Broadview®
www.broadview.com.cn

PEARSON

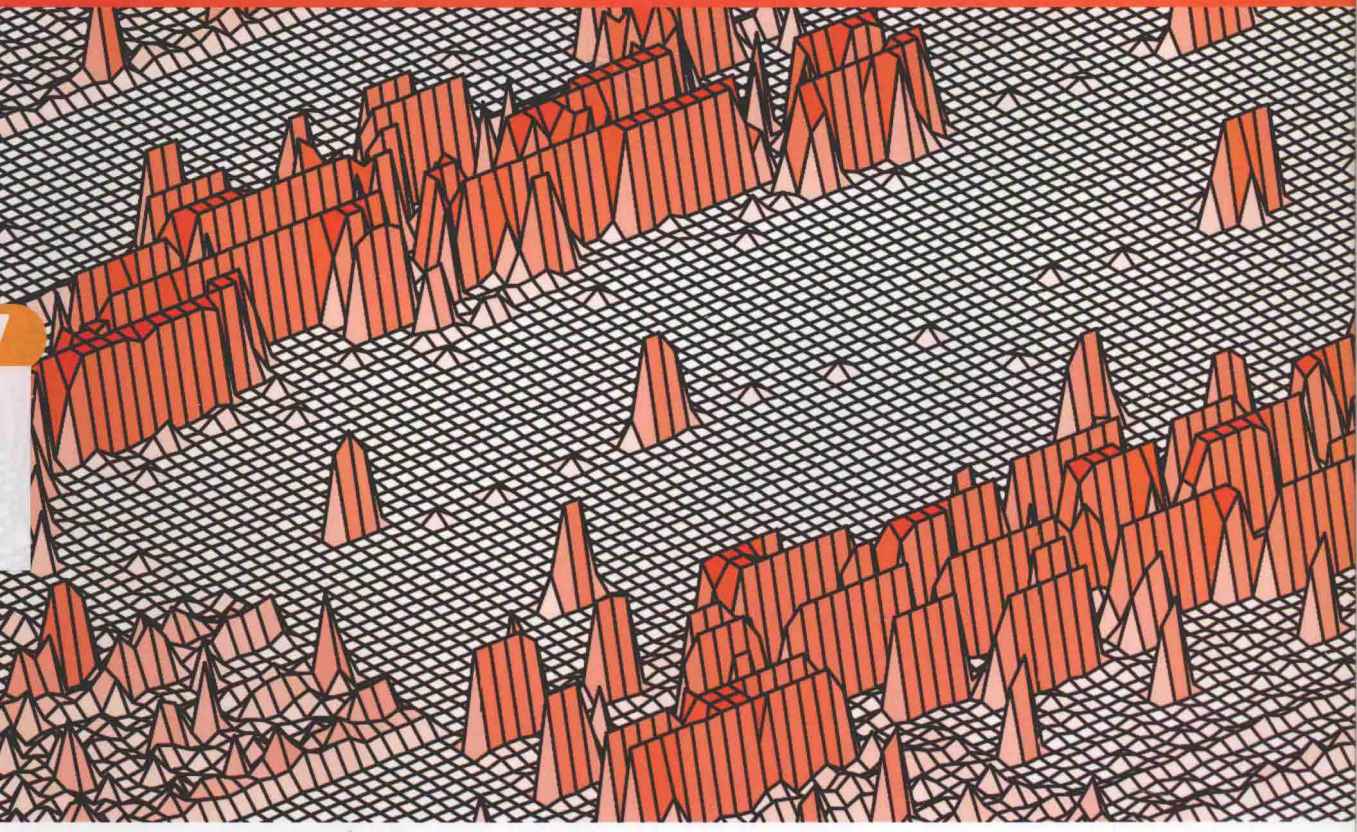
性能之巅

洞悉系统、企业与云计算

Systems Performance:Enterprise and the Cloud

【美】Brendan Gregg 著

徐章宁 吴寒思 陈 磊 译



中国工信出版集团

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

性能之巅

洞悉系统、企业与云计算

Systems Performance: Enterprise and the Cloud

【美】Brendan Gregg 著

徐章宁 吴寒思 陈 磊 译

電子工業出版社
Publishing House of Electronics Industry
北京•BEIJING

内 容 简 介

本书基于 Linux 和 Solaris 系统阐述了适用于所有系统的性能理论和方法，Brendan Gregg 将业界普遍承认的性能方法、工具和指标收集于本书之中。阅读本书，你能洞悉系统运作的方式，学习到分析和提高系统与应用程序性能的方法，这些性能方法同样适用于大型企业与云计算这类最为复杂的环境的性能分析与调优。

版权贸易合同登记号　图字：01-2014-4017

图书在版编目（CIP）数据

性能之巅：洞悉系统、企业与云计算 /（美）格雷格（Gregg,B.）著；徐章宁，吴寒思，陈磊译. —北京：电子工业出版社，2015.8
书名原文：Systems Performance: Enterprise and the Cloud
ISBN 978-7-121-26792-5

Ⅰ. ①性… Ⅱ. ①格… ②徐… ③吴… ④陈… Ⅲ. ①计算机网络 Ⅳ. ①TP393

中国版本图书馆 CIP 数据核字（2015）第 172687 号

策划编辑：张春雨
责任编辑：贾　莉
封面设计：吴海燕
印　　刷：三河市鑫金马印装有限公司
装　　订：三河市鑫金马印装有限公司
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　　邮编：100036
开　　本：787×980　1/16　　印张：40.25　　字数：895 千字
版　　次：2015 年 8 月第 1 版
印　　次：2015 年 8 月第 1 次印刷
定　　价：128.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。
质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。
服务热线：（010）88258888。

推荐序 1

大数据、云计算和人工智能都是热门概念，也是现实问题。很多团队都面对类似的技术抉择：如何用开源软件构架机群、如何选择云服务、如何设计高效的分布式 Web 服务，或者如何开发高效的分布式机器学习系统？当面对这些问题的时候，最好能从一些重要的可度量的方面给出定量分析和比较。可是哪些方面重要，以及如何度量呢？

古往今来有很多重要的系统度量问题，比如“如何度量网络繁忙程度”、“如何得知某台机器还活着吗”、“服务中断是因为某个进程死锁了，还是机器出问题了”，或是“我们的机群中用 SSD 替换硬盘的比例应该多大”，等等。

这些问题的答案都不简单——正确答案往往构建在对操作系统的深刻理解上，甚至构建在统计学和统计实验基础上。可是通常我们是在繁重的业务压力下应对这些问题的，所以只能尽快找一个“近似”测量办法。这终非长久之计——在我们经历了很多问题之后，可能会发现自己从未深刻理解问题，没有深入思考，没有沉淀经验，没有获得成长。

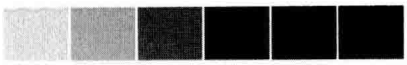
如果有意深入理解问题，借助前人经验是可以事半功倍的。只是关于操作系统的教科书却不能为我们提供足够的基础知识。操作系统是如此复杂，几乎涉及计算机科学的所有方面。学校教育往往只能基于极简化的示范系统，比如 Minix。但实际使用的操作系统，比如 Linux 和 Solaris，有更多需要学习和理解的地方。

我做分布式机器学习系统有八年了，其间很多时候要面对系统分析的问题。但是坦诚地说，大部分情况下我都只能尽快地找一个“近似”方法，处在没有时间深入琢磨上述系统问题的窘境。看到《性能之巅：洞悉系统、企业与云计算》一书之后，不禁眼前一亮。这本书从绪论之

后，就开始介绍“方法”——概念、模型、观测和实验手段。作者不仅利用操作系统自带的观测工具，还自己开发了一套深入分析观测结果的脚本，这就是有名的 DTrace Toolkit（大家可以直接找来使用）。《性能之巅：洞悉系统、企业与云计算》一书介绍的实验和观测方法，包括内存、CPU、文件系统、存储硬件、网络等各个方面。而且，在介绍方法之前会深入介绍系统原理——我没法期望更多了！

很高兴看到这一经典名著的中文版问世。因为被邀作序，有幸近水楼台先得月，深受教益。感谢译者和编辑的努力。相信读者朋友们定能从中受益。

——王益 Linkedin 高级主任分析师



推荐序 2

收到侠少的邀约来写序，多少有点忐忑和紧张。忐忑是因为平时太多时间总是在解决各类问题，没有时间好好沉淀下来读一读书，总结一下发现问题和解决问题的方法；紧张是因为我好像很久没有写字了，上一次写长篇大论还是 N 年前，怀疑自己是不是能够在有限的文字内把系统性能的问题解释清楚。于是，就在这样的背景下，我从出版方那里拿到了书稿，迫不及待地读了起来。

书的作者 Gregg 先生是业内性能优化方面大名鼎鼎的人物，早年在 Sun 公司的时候是性能主管和内核工程师，也是大名鼎鼎的 DTrace 的开发人员，要知道 DTrace 可是众多 trace 类工具中最著名的，并且先后被移植到了很多别的 OS 上。书的全篇都在讨论性能优化，我想了想，我每天从事的工作不就是这个吗？SAE 上成千上万的开发者们，他们每天问的问题几乎全部和性能相关：为什么我的 App 打开比较慢，为什么我的网站访问不了，怎么才能看我的业务哪个逻辑比较慢？对于这些问题，其实难点并不在于解决它，最难的在于发现并定位它，因为只要一旦定位了故障点或者性能瓶颈点，解决起来就并不是很复杂的事情。

对于性能优化，最大的挑战就是性能分析，而性能分析要求我们对于操作系统、网络的性能要了如指掌，明晰各个部分的执行时间数量级，做出合理的判断，这部分在书中有很详细的讨论，让读者可以明确地将这些性能指标应用在 80:20 法则上。

工欲善其事，必先利其器，了解系统性能指标后，就需要找到合适的工具对可能存在的瓶颈进行分析，这要求我们具备全面的知识，涉及 CPU 性能、内存性能、磁盘性能、文件系统性能、网络协议栈性能等方方面面，好在本书详细介绍了诸如 DTrace、vmstat、mpstat、sar、

SystemTap 等工具利器，如何将这些工具组合，并应用在适合的场景，是一门学问，相信读者会在书中找到答案。顺便说一句，Dtrace+SystemTap 帮助 SAE 解决过非常多的性能疑难杂症，一定会对读者的业务分析产生帮助！

单个进程的性能分析是简单的，因为我们可以定位到 system-call 或者 library-call 级别，然后对照代码很快解决，但整个业务的性能分析是复杂的，这里面涉及多个业务单元、庞大的系统组件。最麻烦的是，往往造成性能问题的还不是单元本身，而是单元和单元相连接的网络服务。这就要求我们必须要有一套科学的分析方法论，来帮助我们找到整个系统业务中的瓶颈所在。书中就此介绍了包括随机变动、诊断循环等多个方法，并且介绍了涉及分析的数学建模和概念。不要忽视数学在性能分析中的作用，在实际应用中，我就利用方差和平均值的变化规律科学地分辨一套系统到底是否应该扩容。

找到了性能瓶颈，下一步就是解决问题。当然解决问题的最好办法就是改代码，但是，在你无法短时间内修改代码的时候，对系统进行优化也可以实现这一目的。这就要求我们对于系统的各个环节都要明白其工作原理和联系。本书第 3 章详细讨论了操作系统，这对于读者是很有用的。因为很多时候我们在不改代码的情况下优化系统，就是优化内存分配比例，就是优化 CPU 亲密度，就是尝试各种调度算法，就是做操作系统层面的各种网络参数调优。

对于上述所有问题的认识，我相信读者在通读全书后会有不一样的感觉。记住，不要只读一遍，每读一遍都必有不同的体会。不多说了，我要赶紧再去读一遍:）

——丛磊 新浪 SAE 创始人/总负责人

推荐序 3

人类正在用软件重构这个世界。从上世纪四十年代电子计算机出现，到个人电脑风靡、互联网大行其道，再到如今正兴旺的云计算加移动互联网，还有起步中的物联网，所有这些表面上看是计算机硬件变得无处不在，而实质上是软件一步步掌管起了这个世界。噫吁嚱，短短七十几年，软件轻松征服世界！渗透渗透再渗透，已经进入所有人的生活。

不言而喻，软件对这个世界和人类的重要性也越来越大。我很负责任地说，软件的健康与否关系着世界的安危。君不见，几多时，一个软件漏洞便让全球惊慌。不经意间，我们与软件的关系变得休戚与共。

不幸的是，软件的总体健康状况并不乐观，问题很多。

瑕疵（Bug）是软件行业的一个永恒话题，是破坏软件质量的头号大敌。但迄今为止，完全发现和彻底消除瑕疵还只是奢望，是不可能实现的目标。换句话来说，掌管着我们生活的所有软件都是在带着瑕疵运行，真正的带“病”工作。因为每个软件内部都有纵横交错的无数条道路，CPU 经常奔跑的那些路径上的瑕疵早被发现和移除，所以软件大多时候并不发“病”。但也有时候，CPU 会遭遇软件中的瑕疵，发生意外。目前，我们能做的只有努力多发现瑕疵，并尽可能找到根源，将其去除。但这并不是件容易的事情。

除了瑕疵，性能问题是威胁软件健康的另一个大敌。简单来说，我们把软件中的错误归入瑕疵一类。把那些在速度、资源消耗、工作量等方面的不满意表现纳入性能问题一类。用员工考核做比喻，瑕疵是把一件事做错了，而性能问题是虽然做对了，但是做得不够好，可能开销太大，用的资源太多，可能完成速度太慢，用的时间太久，可能完成的工作量太少，活干的不够多，总之是结果不令人满意，还有必要改进。

举例来说，某支付软件在性能方面就存在问题。一旦运行，会频繁触发大量的缺页异常，消耗的 CPU 时间过多，导致不必要的能源浪费。随着软件变得无处不在，软件的耗电问题已经引起越来越多的关注。在笔者写作这几行文字时，该软件的几个进程仍在一刻不停地触发着缺页异常，过去两天的累积数量已经超过千万，消耗 CPU 的净时间已经超过一小时。粗略估计，这几个进程至少已经消耗了 0.01 度电。不要忽视 0.01 这个看似微小的数字，把它乘以全国的总用户数，立刻就变成一个庞大的数字了。

与软件瑕疵类似，性能问题也可能危害巨大！更可怕的是，性能方面的问题容易触发隐藏在软件深处的瑕疵，直接导致软件崩溃或者其他无法预计的故障。

发现瑕疵根源的过程一般称为调试（Debug）。纠正性能问题，提高软件性能的努力被称为调优（Tune）。不论调试，还是调优，都不是简单的事。对软件工程师的技术要求很高。一些复杂的问题，常常需要多方面的知识，需要对系统有全面了解，既有大局观，能俯瞰全局，又能探微索隐，深入到关键的细节，可谓是 “致广大而尽精微”。

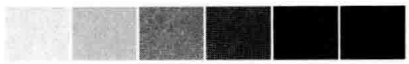
如果一定要把调试和调优的难度比一下，调优的难度更大。简单的解释是，调试的主要目标是寻找瑕疵，瑕疵固定存在软件中的某一个点。因此，调试时可以通过断点等技术把软件静止下来，慢慢分析。而调优必须关注一个动态的过程，观察一段时间内的软件行为。这样一来，调优时常常不可以把软件中断下来静止分析，而需要以统计学的方法或者其他技术对软件做长时间监视。

记得两年前，曾经有一个同行以饱经沧桑的神情问我：“在中国这样的软件环境里，做技术的工程师应该如何发展呢？难道都得像你那样写一本书么？”坦率地说，我当时没能给出让这位同行很满意的回答。因为这个问题确实不太容易回答。此事之后，我常常想起这个困扰着很多同行的问题。多次思考后，我似乎有了个比较好的答案。首先要确认自己是喜欢软件技术的，愿意在技术方向上持续发展。接下来的问题是如何在技术方向上不断前进。“日日新，又日新”。我的建议是，逐级攀登软件技术的三级台阶：编码、调试和调优。

代码是软件的根本，一个好的软件工程师必须过代码这一关，写代码时如行云流水，读代码时穿梭自如，如履平川。以调试器为核心的调试技术是对抗软件瑕疵的最有力工具，是每个技术同行都应该佩戴的一把利剑。调优技术旨在发现软件的性能障碍，让软件跑得更好。随着对软件性能问题的重视，调优技术的发展也越来越快，新的工具层出不穷。调优方面的工作和创业机会也在不断增加。几年前，我写作《软件调试》时，很多人还不太重视调试，但最近几年，软件调试已经逐渐从藏在背后的隐学逐步走向前台，成为一门显学。可以预见，性能调优也会受到越来越多的重视。大家加油！

学习调优技术有很多挑战，很高兴看到有这样一本关于系统优化的好书引进到国内。好友侠少诚邀作序，盛情难却，仓促命笔，词不达意，请诸君海涵，是为序。

——张银奎 资深调试专家，《软件调试》和《格蠹汇编》作者 2015 年 7 月 22 日于上海格蠹轩



推荐序 4

性能调优是每一个系统工程师最重要的技能，也是衡量其水平高低的不二法门。Linux 是开源的操作系统，这也意味着本身可调整范围比较大。近十几年来，硬件设备日益复杂，互联网应用场景及 Web 2.0 蓬勃发展的同时，也带来了各种高并发的业务应用，有些复杂的分布式数据分析系统，单集群的物理服务器数量甚至超过 1 万台。这使得对系统运行环境的一点点优化，带来的收益都可能非常可观。

性能调优这个事情，大家往往都有话想说，技术专家们也都有些秘而不宣、甚至奉为压箱底儿的“活儿”。但这些“活儿”，往往来自 Case By Case 所获得的经验。例如解决了某大型电商网站的 Nginix 服务器问题、某 MySQL 数据库集群性能问题等。这些特定的大型案例，促使参与其中的技术人员，在某个或某些系统性能优化方面，具有较高水平、甚至一定造诣。

但即使这些技术专家，也难以解决所有性能问题。这有两方面原因，一方面自身缺乏对整个系统运行环境的全局把控能力。技术专家们能力的获得，是基于某些问题点扩散开来的，并非基于事先构建好的系统优化的全局观（这也和国内环境有关，大家往往大学毕业后就直接开始从事相关工作，缺少底层、结构性的学习，即使参与了某些培训课程）。

这种系统优化的全局观之所以难以形成，一个原因在于“未知的未知”，也就是说我们不知道自己不知道。比如，我们可能不知道设备中断其实会消耗大量 CPU 资源，因而忽略了解决问题的关键线索。再比如，作为初中级 DBA 并不知道应用程序连接 Oracle 时，每一个数据库连接（Session）实际上都非常消耗物理内存，成百上千个数据库连接长期驻留（看上去状态还是非活动的），PGA 被消耗殆尽，引起各种异常，成为性能问题的罪魁祸首。

另一方面在于性能问题的根源太过复杂。诚如作者所说，一来性能是主观的，连是否有性能问题，都因人而异。例如，磁盘平均读写相应时间为 1ms，这是好还是不好？是否需要调优？实际上取决于开发人员和最终用户（有时还包括领导）的性能预期。二来系统是复杂的。例如，本来 CPU/磁盘/内存各司其责，有了内存缓存（SWAP）机制，内存不够时可以使用部分硬盘空间来顶替。这看上去很好，但对于数据库系统而言，SWAP 是否启用，本身就是一个问题。再比如，对于云计算而言，多虚拟机共享物理机，这进一步增加了问题的复杂度。资源隔离是个技术活，现有技术很难做到磁盘 I/O 完全隔离。另外，最近非常流行的容器技术，如 Docker 等，让问题变得尤为复杂。容器即进程，不像 KVM 等虚拟机（KVM 至少还进行资源隔离）自带操作系统。容器并不是为 IaaS 而生，仅靠 cgroup 等隔离技术能做的非常有限。三是有可能有多个问题并存。有时终端用户抱怨系统慢，很可能不仅是由单个原因引起，例如，业务负载猛增，内网 1000Mps 其实已经不够，但没引起注意；或是整体对外交付能力貌似还正常，但数据库磁盘 I/O 非常繁忙；还可能正偷偷地进行大量 SWAP 交换。

这两方面原因，使得大部分技术专家，即使对系统优化的某些领域确有独到见解，但说到能否解决所有系统性能问题，其实会有些底气不足。但本书作者看起来是个例外。纵观全书，作者建立了系统性能优化的体系框架，并且骨肉丰满，很明显，他不仅擅长某方面的性能优化，而且是全方位的专家，加之作为 DTrace（一种可动态检测进程等状态的工具）主要开发者，使得本书的说服力和含金量大增。

本书首先提及性能优化的方法论和常见性能检测工具的使用，具体内容更是涉及可能影响 Linux 系统性能的方方面面，从操作系统、应用程序、CPU、内存、文件系统、磁盘到网络，无所不包。在以上这些话题的探讨中，作者的表述方法值得称道——每个章节都程式化地介绍术语、模型、概念、架构、方法、分析工具和调优建议，这对于由于长期工作形成一定强迫症的某些技术人员，如我自己，阅读起来赏心悦目，也从侧面体现了作者的深厚功底和驾驭能力。

本书提供的性能优化方法论也令人印象深刻，包括几种常见的错误思考，如街灯法、随意猜测法和责怪他人法。街灯法来自一个著名的醉汉的故事——醉汉丢了东西，就只会在灯光最亮的地方着手。这种头疼医头脚痛医脚、错把结果当原因的事情，相信很多人都过类似经验。大型业务系统上线，大家都围着 DBA 责问数据库为什么崩溃了。但数据库出问题是结果，数据库本身一定是问题根源么？是否更应该从业务负载、程序代码性能、网络等方面联合排查？在列举各种不正确的方法后，作者建议采用科学法，科学法的套路是：描述问题→假设→预测→试验→分析。这种办法的好处是，可以逐一排除问题，也可以降低对技术专家个人能力和主观判断的依赖。

本书用单独的章节系统性介绍了操作系统、性能检测方法和各种基准测试，特别是作者主导开发的 DTrace（本书例子中用 DTrace 监控 SSH 客户端当前执行的每个命令并实时输出）。

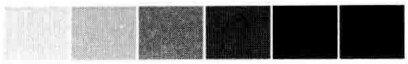
这使得本书作为工具书的价值更得以彰显。云计算的出现，对系统优化带来新的挑战。作者作为某云计算提供商的首席性能工程师，带来一个真实的云客户案例分析，包括如何利用本书提及的技术、方法和工具，一步一步分析和解决问题。

很多时候，受限于语言障碍，系统工程师往往通过国内 BBS、论坛等获得知识，只是在性能问题确实棘手时，被迫找些英文资料，寻找技术解决思路。

博文视点推出的本书中文版，对于国内广大运维同仁而言，实属幸事。这让我们有机会系统学习和掌握性能优化的各个方面，有机会建立一种高屋建瓴的全局观。这样，在面对复杂系统问题时，不至于手足无措，或只能盲人摸象般试探。另外，虽然面对日益复杂的硬件设备和高并发的业务应用，问题不是变得简化而是更为复杂，但 Linux 系统演化至今，其最基础的体系架构和关键组件并未发生多大改变，这使得这本好书即使历经多年，价值毫无衰减，反而历久弥新。

总的来说一句话：如果早些接触到本书，该有多好！

——萧田国　触控科技运维总监　高效运维社区创始人



推荐序 5

性能的话题，从一开始就是复杂的。性能是一种典型的非功能需求，然而又贯穿在任何一种功能需求中，直接影响系统运行效率和用户体验。也正是由于这一特性，性能无法简单地通过单一的、直线式的思维来度量和管理，而注定需要以系统工程的方法来掌握和调整。绝大多数的图书在谈到性能问题的时候，都是仅从片面的若干现象出发来触及问题的冰山一角，抑或干脆语焉不详甚至避而不谈。这也难怪，因为这个话题一旦展开，就会占用极大篇幅，相对于原先的论题而言就显得喧宾夺主。然而更重要的原因，也在于对性能问题有着全面认识，并且能够给出一个系统化的分析和全栈式的论述的作者实在不多。相关的要求近乎苛刻：既要对系统的每一个部件都了如指掌，又要深入理解部件之间的协作方式；既要精通系统运行的细节，又要明白取舍逻辑的大局观；既要懂得现象背后的原理，又要把握从开发部署的工作人员直至终端应用用户的需求乃至心态。

《性能之巅：洞悉系统、企业与云计算》以一种奇妙而到位的方式，把高屋建瓴的视角和脚踏实地的实践结合了起来，对性能这一复杂、微妙甚至有些神秘的话题进行了外科手术式的解析，读来真是让人感觉豁然开朗。

全书以罕见的遍历式结构，对软件系统的每一个部件都如庖丁解牛般加以剖析，几乎涉及业务的每一个细节。然而，这些细节并非简单的罗列，而是每一段论述都与具体的角色和场景紧密结合，取舍之间极见智慧。方法论更是不单说理，而是通过一个又一个的具体实例，逐步地建构起来，并反复运用于各个部件之上，使读者明白原理普适性的同时也知道怎样举一反三。

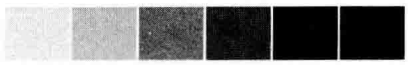
本书也是难得的 UNIX/Linux 系统管理员和运维工程师的百科全书式参考手册，相对于工

作于 Windows 上的同行而言，他们获得的知识更加零碎，甚至很多场合下不得不求助于网络上的只言片语，并只能通过耗时的、高风险的生产环境实验来取得一手经验数据。本书当然提供了不少趁手的软件工具供人使用，然而其更大的价值在于心法的传授，即怎样利用工程师现在就熟悉、现在就可用的工具来迅速地进行性能建模，完成故障排除和调优的关键步骤。书中的内容非常新，作者见过大世面，是从最与时俱进的大型云计算系统为出发点来落笔的，对付日常的性能问题完全没有压力，即使最新的硬件也能找到对应的解决方案。

本书的译者团队阵容强大，皆是在底层系统有多年一线工作经验的运维工程师和开发工程师。徐章宁同志几乎是以一己之力支撑起 PB 级数据运维的明星 DevOps，而另外两位也都是手工实现过复杂生产环境中文件系统和网络协议的大牛。可以说，他们对于性能的认识是经过多年实际工作的考验的，是深刻而且务实的，这为本书翻译在专业性方面提供了坚实的保证。加上他们多年养成的认真严谨的工作习惯，和深厚的中文功底，更是为该译本的可读性锦上添花。

希望所有的 IT 从业者都能从本书受益，让天下的系统都能达到性能之巅！

——高博 青年计算机学会论坛（YOCSEF）会员，文津奖得主，《研究之美》译者



推荐序 6

性能问题一直是个热门话题，在单机时代我们就已经投入了不计其数的人力物力进行研究。但是随着互联网行业的发展，分布式系统开始大量投入应用，对于性能问题的分析、调优提出了很多前所未有的新挑战。特别是如何做到单机性能与集群整体性能的平衡，以及在各种影响性能的要素之间进行取舍，成为摆在开发运维人员面前的巨大难题。

本书采用了自下而上的结构，从底层的操作系统、CPU、磁盘等基础元素开始，到工作原理层面分析性能受到的各种不同影响，以及如何评估、衡量各项性能指标，让读者知其所以然，在面对实际情况时能够更有针对性地做出判断和决定，而不是机械地、教条地行事。本书还提供了案例，手把手地展示了实际性能问题的排查调优过程。读者可以根据案例，结合业务系统实际情况展开工作。此外，本书还对常用的性能分析工具的使用和扩展做了详细介绍，对于日常工作效率的提升也有着很大的帮助。

译者徐章宁曾与我在 EMC 的云存储部门共事多年，在系统性能排查、调优方面有着丰富的经验，很高兴他能参与此书的翻译。审稿过程中我感觉译者不仅忠实地还原了原著的精华，也融合了自己多年工作中积累的经验教训。

我坚信，本书无论对于开发还是运维人员，无论对于设计、编码或者排查调优工作，都能发挥重要的参考作用，尤其适合常备案头。在此诚挚向大家推荐。

——林应 淘宝技术部高级技术专家

译者序

作为一名运维工程师，系统出现一些“诡异”问题的情况并不罕见。有些时候面对束手无策的问题着实让人头痛，这时我总会感慨，学生时代课本上计算机科学那些诸多的概念和理论所呈现出的完美感觉更多是在书本上，在真实系统中往往更多是另外一幅更为“现实”的景象。工作多年后，我自发形成了一个简单的认知：当系统庞大到一定程度时，其复杂性会变得不容易控制是一件很正常的事情。用技术手段把这些“失控”的点妥善摆平就是工程师价值的体现。在翻译本书的过程中，我对这个问题又有了不同的认识：

“已知的已知，已知的未知，未知的未知。”

这是本书多次提到过的概念，说的是有些事情我们知道自己知道，有些事情我们知道自己不知道，还有些事情我们不知道自己不知道。这个概念就系统（特别是复杂系统，诸如云计算或大数据）而言，特别贴切：就系统内部来说，无论用到的是某一操作系统，还是某一编程语言，其实本身就已经是复杂度较高的实体，要透彻掌握并非易事，何况系统皆由这些技术组合构建而成，方方面面无所不知是不可能做到的事情，这里的未知源于技术本身的复杂；就系统外部来说，如今时事变化一日千里，现在系统要处理的外界变化，可能最初的系统设计者都从未想过，这里的未知来源于未来的不定。所以，有让你手足无措的问题出现其实是一种很正常的状态，对此的恐惧只是人施加给自己的情感层面的东西。与此相反，始终对未知心生敬畏才是对待未知正常的态度，更是本应有的觉悟。

这里并不是说我们要对未知“投降”，而是说对“未知”有正确的认识才是我们取得进步的前提。其实我们多数人对“系统的未知”存在误区，我们常常将系统等同于具象的技术实体，

例如某种编程语言、OS 内核、网络。总觉得系统出问题肯定是我某些技术知识有漏洞没学好。可惜“学海无涯，而吾生有涯，以有涯随无涯，殆矣”，拘泥于各种眼花缭乱的技术只会让自己迷失造成时间的浪费。技术都是末节，真正要把握的主体其实只是系统本身。道理虽简单，但舍本逐末的事情却还是屡见不鲜。

要做好这一点，首要的是要有全局的系统观。更准确地说，是要有对系统各层知识的理解和实践能力，要有对系统架构的认知和理解的能力。只有对系统了如指掌，才有希望将已知的未知转化为已知的已知，将未知的未知转化为已知的未知，进而增加对系统的掌控能力，避免盲人摸象的悲剧。事实的真相是也本应是这样：技术的价值依附于系统及其价值，没有孤立存在的技术，一切价值的体现都在于系统本身。唯有立足系统本身，工程师才能打通性能这条经脉，领略到性能之巅的风采！

回观《性能之巅》这本书，全书的安排也深有此意。本书第 1~5 章可谓是精华部分，讲述了与系统性能相关的通用模型和通用方法；第 6~12 章才落实到具体的知识细节，讲述各性能组件（CPU、内存等）的知识（第 6~10 章）、云计算的基础（第 11 章），基准测试的方法（第 12 章）。本书最后一章是一个性能分析的实例，若是让长期与系统打交道的工程师读来，必然感同身受；若是性能新手阅读，则可以对性能工作的日常状况有个基本的了解。

本书是一座桥梁，作者 Brendan 是在系统性能领域耕耘多年的技术专家，在 Sun 和 Oracle 公司有过卓越的贡献，动态跟踪工具 DTrace 就是他主力开发的，他用自己多年的经验和实践归纳并总结出了系统性能的理论和方法，这些理论和方法的作用就像桥梁，把业界可用的工具（或是你自己开发的工具）与系统内部的原理机制联通让它们有机地结合起来，让与性能相关的工作（无论是性能分析还是性能调优）做到有的放矢、有章可循！这与单纯提供知识的技术书籍截然不同，“授人以鱼不如授人以渔”，其立意确实难能可贵。

现代 IT 技术的源头并非中国，但 IT 技术在这片土地上生根发芽，欣欣向荣。如今国人日常生活中所依赖的系统服务已经比比皆是，不信者打开自己的手机数数所装的 App 自然清楚，这些 App 背后多半都有远在某个数据中心的一个或多个系统作为支撑。随着互联网技术向各行业以及生活各方面的渗透，这样的系统今后会越来越多。加之伴随着云计算和大数据技术的兴起和蓬勃发展，除了系统越来越多之外，系统自身还会变得越来越庞大和复杂。在这么一个总的大趋势下，系统性能的重要性自然不言而喻。你会发现 Brendan 所著的《性能之巅》是如此地契合我们这个时代，本书不是第一本论述系统性能的书，但本书对现有系统性能的方法和理论所做的提炼、概括和归纳，不敢说后无来者，但绝对可以称得上是前无古人的了。

全书翻译由 EMC 资深软件工程师吴寒思、点融网资深运维工程师陈磊与我共同完成，在此感谢二位的辛苦耕耘和我们作为团队三人之间彼此的精诚合作，一年多的翻译历程，大量的时间和精力的投入自是不提，但回过头来看整个过程于我们译者自觉仍是获益良多的。本书内