的一些事情。
- 这些事情并不是罚款。
- 这些事情必须能显著地提升可靠性。

错误预算策略的例子

- 在服务再次达到可靠性目标之前（换句话说，错误预算再次变成为正数之前），不能再发布新功能。
- 下一步的工作计划只能选自事故的事后总结里提出待办项目。
- 开发团队每天都必须与SRE团队碰头，报告他们的改进工作。

小结：SRE的重要原则

- SRE需要有执行力的服务质量目标（SLO）。
- 任何团队，即使没有SRE，也可以设计错误预算策略。
- 错误预算策略是你的一把尚方宝剑，最终目的是防止用户在使用你的服务时感到痛苦。
- 它很容易上手，只要度量、计算、行动就可以了。

案例分析：全球Chubby服务计划内停机

- Chubby是Google的一个内部的分布式锁服务。随着时间的推移，我们发现当Chubby出问题的时候，一些其他服务总是随即发生事故。事实上，由于Chubby的故障率太低，以至于其他服务的负责人开始觉得Chubby服务永远不会出故障，从而将更多的服务依赖与它。因而，Chubby的高可靠性导致同事们产生了一种安全假象。
- 我们在这里采用了一个很有趣的解决办法：SRE保证Chubby能够达到预先制定的可靠性目标，但同时也会保证服务质量不会大幅超出该目标。每个季度，如果真实故障没有用完错误预算，就故意安排一次可控的故障，将服务停机。
- 这么一来，我们很快就找出那些对Chubby服务的不合理的依赖关系，强迫那些服务的负责人尽早意识到系统设计上的问题。

基于SLO的警报

设置警报时需要考虑的——

- **查准率（precision）**：在所有拉响的警报中，有多少真的发生事故了。如果放羊娃每次喊了“狼来了”的时候，狼都真的来了，那查准率就是100%。
- **查全率（recall）**：所有真实的事故里，有多少触发了警报。如果每次狼真的来的时候，放羊娃都喊了“狼来了”，那查全率就是100%。
- **检测用时**：从事故发生到触发警报花了多久。检测用时越长，消耗的错误预算也会越多。
- **重置用时**：事故解决（消失）以后，警报还会持续多久。重置用时越长，引起的混淆也会越大。

警报策略1：当错误率 ≥ 某个阈值

- 例如，如果一个月里的可靠性目标是99.9%，那么当连续10分钟的错误率 ≥ 0.1%就发出警报。
- 优点
  - 检测用时短：如果系统彻底停机（错误率100%），那么0.6秒后就会发出警报。
  - 查全率高：任何威胁SLO的事件都会触发警报。
- 缺点
  - 查准率低：许多不威胁SLO的事件也会触发警报。在10分钟里，如果系统的错误率恰好只有0.1%，其实只消耗了当月错误预算的0.02%。
  - 在极端情况下，你每天可能收到144个警报，即使全部忽略，你的可靠性目标仍然可能达标。

警报策略2：延长警报的时间窗口

- 在上述策略的基础上，我们可以通过延长时间窗口提高查准率。
- 例如，你决定在消耗了一个月的错误预算的5%时，才触发警报，也就是把警报的时间窗口从10分钟延长到了36小时。
- 优点
  - 检测用时仍然很短：如果系统彻底停机（错误率100%），那么2分10秒后就会发出警报。
  - 查准率变高了：由于错误率持续了更长的时间，因此触发警报的更可能是对错误预算构成重大威胁的事件。
- 缺点
  - 重置时间太长：如果系统彻底停机，那么无论如何警报都将持续36小时。
  - 在较长的时间窗口上计算错误率，会在内存或I/O操作上付出很大的代价。

警报策略3：延长警报出发前的触发时间

- 只有当错误率在持续的一段时间（如1个小时）里始终超出阈值，才触发警报。
- 优点
  - 查准率提高了：在警报触发前，错误率需要保持一段时间，意味着警报更可能对应着真实的事故。
- 缺点
  - 检测用时变长了：由于持续的“一段时间”这个参数不会随着事件的严重程度而变化，如果系统彻底停机（错误率100%），那么1个小时以后才会发出警报，这与错误率为0.2%的事故的检测用时是一样的。
  - 查全率变低了：如果错误率是波动的，一会儿超出阈值、一会儿没有超出，那可能永远都不会触发警报。

警报策略4：根据燃烧率发出警报

- 燃烧率是指你的服务消耗错误预算的速度。
- 燃烧率=1的意思是：按这个速度消耗错误预算，那么在一段规定的时间（如1个月）后，错误预算恰好变成0。
- 如果这1个月的可靠性目标是99.9%，而你的服务的错误率始终是0.1%，那么1个月过后，就恰好用完了所有的错误预算，燃烧率=1。
- 例如，我们规定，如果1个小时内就消耗了5%的本月错误预算（即燃烧率为36），就发出警报。

警报策略4：根据燃烧率发出警报

- 优点
  - 查准率较高：在消耗了相当大的错误预算之后，才发出警报。
  - 时间窗口更短，意味着计算的代价更小。
  - 检测用时更短。
  - 重置用时相对短了一些：58分钟。
- 缺点
  - 查全率较低：如果一个事故的燃烧率是35，那么一直都不会触发警报，但20.5个小时以后，就会耗尽一整个月的错误预算。
  - 重置用时：58分钟还是有点长。

警报策略5：基于多个燃烧率的警报

- 触发警报的策略可以考虑多个燃烧率，对应着多个时间窗口，例如：

错误预算消耗比例	时间窗口	燃烧率	警报级别
2%	1小时	14.4	警报级
5%	6小时	6	警报级
10%	3天	1	工单级

警报策略5：基于多个燃烧率的警报

- 优点
  - 能够根据问题的严重性来配置监控系统，使之适应各种情况。如果错误率很高，那就迅速触发警报，如果错误率低但持续时间很久，那么持续一段时间以后，最终还是会触发警报。
  - 查准率高：这一点与基于固定预算的警报策略一样。
  - 查全率高：因为有那个3天的时间窗口。
  - 我们能根据紧急程度来选择最合适的警报级别。

警报策略5：基于多个燃烧率的警报

- 缺点
  - 我们需要管理并推敲更多的数字、窗口大小和阈值。
  - 重置用时变长了：主要是因为有那个3天的时间窗口。
  - 当所有条件都满足的时候，例如5分钟内消耗了10%的预算，意味着6小时内消耗了5%的预算、1小时内消耗了2%的预算，那么将触发多个警报。你还需要设计一个抑制警报的措施，避免狼只来了一次，却喊了3声“狼来了”。

警报策略6：多个时间窗口、多个燃烧率

- 我们可以基于多个燃烧率，在第5个策略的基础上持续进行优化，只有当错误预算被活跃消耗的情况下才发出警报，从而减少误报的数量。
- 为此，我们需要添加另一个参数：一个较短的时间窗口，用于检查在触发警报时，错误预算是否仍在活跃消耗中。

警报级别	长窗口	短窗口	燃烧率	消耗的错误预算
警报级	1小时	5分钟	14.4	2%
警报级	6小时	30分钟	6	5%
工单级	3天	6小时	1	10%

警报策略6：多个时间窗口、多个燃烧率

- 优点
  - 更灵活的警报策略：允许你根据事件的严重性和团队的要求选择警报级别。
  - 查准率较高：这一点与基于固定预算的警报策略一样。
  - 查全率高：因为有那个3天的时间窗口。
- 缺点
  - 需要配置的参数太多了，可能使警报的规则难以管理。

事后总结：从失败中学习

为什么需要事后总结?

- 作为SRE，我们负责运维大型的、复杂的分布式系统。同时，我们还在不断地增加新功能、增加新系统。以我们的开发速度和部署规模，事故是难以避免的。
- 在事故发生后，我们要修复根源性问题，同时将服务恢复到正常状态。如果没有一种方法，让我们从已经发生的事故中学习经验，那么事故就可能循环反复地发生。甚至，随着系统规模和复杂度的增加，事故可能成倍增加，最终导致我们没有足够的资源处理事故，从而影响我们的用户。
- 因此，事后总结是SRE的一个必要工具。

事后总结的哲学

- 撰写事后总结的主要目的，是为了保证事故被记录下来，理清所有的根源性问题。同时，最关键的是，确保实施有效的措施，使未来类似事故重现的几率得以降低，甚至彻底避免。
- **“对事不对人”**的事后总结是SRE文化中最重要的。一篇事后总结必须重点关注如何找到引起这次事件的根本原因，而不是指责某个人、某个团队的错误或不恰当的举动。
- 如果因为一些“错误的”举动就公开指责或羞辱某个人、某个团队，那么人们就会自然地逃避事后总结。

两个例子

- 带有指责的事后总结
  - “我们一定要重写整个复杂后端系统！在过去的三个季度里，它每周都在出问题。每次我们都要一点一点地进行修复，真是烦透了！说真的，如果我再看到一个警报，那我就自己重写了。”
- 对事不对人
  - “如果重写整个后端系统，那么可能会避免这些持续产生的警报，目前版本的维护手册冗长，学习成本很高。相信通过重写，可以减少警报信息，未来的oncall工程师会感谢我们的。”

谢谢！