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”。

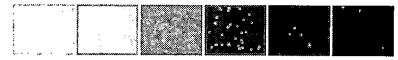
如果一定要把调试和调优的难度比一下，调优的难度更大。简单的解释是，调试的主要目标是寻找瑕疵，瑕疵固定存在软件中的某一个点。因此，调试时可以通过断点等技术把软件静止下来，慢慢分析。而调优必须关注一个动态的过程，观察一段时间内的软件行为。这样一来，调优时常常不可以把软件中断下来静止分析，而需要以统计学的方法或者其他技术对软件做长时间监视。

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）。

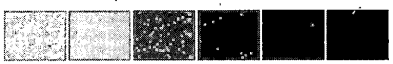
这使得本书作为工具书的价值更得以彰显。云计算的出现，对系统优化带来新的挑战。作者作为某云计算提供商的首席性能工程师，带来一个真实的云客户案例分析，包括如何利用本书提及的技术、方法和工具，一步一步分析和解决问题。

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