《SRE: Google运维解密》

图书基本信息

书名：《SRE: Google运维解密》

13位ISBN编号：9787121297264

出版时间：2016-10-1

作者：【美】Betsy Beyer（贝特西 拜尔）等

页数：480

译者：孙宇聪

《SRE: Google运维解密》

内容概要

大型软件系统生命周期的绝大部分都处于“ 使用 ”阶段，而非“ 设计 ”或“ 实现 ”阶段。那么为什么我们却总是认为软件工程应该首要关注设计和实现呢？在《SRE：Google运维解密》中，Google SRE的关键成员解释了他们是如何对软件进行生命周期的整体性关注的，以及为什么这样做能够帮助Google成功地构建、部署、监控和运维世界上现存最大的软件系统。通过阅读《SRE：Google运维解密》，读者可以学习到Google工程师在提高系统部署规模、改进可靠性和资源利用效率方面的指导思想与具体实践——这些都是可以立即直接应用的宝贵经验。
任何一个想要创建、扩展大规模集成系统的人都应该阅读《SRE：Google运维解密》。《SRE：Google运维解密》针对如何构建一个可长期维护的系统提供了非常宝贵的实践经验。

《SRE: Google运维解密》

作者简介

Betsy Beyer 是Google 纽约负责SRE 的一名技术文档作家。她之前曾为遍布全球的Google 数据中心与Mountain View 硬件运维团队编写文档。在搬到纽约之前，Betsy 是Stanford 大学技术性写作课程的讲师。她曾经学习国际关系与英文文学，并在Stanford和Tulane 获得学历。
Chris Jones 是Google App Engine 的一名SRE。Google App Engine 是一个PaaS 服务，每天处理超过280 亿个请求。他的办公室在旧金山，他之前的工作包括Google 广告统计、数据仓库，以及用户支持系统的维护。在之前，Chris 曾经在学校IT 行业任职，同时参与过竞选数据分析，以及一些BSD 内核的修改。他有计算机工程、经济学，以及技术政策学的学位。同时他也是一名有执照的职业工程师。
Jennifer Petoff 是Google SRE 团队的一名项目经理，工作地点在都柏林，爱尔兰。她曾经负责管理大型全球项目，包括：科学研究、工程、人力资源，以及广告等。Jennifer在加入Google 之前，曾在化工行业任职八年。她获得了Stanford 大学的化学博士与学士学位，同时她还拥有Rochester 大学的心理学学位。
Niall Murphy 是Google 爱尔兰团队广告SRE 的负责人。他拥有20 年互联网行业经验，目前是INEX（爱尔兰网络互联枢纽）的主席。他曾经写作以及参与写作很多科技文章与书籍，包括O ’ Reilly 出版的IPv6 Network Administration，以及很多RFC。他目前在参与书写爱尔兰互联网发展史。他拥有计算机科学、数学，以及诗歌学的学历（他当时一定是想错了！）。他目前与妻子和两个儿子居住在都柏林。
译者
孙宇聪，前Google SRE（2007-2015），山景城总部，曾参与构建运维Youtube 全球CDN网络，2008年奥运会直播项目，构建维护海量视频编码传输系统。后参与Google内部云平台运维工作，负责运维全球百万级别服务器集群，以及Borg、Omega等大规模集群理系统。2015年加入Coding，任CTO一职。回国后，积极推动国内容器化运维架构升级。目前是开放运维联盟之应用运维规范制定组，高可用运维规范制定者。

《SRE: Google运维解密》

书籍目录

《SRE: Google运维解密》

精彩短评

1、详细解释了google sre方面的理论和实践
2、很快速的翻了一遍，SRE观点应该是每个进行系统研发和运行的团队都应该关注的，也是每个工程师都应该关注的。主要讲了Google实践的一些观点、方法论和实践经验
3、SRE百科全书，每个人都能得到自己的收获，Google实现自动化之路是将所有东西都封装成API，包括屏蔽Shell脚本。
4、作为某大互联网公司的运维，国内没有公司能将运维做的这么规范，这么好，google的sre也是我们一直努力的方向。这本书也算是传统op向devops转型的指南。
5、对系统设计有一定启发
6、值得放在办公桌边，经常翻起审视和改进自己公司的业务。
7、其实不用那么神话，整本书有些章节非常的啰嗦还说不清楚，但是开头结尾不错。记住三点：第一，SRE都是通过google开发笔试的人，这保障了开发的效率和质量。第二，良好的体制，on-call机制和SLO机制。第三，非常重视事后总结，多个章节有提到。
8、买一本放在工位上，常常审视自己的项目是否到这种标准
9、粗读，没有读完，内容很多，自己功力不够
10、很不错的索引书
11、非常受用，连锁故障部分需再消化
12、作为一名OP，正在向OPs以及「FRE」转变。
13、绝不只是给运维读的书，所有的IT工程师应该都看看。
14、任何一个架构和程序设计没有从运维角度进行思考都是错误的
15、谷歌给我们的运维工作证名了，有规划了

《SRE: Google运维解密》

精彩书评

1、原文来自：http://blog.csdn.net/xindoo/article/details/52723114《SRE》这本书英文版已面世半年后，中文版终于面世。从4月、5月的时候，我就一直在尝试看英文版，由于自己英文水平有限，阅读进度和深度实在有限，看到中文版，对很多章节的内容才算是有了较深入的理解，一句话评价此书，这是一本运维转型的指导性书。 看过原版，再对照中文版，从内容上，并不比原版少什么，所以各位读者不必担心内容相对原版是否缺失，如果各位英语不好、但又想了解Google的SRE，放心大胆的买中文版吧，因为译者也是Google的前SRE，翻译的不能说原汁原味，但也八九不离十。 我自己本身也在国内某大厂做运维，我们也面临着传统运维向devops的转型，接下来我就结合自己实际工作的经历，谈谈我对这本说的理解。 这本书基本上可以分成几个大部分。* SRE的诞生 * Google内部软硬件环境 * SRE和Dev的协作* SRE自己是如何做事的 SRE是为了解决op和dev相互之间的矛盾和割裂的问题，用一些工程和规范来让op和dev之间有个平衡，并且最优化系统的发展。书中举出大量dev和sre系统的方法和规范，比如错误预算、部分运维工作交还dev、SRE协助dev团队健康发展等…… 从我自己的经验来看，其实作为一个op，一天到晚有一堆乱七八糟的事，曾经因为这些事，搞的我情绪都不太好。不同于国内一些公司，google考虑到了这些，制定了一系列的标准来平衡SRE工作上的问题，比如最多50%运维工作、完善的轮值机制、完善的SRE培训体系……，前两天还看过google的《重新定义公司》，从他们内部各种福利政策来看，google是一个非常人性化的公司。 运维工作中，有些是管理层需要做的事，但也有些内容能让你自己提升自己运维的效率。这么多年，SRE总结出了一套完善的方法论，比如和Dev团队的协作沟通，SRE在风险管理、on-call、故障排查、问题处理、故障后总结……，google都总结出了想当好的经验。 书中也介绍了google的一些软硬件环境，比如数据中心、网络、borg、Chubby（zookeeper）、监控系统、负载均衡、cron等。书中介绍了一些这类软硬件设计的思路，可以给那些想自己设计软硬件系统的公司一些方向。> We want our systems that are automatic, not just automated. >上面这句话是英文版原文，如何理解这句话?我们想让要系统是自治的，而不仅仅是自动的。这是一个设计系统先进的理念，想想我们往常是怎么设计系统的，是不是专注于解决一个问题，流程在这里卡了，需要人为干预，甚至是再做一个新的系统来解决某些问题。 举个例子，一个应用有数十台服务器，服务器会宕机，然后需要把服务器下线，再扩容一台。然后就会有一个监控系统发现问题，再弄个服务器下线的系统，自动化扩容的系统，然后都需要手工提单。这个时候，有人嫌提单麻烦，又写了个自动调其他三个系统解决问题的系统。就因为服务不是自治的，而我们一味强调自动化，导致系统越来越复杂，越来越难以自动化。其实我举的这个例子，google的borg已经很好的解决了，我认为borg基本上就是一个自治的系统。 总结一把，我觉得这本书并不是直接告诉你应该怎么做，因为不同的公司在不同的阶段关注的重点是不一样的，做的事也不可能和google相同，盲从某些方法论可能会得到x'fan相反的结果，所以我的建议是把这本书当成一种方向性的指导。

《SRE: Google运维解密》

章节试读

1、《SRE: Google运维解密》的笔记-第15章，事后总结：从失败中学习

事后总结的主要目的是为了保证该事故被记录下来，理清所有的根源性问题，确保实施的有效措施在未来重现的几率降低甚至避免重现。基本的时候总结条件为：1. 用户可见的宕机时间或者服务质量降级程度达到一定标准2. 任何类型的数据丢失3. on-call工程师需要人工介入的事故4.问题解决耗时过长5.监控是由人工发现的，而非警报系统事后总结应该对事不对人，一篇事后总结的重点必须是关注如果造成这次事件的根本问题，而不是指责某个人或某团队的错误或者不恰当的举动。如果时候总结的重点是指责某个人的错误举动，那人们就会逃避时候总结，模糊事情的原因。引入时候总结机制的时候，最大的阻力来源于对投入产出比的质疑。帮助面对挑战的策略：1. 逐渐引入事后总结流程，首先进行几次完整的和成功的事后总结，证明它们的价值，同时帮助确定具体书写事后总结的条件。2. 确保对有效的书面总结提供奖励和庆祝。不光通过前面提到的公开渠道，也应该在团队或个人的绩效考核中体现。3. 鼓励公司高级管理层认可其中

2、《SRE: Google运维解密》的笔记-第184页

任何足够复杂的软件工程项目都会遇到一定程度的不确定性，要么是组件的设计方法不确定，要么是问题的解决方式不确定。

3、《SRE: Google运维解密》的笔记-序言

可靠性就像安全性，越早关注越好。

4、《SRE: Google运维解密》的笔记-第1页

只有靠着对细节的不懈关注，做好充足的灾难预案和准备工作，时刻警惕着，不放过一切机会去避免灾难发生。— 这是SRE最重要的理念

5、《SRE: Google运维解密》的笔记-第118页

值得注意的是，在某些系统中某一类问题可能全被排除了。例如，在细心设计的集群文件系统实现中，由于某个磁盘出现问题导致延迟问题是非常不可能的。

6、《SRE: Google运维解密》的笔记-第23页

* 目标
* 没有 100% 可靠的服务, 达到一定程度的可靠性之后, 应把精力转向他处.
* ** ” 当设立了一个可用性目标为99.99%时, 我们即使要超过这个目标, 也不会超过太多, 否则会浪费为系统增加新功能, 清理技术债务或者降低运营成本的机会. ” **
* 可靠性目标成为错误预算: 提供明确和客观的指标决定服务在一个季度中接受多少不可靠性(用于 SRE 部门和产品部门的沟通). 只要错误预算耗尽, 新版本的发布就会暂停(?但是错误率由 SRE 部门提供, 而发布由产品决定?) -> 认为风险由产品开发决定, 一个变通是, 当错误预算即将用尽时, 降低发布的频率. 即使是网络中断或者数据中心故障影响了错误率, 发布频率也会降低, 因为 ” 每个人 ” 都有义务保障服务的正常运行.
* 可用性指标: 请求成功率. **用我们记录的请求成功率与用户期望的服务水平做对比.**

* 成本
* 可用性: 99.9% 到 99.99%; 收入: 1000000刀 -> 改进后的价值: 1000000 * 0.09% = 900 刀

* 需求
  * 面向消费者需要低延迟(队列空为好), 离线计算需要吞吐量(队列满为好). 需要分别响应不同的需求. -> 两个集群, 低延迟/高吞吐量

7、《SRE: Google运维解密》的笔记-第71页

● 某个最主要任务是加速现存的集群上线的团队是没有动力去减少服务运维团队在生产流程后期运维服务产生的技术负债的。
● 一个不亲自运行自动化的团队是没有动力去建设一个能够很容易自动化的系统的。
● 一个产品经理的时间表如果不受低质量的自动化影响，他将永远优先新功能的开发，而不是简化和自动化。

8、《SRE: Google运维解密》的笔记-第3页

研发部门最关注的是如何能够更快速地构建和发布新功能。运维部门更关注的是如何能在他们值班期间避免发生故障。研发部门想要：“ 随时随地发布新功能，没有任何阻拦 ”，而运维部门则想要：“ 一旦一个东西在生产环境中正常工作了，就不要再进行任何改动。 ”
运维团队宣称，任何变更上线前必须经过运维团队制定的流程，这有助于避免事故的发生。
研发团队吃过苦头之后则宣称不再进行大规模的程序更新，而是……

9、《SRE: Google运维解密》的笔记-第一章

监控系统不应该依赖人来分析报警信息，而应该由系统自动分析，仅当需要用户执行操作的时候，才需要通知用户

10、《SRE: Google运维解密》的笔记-第36页

我们在这里采用了一个很有趣的解决办法：SRE保证全球Chubby服务能够达到预定义的SLO，但是同时也会确保服务质量不会大幅超出该SLO。每个季度，如果真实故障没有将可用性指标降低到SLO之下，SRE会有意安排一次可控的故障，将服务停机。利用这种方法，我们可以很快找出那些对Chubby全球服务的不合理依赖，强迫服务的负责人尽早面对这类分布式系统的天生缺陷。

11、《SRE: Google运维解密》的笔记-第8页

初期几个万能的工程师的确可以解决生产问题，但是长久看来一个手持“ 运维宝典 ”经过多次演习的on-call工程师才是正确之路。

12、《SRE: Google运维解密》的笔记-第86页

鼓励删代码，统计源代码库里删除代码的行数

13、《SRE: Google运维解密》的笔记-第208页

无法预知的性能因素：坏邻居、任务重启

14、《SRE: Google运维解密》的笔记-第39页

我们应该从思考（或者调研）用户最关心的方面入手，而非从现在能度量什么入手。

15、《SRE: Google运维解密》的笔记-第6页

可用性改进，延迟优化，性能优化，效率优化，变更管理，监控，紧急事务处理以及容量规划与管理。

16、《SRE: Google运维解密》的笔记-第85页

我们的工作最终是在系统的灵活性和稳定性上维持平衡。

17、《SRE: Google运维解密》的笔记-第44页

Google员工每周要写一个短小的无固定格式的工作总结，称为“ Snippets ”。

18、《SRE: Google运维解密》的笔记-第16页

主要有以下几种方式使用监控系统：
对真实问题进行报警。
对比服务更新前后的状态变化：新的版本是否让软件服务器运行得更快了？
检查资源使用量随时间的变化情况，这个信息对合理制定资源计划很有用。

19、《SRE: Google运维解密》的笔记-第71页

自动化代码和单元测试代码一样，当维护团队不再关心它与它所覆盖的代码仓库同步的时候就会逐渐死去。

20、《SRE: Google运维解密》的笔记-第58页

两个例子的决策都很值得思考

21、《SRE: Google运维解密》的笔记-第12页

Google并不会使用专门的物理服务器运行专门的软件服务器。例如，并不存在一个具体的物理服务器运行我们的邮件系统，而是采用一套集群管理系统进行资源分配，它的名称为
Borg。

22、《SRE: Google运维解密》的笔记-第118页

相关性（correlation）不等于因果关系（causation）。

23、《SRE: Google运维解密》的笔记-第1页

不能将碰运气当成战略。

24、《SRE: Google运维解密》的笔记-第79页

所有代码都默认提交到主分支上（mainline），然而大部分的项目都不会直接从主分支上进行直接发布。我们会基于主分支的某一个版本创建新分支，新分支的内容永远不会再合并入主分支。Bug修复先提交到主分支，再cherry picking到发布分支上。这种方式可以避免在第一次构建之后，再引入主

分支上的其他的无关改动。利用这种分支与cheery picking的方法，可以明确每个发布版本中包含的全部改动。

25、《SRE: Google运维解密》的笔记-第34页

* 质量度量
* 请求延迟 (Req time)
* 错误率 (Web errors)
* 吞吐量 (Web QPS) -> **分应用展示?**
* Google 云计算的可用性指标: 99.95% -> 60*24*365*0.0005 = 262.8 min/year -> 我们的可用性?

* 特色
* 4/5个指标, 多/少都不好.
* 监控, y 轴指数分布
* 数据收集每10秒一次, 每一分钟汇总一次. 目标像这样: **99% 的 get RPC 调用在 < 100ms 的时间内完成.**, 每天可以出一个这样的报表.

* 总结
* 指标越少越好, 少到不能更少
* 性能指标保持简单
* 从松散的目标开始, 逐渐收紧. 不要一开始就追求完美
* 对内指标要求可以比对外高一些, 留有余地

26、《SRE: Google运维解密》的笔记-第11页

* 集群资源分配: Borg(分布式集群操作系统), 下一代 Kubernetes(2014)
* Large-scale cluster management at Google with Borg
* Borg, Omega, and Kubernetes
* 负责运行用户提交的任务. 每个任务由多个实例组成, Borg 会为每一个实例安排一台物理服务器, 执行具体的程序启动它
* 负责任务的监控, 如果异常, 终止并重启
* 命名: BNS: /bns/<cluster>/<user>/<task>/<instance>
* 任务需要在配置中声明其所需的具体资源(cpu/mem), 超过则立即 kill

* 存储
* 分布式存储, 小文件和大文件进不同的集群.
* 单个集群一年内会损失上千块硬盘, 数据中心有专门的团队来处理

* 网络
* 这些概念比较陌生, OpenFlow 的软件定义网络, 带宽控制器优化带宽.
* 从地理位置, 用户服务和远程调用三层进行负载均衡

* 监控报警
* 定时抓取指标, 超出触发报警
* **新旧版本的对比: 新版本是否让软件服务器更快了?**
* 检查资源用量随时间的变化, 制定资源计划.

* 服务

* 所有服务使用 RPC 通信, 开源实现为 gRPC
* 格式为 Protocol Buffer(与 Apache Thrift 相比) (大小比 xml 小 3-10 倍, 序列化/反序列化快 100 倍) (和 json 比?)
* 服务和存储根据流量分散到各大洲的机房

* 开发
* Code review

27、《SRE: Google运维解密》的笔记-第1页

* DevOps 在 Google 的实践

传统开发/运维分离的解决方案在规模扩大后沟通成本上升(“ 随时发布 ” vs. “ 不再改动 ”) -> 新型运维团队 SRE(50%-60%标准开发, 其他为85%-99%能力的开发, 为了开发系统代替手工操作) -> 最多 50% 时间用于运维工作, 余下开发系统来自动化

* SRE 方法论

* 运维工作最多占用 50% 时间
* 遇到故障事后写总结
* 因为信息系统的特点, 不是也不该追求 100% 可靠, 给出现实的可靠性. 在实现这个可靠性的前提下, SRE 可以做各种创新
* 监控, 通过预案/手册缩短平均恢复时间
* 70% 的事故源于部署变更 -> 渐进发布, 精确检测, 回滚机制

28、《SRE: Google运维解密》的笔记-第42页

理解系统行为与预期的符合程度可以帮助决策是否需要投入力量优化系统，使其速度更快、更可用，或者更可靠。如果服务一切正常，可能力量应该花在其他的优先级上，例如消除技术债务、增加新功能，或者引入其他产品等。

29、《SRE: Google运维解密》的笔记-第246页

理解如何构建分布式共识实际上就是理解某个服务应用如何实现强一致性和高可用性。

30、《SRE: Google运维解密》的笔记-第7页

事后总结应该包括以下内容：事故发生、发现、解决的全过程、事故的根本原因，预防或者优化的解决方案。

31、《SRE: Google运维解密》的笔记-第14页

一个缓慢的不断重启的实例要好过一个永远不重启一直泄露资源的实例。

32、《SRE: Google运维解密》的笔记-第120页

在日志中支持多级记录是很重要的，尤其是可以在线动态调整日志级别。

33、《SRE: Google运维解密》的笔记-第76页

从代码修改提交到部署到生产环境一共需要多长时间（也就是发布速度）

《SRE: Google运维解密》

版权说明