《SRE：Google运维解密》译者导读

前言

问世近一年以来，《SRE: Google 运维解密》一书销量累计已两万余册。我想首先感谢各位读者对本书的支持，真的是衣食父母呀！如果还没有下单购买，是不是看过本文之后可以考虑尽快呢？

随着SRE概念在在国内外的火爆传播，相信很多朋友也对此书有了一定程度的了解。感谢 GitChat 平台，我这次有机会和大家再分享一下书内书外的故事。希望不管您是否看过本书，本文的信息都能对您有用！

成书花絮

首先要明确的是，这本书是一本文集。Google内部在2014年进行了一次题目+作品征集工作，由封面上的编者团将收集来的投稿筛选，据称砍掉2/3内容才（原始资料据称超过1500页）最后得以问世。技术类书籍成书难，和大数据项目的关键难点很像。那就是——没有数据。据我和国内运维岗位交流的感觉，别说写文档了，就连服务器密码都是只记在脑子里的，根本没时间写下来好嘛。玩笑归玩笑，但是道理不变，Google SRE 给我留下的一个不可磨灭的印记就是——文档重过一切。没写文档，救火责任永远在你自己身上背着，出问题半夜也要爬起来解决啊。写完文档，救火责任就变成大家共享的了，就可以开心的睡觉了。每个团队成员会主动维护本组文档，时刻保持其为最新。知识共享的第一步，当然就是要先把知识记录下来啊。

2016年初我就听说这本书很快会发行，4月初英文版上架之后，联系出版社等一系列事情做完之后，已经是近5月底了。6月和7月真的是日以继夜，每天翻译差不多 6- 8小时 ，8月做完审校等工作，9月正式完成首发和大家见面 。我做过统计，每次我能坚持全神贯注翻译两个小时，最多只能翻译10页原文。可是该书英文版就550页啊，哭死。在无数个挑灯夜战的夜晚过后，我自问还算是交上了一份差强人意的答卷，希望没有辜负大家的期待和信任。

本书封面：巨蜥

据编者透露，他们选择了很多动物的方案，可惜都曾被 O’Reilly 其他的书用过了，最终决定这个方案的原因是，巨蜥的英文名（(Monitor Lizard)，Monitor 与 SRE常用词 ”监控“ 一模一样，于是就这么愉快的决定了。（nerd的日常，笑）

首发现场请到了编者之一 Chris Jones, 时任 App Engine SRE，很多成书的内幕消息就是他告诉我的。据称美国首发仪式未做宣传，一共只发出不到10本。而在中国，在 GOPS 大

GitChat

会主席萧田国萧帮主的大力支持和宣传下，首发一个小时之内准备的两百本签名书全部售罄。

下面咱们言归正传，详细逐一说一说书中的内容。

第一章： SRE 概述

基本上在每次交流的过程中，大家总会谈到 SRE 理论体系的建立过程，以及与国内常见的运维岗位定义之间的区别与相同之处。这里我提出几个看法：

马后炮

我曾经在数次演讲中用过“马后炮”这个词来描述SRE的理论体系。中文”马后炮“一词的确略有 贬义，但是感觉用在这里却恰如其份的描述了SRE建立过程中摸着石头过河的过程。世界上不存在万能药，SRE 体系也绝非一天之内设计建成的完美罗马城。这里有我的个人理解：SRE 代表的是一种新的工作方式。通过主动收集以前散落给各个部门的一系列具有相关性的职责(Accountability)，管理层同时配发了一定的决策权(Authority)，再利用一定程度的自治权(Autonomy) 达到可持续发展的目的，这三种因素相互结合，相互作用最终造就了 Google 内部举足轻重的千余人 SRE 团队。

GitChat

与国内运维的本质区别在于职责与权力的统一

书中本章所列八条方法论，可以和上述AAA分别对应。我在交流中经常开玩笑的说，SRE 也是背锅侠，不一样的是 SRE 是自豪的背锅侠。不仅有锅背锅，还经常找到暂时没有造成业务影响的锅来主动背，自豪的背。如果要问这是为什么？那我可以反问，这难道不就是职责所在吗？既然 SRE 负责服务的稳定性，那就应该承担服务不稳定的后果。对职责的抗拒通常是由于决策权不统一，运维团队无法消除问题隐患，也无法改进代码，再加上常常归责于人的制度而造成的。就事论事，说起来容易做起来难呀。但是我觉得一定要意识到，个人英雄主义永远是不太稳定的依靠。钢铁侠也有大起大落，对吧。相比中国文化中常常灌输的挽狂澜于既倒这样的大场面，SRE 更喜欢平时排除隐患，到点就回家睡觉。这些问题在书中后面的章节不是会有讨论，希望大家看得时候可以多留意一些。

对SRE风潮的个人理解

很多企业的产品流程仍然停留在所谓计划经济+链条型体制下运转。 业务交需求给研发，研发交代码给运维上线，许多许多问题都是在这种“扔过墙”的过程中产生的。

目前逐渐风行的 DevOps 理念强调的是将IT能力向链条前部推进，如果将此应用在 Dev 与 Ops 之间，那么就是通过 CI/CD 等手段，达到促进 Dev 和 Ops 之间的合作关系的目的。而 SRE， 或者说 Google， 则更进一步，将链条式体系升级换代为三根支柱体系（我冠名的）。 业务、研发、 SRE 三根支柱互相支撑，互相协作，达到了一个更高效的协作高

度。这本书中蕴含的方方面面，都基本可以归结为 SRE 与业务团队和研发团队之间交互所涉及到的方方面面。 所以，SRE比我们中国语境中的运维岗位要广泛很多，可以称为一个职业了。

第二章： Google 的生产环境

在本章中 Google 描述了其世界闻名的数据中心设计和内部应用软件架构。用一句简单的话讲， Google内部实现的是资源交付的平台化。一个普通员工，甚至非程序员所能调动的资源规模都是几乎令人难以置信的。举个例子，每个 Google 程序员入职第一周都会运行一个运行在数百台机器上的分布式排序程序，不需要申请，也不需要排队，随时需要随时运行。Google 最近计算出的 SHA1 冲突值，耗费了 6500 个CPU 年，以及110 个 GPU 年。其实算算也不过一万多台8核服务器跑一个月哦，还好，还好。

我在这里将这一章书中讲的很多小点归结为三类基本资源分别进行描述。

计算资源

计算资源的分配和使用，是现在炒得最热门的话题了。不管是之前的OpenStack, vmware, xen, Ganeti (http://www.ganeti.org/ 很多人不知道这个优秀的Google Xen 管理套件开源项目）, 还是最近火热的容器云, Docker, Kubernetes, Mesos 本质上都是在做同一件事 分配和使用物理资源。在本章中，Google描述了Borg系统的架构。

GitChat

由于 Google 物理资源规模十分巨大，全球百余个集群，百万+量级的服务器数量，很多传统的管理方式已经不适用了。 首先，虽然单个物理资源损坏率不高，但是在这么庞大的基数下，基本每分钟都有服务器宕机，重启，或者出现硬件故障。如果以传统 Ops 的方式来处理这些故障，由某Ops远程处理或者亲自飞到对应地点，当场调试解决所有问题再回家，那招多少人也不够用呀。所以在Google这个体量规模下，全球部署的现状下，这样做法明显是不行的。

有位名人曾指出（真说过，不过半开玩笑）：All computer problems can be solved with one more layer of indirection. 为了解决极大规模集群事故造成的维护困难问题，SRE推行了物理资源与逻辑分离的做法。利用 Borg 系统，创造了一种新的资源单位——容器。 我当时所处的Borg SRE 团队起的正是这样一个承上启下的作用。总有人问我 Google 内部是不是也用容器云，此处我只能说，Google 用的时候，这玩意还木有名字呢。

向下，Borg SRE 通过书中描述的“系统管理软件”（自动化系统）自动化的处理所有物理资源的异常状态，根据需要利用自动化工单系统与数据中心驻场服务人员沟通解决问题。驻场人员负责执行物理操作，SRE 则负责制定计划。分工明确又可以共同提升。

向上，Borg 则提供了一套虚拟资源的抽象层，包括相应的交互模型，让用户可以直接以 Task, Job 等原语构建自己的应用，无需再关心物理资源的分配与使用问题。

Borg SRE团队负责运行维护自动化系统，保障虚拟资源的交付能力 (Scheduabity) 和交付质量 (Scheduling Dealy)。Borg SRE 同时负责维护基础的生产系统环境变更，例如每两周

更新一次全部生产服务器的Kernel版本。在这个体量下，任何需要人工干预的过程都是不可想象的。

存储资源

我经常说分布式系统中最大的问题就是存储。如果将进程的状态存在于某个分布式存储系统中，进程本身就可以做到无状态，可伸缩，可迁移。

Google的存储系统明显是分层形式构建的，大致分为 机器本地SSD / （ HD基本逐渐已被淘汰），集群本地的 Chubby, Bigtable, Colossus 等，以及跨集群自动同步的 Megastore, Spanner 等等。 这样，研发可以根据需要选择最符合当前使用需求的场景的存储服务。

这里值得一提的是，Google大部分系统都是基于“集群本地” (Cluster local) 的概念构建的。整个集群内部通常可以被视为“局域网”，其中所有机器共享一个大的网络灾难域，以及数个小的供电灾难域。Google集群虽多，但是每个集群的结构都高度一致。可以真正做到无差别迁移。SRE经常唯一需要记住的就是集群的命名规则（物理位置）。在集群间迁移变得如此简单，甚至有的团队每个季度都全球布局一次。如果底层系统不将这类信息公布给使用者，而是自下向上替用户决策，将造成极大的浪费。 所以有人问我，Google云平台是否超卖，答案是否。如果虚拟资源超卖，直接导致用户性能没有保障，也就使得用户必须买更多的资源，这中间的浪费，谁来买单呢？曾经有人说过，如果公司内部激励机制出了问题，数目惊人的浪费还真不如直接烧钱，每天几百万，还能烤烤火呢！

GitChat

网络资源

Google 网络设计中分数据中心网络，B4 (Backend backbone), B2 (Backbone) 三类。其中集群网络大家可以阅读 Jupiter Rising: A Decade of Clos Topologies and Centralized Control in Google’s Datacenter Network 这篇 paper. 数据中心内部全三层网络，双机之间5微秒延迟，每台机器40G上下行带宽，真的很难想象。

2008年的金额危机期间，美国、欧洲很多中小型ISP受损倒闭。Google作为当时的现金大户收购了超级多的Dark fiber，随后这些就Google骨干网络的核心。我在 Youtube 期间，又参与构建了Google的CDN网络，在这里我就不多写了，有兴趣可以单独交流。

理论篇

如同前文所讲， 我认为SRE的成功基础在于职责、决策权和自治权的统一。这一篇的内容就充分体现了这三方面的内容。

首先，SRE团队运作过程的关键就在于制定和维护一套可以高度量化的目标，即服务质量指标（ SLO ）。有了这样一个目标，通过不断地制定和完善执行方案，对执行过程中遇到的问题不停地追根溯源，形成了SRE独特的而高效的工作方式。 书中第3章，风险的量化定义，是SRE的立身之本。稳定性和可靠性应该作为产品的一个重要属性进行定义。提供定义依据，制定衡量标准，保证达标则是SRE职责所在。

SRE 应该直接向最终用户负责，SRE 工作质量的衡量标准就是服务的质量高低。我在国内某次交流中，曾经与人聊过Google大量采用自下向上的 OKR 管理方式。但是这往往只是一个侧面印象，其实 Google 内部当然也有 自上至下的管理方式，SLO 就可以算作 Google SRE 的 KPI。 这里的详情可阅读书中第4章，服务质量指标，就不再赘述了。

确立了目标之后，有关如何提高或者保持该目标。可以阅读书中第6章，分布式系统的监控来详细了解。本章里面提到的四大黄金指标是基础中的基础。但是更让人脑洞大开的一点还是集中于“长尾”问题的讨论上。”长尾”难题几乎是大型分布式系统在解决完生存问题之后遇到的标配问题。由于长尾问题复现困难，对数据分析能力，系统调试与测量能力要求很高，难以根治。

长尾问题还有放大效应，底层服务的长尾问题会给调用频繁的上层应用带来数倍以上的性能影响。曾经我有一个 Google 同事，任职期间所做的唯一工作就是跟踪调查某存储系统磁盘IO的长尾难题，从应用一路查到内核。最后在内核中发现并解决了一个锁竞争问题。由于他的这几行改动，底层存储服务的长尾延迟降低了，上层大量依赖操作存储的应用服务便可以省去很多超额部署的容量，这几行代码为 Google 节省了每年几千万美金的资源成本。他笑称，自己应该可以免费拿十年工资了。（当然了，估计没发，所以他走了。）

本大章中的其余章节，分别描述了在SRE工作过程中所涉及到的方方面面。其中对琐事(Toil) 的处理很有意思。Google SRE 每个季度都曾经发送内部调查问卷收集关于琐事所占比例的信息。但是估计某位 Manager 催手下填问卷催的急了一些，这位大侠一怒之下向全体 SRE 发送了”关于每季度琐事调查问卷繁琐程度的调查问卷“，笑喷了。

GitChat

第9章，简单化，这一章中的内容也很有意思。在 Google 内部我曾经参与组织过很多次 ”删代码竞赛“ 活动，一删几万行。后来在 Coding 也曾经举办，效果很好。 程序员一定要记得，代码即债务。如果无法安全的删掉垃圾代码，不仅仅是编译速度的问题，这恰恰反映了深层次的研发架构问题。Google 工程师都以少写代码为荣，我曾经有连续几个月都是代码行数净输出为负，如果按五行一元起发工资，那岂不是要哭死了。

有关简化认知复杂度的问题，是一个值得深思的话题。一个人的认知能力总是有限的，在 Google 这样的体量下，必须要进行统一化，简单化，否则谁也不可能记住端到端的所有信息。正如 Borg 通过抽象资源降低了大部分 Google 程序员对物理资源使用的认知复杂度一样，大部分 Google 自动化系统都在由 指令型系统向意愿型系统转变，这一点有兴趣的话，可以线下交流。

实践篇

这一部分内容较多，我将书中章节按照自己的理解重新排序了, 方便大家阅读和理解。

前文说过 Google SRE 都是自豪的背锅侠，那么如何专业的而负责的背锅呢？这里就应该看以下章节了：第11章 Oncall ， 第12章 故障排查方式， 第13章 紧急事件响应 第14章 事故流程管理 ，第15章 事后总结， 以及第16章 故障跟踪工具。 我曾经举过好多例子，SRE 的 Oncall 是一个人负责整个团队的紧急事务处理，绝非各管各的。 这样，非Oncall 的人可以真正投入到开发项目中去。我常常开玩笑的说，什么时候运维团队成员可以做到能够随时按需请假了，下班关手机，那就说明团队走上正轨了。一个单独的 SRE 团队

是一个高度协作的团队，每个人有自己的事业专注点，也有共同维护的一亩三分地。平时的监控平台维护，文档修改，报警规则等等，都会及时与大家同步。

另外一个值得关注的重点是负载均衡问题，主要参看以下几章： 第19章 描述了数据中心之间的负载均衡系统的设计实现，第20章 描述了数据中心内部的负载均衡系统设计实现。第21章讨论过载问题，第22章讨论了雪崩效应和连锁故障问题。我曾经和很多人讨论过，Google 的数据中心设计SLO为99%。 ”冗余“ 基本上是各种服务上线第一天就所必备的要求。冗余又分”单活“ ，”多活“等等，在存储层面的”多活“功能支持下，Google 大部分系统都可以做到随意迁移，跨区域多活，这中间主要就靠负载均衡体系的完善。一个用户请求从发起到抵达 Google 生产服务器，中间要经过至少三层负载均衡系统，处理过程中更是触发N个RPC 在许多机器上并发执行。这中间又离不开数据中心内部负载均衡系统的帮助。有人问 N + M 冗余如何做到，我总是笑着先问他 N 是多少。 不压力测试，根本没办法谈什么冗余。很多时候，为了更好的，更精确的制定N，我们还要拆分服务，重组架构，这些都是 SRE 的关注范围。

过载与雪崩效应这两章写的非常之好，这和下文会提到的分布式共识系统一同成本书我最喜欢的前三章。面对着铺天盖地的用户流量，负载均衡系统，服务器组件，甚至重试逻辑等等设计稍有不慎就会触发比全天峰值还要高几倍的流量攻击自己，实在是不能大意啊。一旦出现雪崩效应，很难恢复正常，常常需要全服务停机一段时间才行。App Engine 就曾经出现过这样的大问题。某种因素导致的数据中心下线，导致全球流量就像洪水一般一个一个将残留的数据中心淹没了。

GitChat

第23章 分布式共识系统，第24章 分布式周期性任务系统 第25章 数据处理流水线 更是绝佳之作。在我参与设计实现Youtube直播系统之时，曾经花费大量精力参与设计了一套基于 BASE 存储系统的一致性解决方案。在结束两周讨论之后，回程飞机的路上，我翻看内部文档看到了对Paxos系统的讨论，当时真如醍醐灌顶，目瞪口呆。前文讨论过，分布式系统最难的设计难点就在于状态存取一致性问题，而有了Paxos之后，一切都不再是问题。回想起来，看样还是应该好好上学，80年代就发表了的 Paxos 系统设计居然直到2010年才学到，读书少总被骗啊！

我强烈建议大家真的仔细读这一章，目前开源实现中ZooKeeper, Etcd, Consul等基于的也都是Paxos协议的变种，相信读过这篇会对大家未来的分布式应用设计有直接的启发作用。这之后的分布式周期系统，数据处理流水线系统，基本的水平扩展方式都依赖 Paxos ，理论指导加具体实践，感觉全书就看这三章，足矣。

其他章节中例如第26章 数据完整性，是比较难啃的一章。Google 存储的海量数据管理所遇到的困难数量也绝对是海量的。Gmail 当年出事之后，靠磁带备份系统成功恢复了用户数据，有兴趣的人可以搜索相关的 Youtube 视频有详细介绍。视频中曾说过一句话，备份系统不重要，重要的是恢复系统。如果你只是搭了一套备份系统，没有平时经常演习数据恢复，那么关键时刻它一定会失效的。唯一能确保安全的方法，就是建立自动化的，随机的数据恢复机制。

本部分其他章节在这里就不再赘述了，测试这一章也比较难啃，如有兴趣大家可以直接找我私下交流。

SRE管理

前文提到，SRE 的自治主要体现在 50% 的研发工作红线上。 本部分中最有启发性的讨论，莫过于 第29章，对中断性任务的处理。 我们现在都生活在一个信息爆发的年代，每天各种方式的通知消息不停地打断我们的思路，甚至很多人根本已经无法长时间集中注意力。自从学习了有关 流状态（ Flow state) 的相关知识之后，我常常自省，今天又在回微信，查邮件，以及刷朋友圈中度过了，项目木有进展啊，长此以往情何以堪。

这的确是值得我们深思的问题，运维团队由于负担日常运维工作，中断型任务不可避免，如果长时间让自己处于这个状态中会有许多不良现象。所以，为了自己的身心健康，也应该多多关注本章。

本部分其他章节详细描述了SRE与研发团队，产品团队的合作关系。中美组织结构，企业文化，团队关系还是略有区别，计划经济与市场经济的对比，从这几章中可以窥见一斑。

与其他行业的对比

这部分虽然安排在最后，但是我认为是非常值得一读的。伴随而来的问题也让人脑洞大开，比如SRE/运维团队与民航大飞机驾驶员的共同点与区别在哪？与消防队员，救生员相比呢？这些行业都远比软件行业成熟，他们职业的发展路线是不是对SRE的职业发展方向有参考价值呢？相信看完本章，你会找到自己的答案的。

结束语

GitChat

感谢耐心读者看到最后，希望本文能给您带来一些有用的信息，《SRE：Google 运维解密》目前各大网店均有销售，期待与大家的交流活动！