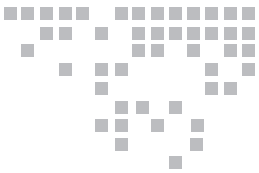

第 1 章 Chapter 1

性能追踪建议

没有远见和规划，这样解决性能问题是痛苦的。产生问题的原因一次又一次从指尖溜走，不仅浪费时间并且让你备感受挫。但是，如果按照正确的步骤，就可以把令人沮丧的性能追踪转变为有趣的侦探故事。每一条信息都让你更接近问题的根源。人不可能总是可信的，证据将是你唯一的朋友。当你开始研究问题的时候，会遇到不寻常的波折，追踪之初发现的信息最后可能会帮你解决问题。最棒的部分就是，当你最终逮住“坏小子”并修复问题时，会感觉到肾上腺素的刺激和成就感。

如果你从来没有调查过性能问题，那么第一步会是决定性的。不过，听从下面几个明显或隐晦的建议，可以节约时间，并按照自己的方式来找出性能问题的原因。本章目标是提供一系列建议和指导来帮助读者追踪性能问题。在研究系统或应用程序出现的问题时，这些建议告诉你怎样避开一些常见的陷阱。这些建议，大多数都是从浪费的时间和令人沮丧的死胡同中辛苦得到的教训，它们有助于快速并有效地解决你的性能问题。

阅读本章后，你将能够：

- ❑ 避免重复他人的工作。
- ❑ 避免重复自己的工作。
- ❑ 避免因收集的误导信息而导致的虚假线索。
- ❑ 为你的研究创建有用的参考文档。

尽管所有的性能调查都是有瑕疵的（“如果一开始就能想到”是你的口头禅），但这些建议将会帮助你避免性能研究中的一些常见错误。

1.1 常用建议

1.1.1 记大量的笔记（记录所有的事情）

在调查性能问题时，你可以做的最重要的事情大概就是记录下看到的每一个输出、执行的每一条命令，以及研究的每一个信息。结构清晰的记录能让你只查看记录就可以检验关于性能问题原因的猜想，而不是重新运行测试。这能节约大量时间。写下来并且创建性能记录。

在性能调查之初，我通常会为其创建一个目录，在 GNU Emacs 中打开一个新的“Notes”文件，开始记录系统的信息。之后，将性能结果保存到这个目录，并将有意思的和相关的信息保存到 Notes 文件。建议将下面的内容添加到你的性能调查文件和目录中：

- 记录硬 / 软件的配置情况——记录下的信息包括硬件配置（主存容量、CPU 类型、网络和磁盘子系统）和软件环境（OS 和软件的版本、相关配置文件）。这些信息看上去很容易在之后重现，但是在追踪问题时，你可能会大幅度地修改系统配置。认真细致的笔记有助于在特定的测试过程中弄清楚系统配置。

  示例：每次测试时，保存 cat /proc/pci、dmesg 和 uname -a 的输出。
- 保存并组织性能结果——运行很长时间后还能评估性能结果是很有价值的。记录系统配置的同时，也记录测试结果。这使你得以比较不同的配置是如何影响性能结果的。如果需要，可以重新运行测试，但是测试一种配置是耗费时间的过程。只需让笔记保持条理清晰，避免重复工作则效率更高。
- 写下命令行调用——在运行性能工具时，常常需要用困难复杂的命令行来准确定位到你感兴趣的系统区域进行测量。如果想重新测试，或在不同的应用程序上运行相同的测试，那么，复制这些命令行不仅令人厌烦，并且在初次尝试时，不容易做对。更好的办法是准确记录下你键入的信息。这样在之后的测试中就能够完全重现命令行，而在回顾之前测试结果时，也可以看到你测量的内容。Linux 命令 script（详见第 8 章）或者从终端“剪切粘贴”都是完成这项工作的好方法。
- 记录研究信息和 URL——调查性能问题时，将在互联网上发现的相关信息记录下来是很重要的，不论发现的途径是电子邮件，还是人际交往。如果你找到一个看上去相关的网站，就把它剪切粘贴到你的笔记中。（网站是会消失的。）当然，还要记录下 URL，因为你可能在之后还要查看这个网页，或者网页所指信息对后面的调查会变得重要起来。

在收集和记录所有这些信息时，你可能会疑惑：这样做值得吗？有些信息眼下显得毫无作用或有误导性，但它在将来可能是有用的。（好的性能调查就像一部优秀的侦探剧：尽

管开始的时候所有的线索都令人迷惑，但最终都会真相大白。）在调查问题时，请牢记以下几点：

- 结果的含义可能是不明确的——性能工具给你的信息并不总是清晰明了的。有的时候，你需要更多的信息才能理解某个结果的含义。之后，你可以回过头以新的视角重新审视那些看似无用的测试结果。实际上，旧信息可能会驳斥或者证明关于性能问题本质的某个特定理论。
- 所有的信息都是有用的（这也就是你要记录的原因）——记录已运行的测试信息以及系统配置信息的原因不见得会立即明晰。这一点在你试图向开发人员或管理人员解释系统性能不佳的原因时是非常有用的。通过记录和整理调查过程中你所见的一切，你就有证据支持特定理论，同时也具备大量的测试结果来证明或驳斥其他理论。
- 定期回顾你的笔记可以得到新的想法——当你为性能问题积攒了大量的信息时，那就定期回顾它们。重新审视会让你关注结果，而不是测试。当许多测试结果放在一起被同时查看时，问题的原因也许就会自动浮现。回顾你收集的数据，就可以在不实际运行任何测试的情况下进行理论检验。

在调查问题时，重做一些工作虽然是不可避免的，但是，在重做工作上花费的时间越少，你的效率就越高。如果你写了大量的笔记，并有办法在发现信息时记录它们，那么你就可以依赖已经做过的工作，而避免重复运行测试以及重复研究。保持笔记的可靠性和一致性，从而节省时间减少挫折。

例如，在调查性能问题后，最终确定为硬件原因（主存慢、CPU 慢等），你可能会想通过升级慢速硬件，并重新运行测试来检验这个想法。通常要花一点时间才能获得新硬件，而在你可以重新运行测试之前可能已经过了很久。当最终可以开始的时候，你想要在新老硬件上运行同样的测试。如果你已经保存了之前的测试调用和测试结果，那么，你马上就知道要怎样为新硬件进行测试设置，同时也可以比较新的结果与保存的旧结果。

1.1.2 自动执行重复任务

当开始调整系统改进性能时，键入复杂命令行很容易出现错误，而无意中使用的不正确参数和配置则会产生误导性的性能信息。因此，自动执行性能工具调用和应用程序测试是一个好办法：

- 性能工具调用——有些 Linux 性能工具的命令行相当复杂，给自己省点力，把它们保存到一个 shell 脚本中，或是将所有命令都放到可以进行剪切粘贴的参考文件中。这可以让你少受些挫折，并且让你多少有些信心：用来调用工具的命令行是正确的。
- 应用程序测试——大部分应用程序有着复杂的配置，要么通过命令行，要么通过配置文件。你会经常重新运行要多次测试的应用程序，若将调用保存为脚本，就能少

走弯路。虽然刚开始的时候，键入 30 个字符的命令看上去很容易，但这样的操作重复 10 次后，你就会向往自动执行了。

尽可能多地自动执行，就能减少错误。使用脚本自动执行，可以节省时间，并有助于避免因不当工具和测试调用造成的误导性信息。

举个例子，你想在特定工作负载下或某段时间内监控系统，但是在测试结束时，你可能不在现场。这种情况下，脚本就很好用了，在测试完成时，可以自动收集、命名、保存全部生成的性能数据，并将它们自动放到“Results”目录中。有了这些基础之后，你就能按照不同的优化和调整重新运行测试，并且不用担心数据是否已经保存好。你反而可以集中精力找出问题的原因，而不是去管理测试结果。

1.1.3 尽可能选择低开销工具

一般情况下，观察系统会修改系统的行为。（对物理爱好者来说，这就是海森堡（Heisenberg）不确定性原理。）

具体而言，在使用性能工具时，它们会改变系统的行为方式。调查问题的时候，你想要看看应用程序是如何执行的，同时还必须处理性能工具引发的错误。这是不可避免的弊端，但是你要知道它的存在，并努力将其最小化。有些性能工具能够给出高度精确的系统信息，但其检索信息的开销也很高。高开销工具对系统行为带来的变化大于低开销工具。如果你只需要了解系统的粗略信息，那么使用低开销的工具是更好的选择，即使它们不够准确。

例如，对于正在使用的应用程序，工具 ps 能给出其主存数量和类型的相当不错但粗糙的概况。那些准确性更高，但是影响较大的工具，如 memprof 或 valgrind，虽然也能提供这些信息，但是它们消耗的主存和 CPU 资源比只使用原始应用程序要大，因此会改变系统行为。

1.1.4 使用多个工具来搞清楚问题

虽然在找出性能问题原因的时候，如果只需要用一个工具那将是非常方便的，但这种情况相当少见。实际上，你使用的每一种工具都会为问题的原因提供线索，因此，你必须同时使用多个工具来真正搞清楚发生了什么。比如，一种性能工具会告诉你系统存在大量的磁盘 I/O，而另一种工具则告诉你系统使用了大量的交换。如果只以第一个工具的结论制定解决方案，你可能会简单地选择更快的磁盘驱动器（然后发现性能问题仅仅改善了一点点）。而将两种工具的结果放在一起，你就会判断出：大量的磁盘 I/O 是由大量使用的交换造成的。在这种情况下，你可能会买更多的主存以减少交换（这样就不会再有大量的磁盘 I/O）。

比起单一地使用任何一种工具，同时使用多个性能工具通常能让你对性能问题有更清晰的了解。

寓言：盲人摸象

三个盲人在一头大象旁边，想要搞清楚它长什么样子。第一个人拉住了尾巴，说道："大象就像一根绳子。"第二个人摸到了象腿，说道："大象像一棵树。"第三个人摸到了大象一侧的身体，说道："大象像一堵厚实的墙。"

显然，没有一个人得出了正确的答案。如果他们将各自的印象进行共享和组合，那么，他们就可能发现大象真正的模样。别做摸象的盲人。同时使用多种性能工具找出问题的原因。

1.1.5 相信你的工具

性能追踪的过程中，最令人兴奋又令人沮丧的时刻之一，就是工具显示了一个"不可能"的结果。某些"不会"发生的事情却明明白白地发生了。第一反应会认为工具坏了。不要被直觉愚弄了，工具是公正的。虽然它们可能会不正确，但这更有可能是因为应用程序做了不该做的事情。要使用工具来调查问题。

举个例子，Gnome 计算器使用超过 2000 个系统调用只为了实现加载和退出。如果没有性能工具来证明这个事实，那么，仅仅为了启动和停止应用程序就需要如此之多的系统调用看上去是没有必要的。但是性能工具能够显示其发生的位置和原因。

1.1.6 利用其他人的经验（慎重）

在调查任何一个性能问题时，你可能会发现问题令人不知所措。不要独自面对它。问问开发者是否见过同样的问题。试着找到其他解决过你所遇问题的人。在互联网上搜索类似的问题，并希望找到解决方案。给用户和开发人员发电子邮件。

本条建议附带一个提醒：即使开发者认为了解自己的应用程序，他们也不见得总是对的。如果开发者不认同性能工具的数据，那么，他们也许是错的。向开发者展示你的数据以及你为何会得出这样的结论。他们通常会帮你重新解释数据或者解决问题。不论是哪种情况，都会将你的调查向前推进一些。如果你的数据表明发生了不该发生的事情，就不要害怕与开发者有分歧。

比如，你通常可以按照在 Google 上搜索类似问题得到的指导来解决性能问题。很多时候，在调查一个 Linux 问题时，你会发现之前已经有人遇到过了（即使是几年前），而且还在公共邮件列表中报告了解决方法。使用 Google 是很容易的，它可以为你节省几天的工作量。

1.2 性能调查概要

本节列出了开始性能调查时的几个重要步骤。由于终极目标是解决问题，因此最好的方法是在你接触性能工具之前就开始研究问题。遵循如下特定步骤是解决问题的有效方法，并且不会浪费宝贵的时间。

1.2.1 找到指标、基线和目标

性能调查的第一步就是确定当前的性能，并明确其应提升的程度。如果你的系统明显性能不佳，你就可以确定值得花时间进行研究。但是，如果系统性能接近其峰值，那么就不值得研究。明确性能峰值有助于你设置合理的性能期望值，并能给你一个性能目标，这样你就知道何时应该停止优化。你可能总是没有目标地时不时对系统做一点调整，这会浪费大量的时间，只为得到一些额外的性能，即使你可能并不真的需要它们。

1.2.1.1 确定指标

要想知道什么时候结束优化，你必须为系统自行确立或是使用已发布的指标。指标是一种客观的度量，用于指示系统的执行情况。例如，如果你要优化一个 Web 服务器，你就可以选择“每秒服务的 Web 请求数”。如果你没有一个客观的途径来度量性能，那么在调整系统的时候，你几乎无法确定是否取得了进展。

1.2.1.2 确定基线

在你明确了如何度量特定系统或应用程序的性能之后，确定当前的性能等级就很重要了。在调整和优化之前，运行应用程序并记录其性能，这就是基线值，它是性能调查的起点。

1.2.1.3 确定目标

在你确定了性能指标和基线后，现在需要确定一个目标，这个目标引导你完成性能追踪。你可以无限期地调整系统，花费越来越多的时间来获得更加好一点的性能。如果你制定了目标，那么你就会知道什么时候该结束整个过程。要选择合理的目标，下面是一些好的起点：

- 寻找其他有相同配置的人，询问他们的性能指标——这是理想状态。如果你能发现某人拥有相似的系统和更好的性能，则不仅能为你的系统选定目标，还能与这个人一起工作：他可以确定为什么你的配置更慢，以及该配置有何不同。在研究问题时，使用另一个系统作为参照被证明是非常有用的。
- 查找工业标准测试程序的结果——许多网站都比较了计算机系统各方面的基准测试结果。有些基准测试结果是经过异常的努力得到的，因此，它们可能不能代表真实的使用情况。不过，很多基准测试网站给出了特定结果使用的配置，这些配置信息

可以为你调整系统提供线索。

- 在不同的 OS 或应用程序上使用你的硬件——可能要在你的系统上运行不同的软件以实现相似的功能。比如，如果你有两个不同的 Web 服务器，其中一个运行较慢，那么就试试另一个，看它的性能是否好一些。或者，在另一个不同的操作系统上运行同一个应用程序。如果上述任一情况下系统执行表现得更好一些，那么，你就会知道原来的应用程序还有改进的空间。

如果用现有的性能信息来指导你的目标，那么你有更好的机会来选择一个积极的，但也并非不可能达到的目标。

抓住低处的果实

性能追踪的另一种方法是选择在特定时间段内进行追踪，而不是选择一个目标，在这段时间内尽可能地对性能进行优化。如果应用程序从未被优化过，那么通常在给定工作负载下，会有一些问题相对容易解决。这些容易被修复的问题称为“低处的果实”。

为什么是“低处的果实”？打个比方，将性能调查想象成你饿了，正站在一棵苹果树下。你会采摘最靠近地面，也是你最容易够到的苹果。这些低处的苹果与果树稍高处较难够到的苹果一样能够填饱你的肚子，但是采摘它们只需花费很少的力气。相似的，如果你要在有限的时间内优化一个应用程序，你可能会试着修复那些最简单明显的问题（低处的果实），而不是做一些更困难的、根本性的变化。

1.2.2 追踪近似问题

使用性能工具为确定问题原因打开第一个口子。通过初始的粗略尝试，你能对问题形成大致的看法。这个简单切口的目的就是收集足够的信息传递给程序的其他用户和开发者，以便他们提出意见和建议。这里非常重要的一点是要有良好的书面记录来解释你认为问题是怎样的，以及什么样的测试使你得出了这个结论。

1.2.3 查看问题是否早已解决

你的下一个目标是确定是否有其他人已经解决了这个问题。性能调查可能是一个冗长且费时的事情，如果你正好可以利用其他人的工作，那么在你开始之前就将抢占先机。因为你的目的就是要改进系统性能，所以解决性能问题最好的办法就是依靠其他人已有的成果。

尽管你很可能对性能问题的具体建议持保留态度，但是这些建议具有启发性，可以使你了解到其他人可能已经研究过相似的问题，他们是如何试着解决问题的，以及他们是否成功了。

下述这些地方也能寻求性能建议：

- 在 Web 上查找相似的错误信息 / 问题——这通常是我调查的第一步。Web 搜索常常

会揭示很多与应用程序以及你正在查找的具体错误情况相关的信息。它们还可以指向其他用户所尝试的系统优化，还可能提示哪些有作用、哪些没有作用。成功的搜索可以产生好几页能够直接用于你的性能问题的信息。要发现有相似性能问题的人，使用 Google 或 Google 论坛搜索是非常有用的方法。

- 在应用程序邮件列表上求助——大多数流行或公开开发的软件都有软件使用者的邮件列表，这是寻找性能问题答案的绝佳地方。读者和贡献者通常在软件运行以及保持其性能良好方面有经验。搜索一下邮件列表的存档，因为有人可能会问过相同的问题。而随后对原始消息的回复也许就描述了一个解决方案。如果没有这样的回复，就向最初提出这个问题的人发邮件，询问他是否找到了解决方法。如果这样也不行，或是没有其他人提出过相似的问题，那么在列表上发邮件说明你的问题，幸运的话，也许已经有人把这个问题解决了。
- 向开发人员发邮件——很多 Linux 软件在文档的某个位置包含了开发者的 e-mail 地址，如果在互联网和邮件列表中搜索失败，你可以尝试直接给开发者发邮件。开发人员通常非常忙，不见得有时间回复邮件。但是，相比其他人，他们更加了解软件，如果你能向开发者提供对性能问题条理清晰的分析，并愿意和他一起工作，那么开发者也许能帮助你。虽然开发者关于性能问题原因的想法不见得正确，但是他们可能会给你指出一个富有成果的方向。
- 与内部开发人员交谈——最后，如果产品是内部开发的，你就可以与内部开发人员通电话或发邮件。这和与外部开发者联系几乎是一样的，只不过内部人员可能会在你的问题上投入更多时间，或是给你指出内部的知识库。

依靠其他人的工作，你也许在性能调查开始之前就能解决问题。至少，你有可能找到一些有希望的方法来调查，所以，最好总是看看别人有什么发现。

1.2.4 项目开始（启动调查）

现在你已经详细了解了别人解决问题的可能性，接下来必须开始性能调查了。后续章节将详细介绍工具和方法，但是现在还有一些提示能让你的工作效果更好：

- 分离问题——如果可能的话，删去任何运行于被调查系统的多余的程序或应用。运行许多不同应用程序的系统，其负载较重，会影响性能工具收集信息的准确性，并最终将你引导到错误的方向。
- 利用系统差异发现原因——如果你能发现一个相似的系统具有更好的性能，那么这对问题调试将是一个有力的帮助。使用性能工具的问题之一就是，你不一定有好的方法知道性能工具的结果是否指明了问题。如果你有一个好的系统和一个差的系统，你就可以在这两个系统上运行同样的性能工具，并比较它们的结果。如果结果不同，

就可以通过找出系统差异来确定问题的原因。

- 一次只改变一件事——这点非常重要。要真正确定问题出在哪儿，一次只能有一个变化。这可能会很花时间，并让你运行多个不同的测试，但它的确是发现你是否解决了问题的唯一途径。
- 始终在优化后重新测量——如果你稍稍调整了系统，那么在调整后对所有的事情重新进行测量是很重要的。当你开始修改系统配置时，所有之前生成的性能信息可能不再有效。通常，在你解决一个性能问题时，别的问题会随之而来。新问题可能与老问题有着极大的不同，因此，你真的需要重新运行性能工具来确保正在调查的问题没有出错。

遵循这些建议能帮助你避免误导，并有助于确定性能问题的原因。

1.2.5 记录，记录，记录

如前所述，记录你所做的事情以便之后回顾和审查，这一点确实很重要。如果你已经开始追踪性能问题，那么在你的脑海中就会有大量新增的笔记和 URL。它们可能杂乱无章，混成一团，但现在你明白它们的意思，知道它们的组织结构。在解决问题后，花些时间重写你的发现以及为什么你认为这么做是对的。包括测量得到的性能结果和做过的实验。虽然看上去工作量很大，但却是非常值得的。几个月后，曾经做过的测试很容易就会被忘记，如果没有将结果记录下来，最终你可能会重做测试。如果在这些测试还记忆犹新时撰写了报告，你就不用重做这些工作，而只需要依靠这些记录就行。

1.3 本章小结

追踪性能问题应该是个令人满意且兴奋的过程。如果用正确的方法去研究和分析，你的问题追踪将会事半功倍。首先，确定是否有其他人遇见过相似的问题，如果有，尝试他们的解决方案。要对他们告诉你的保持怀疑，要寻找具有类似问题经验的人。为你的性能追踪设立合理的指标和目标，指标使你知道什么时候应该结束追踪。自动执行性能测试。生成测试结果和配置信息时，要确保将它们记录下来，以便之后可以审查这些结果。保持结果的条理性，记录下任何与你的问题相关的研究和其他信息。最后，定期回顾你的笔记，找出之前可能被漏掉的信息。如果遵循了这些原则，你的问题调查将会有一个明确的目标和一个清晰的过程。

本章给出了性能调查的基本背景，后续章节将会覆盖 Linux 特有的性能工具。你将学习如何使用工具，它们可以提供什么类型的信息，以及如何组合使用它们找出特定系统的性能问题。