第5章

C H A P T E R 5

优化程序性能

写程序最主要的目标就是使它在所有可能的情况下都正确工作。一个运行得很快但是给出错误结果的程序没有任何用处。程序员必须写出清晰简洁的代码，这样做不仅是为了自己能够看懂代码，也是为了在检查代码和今后需要修改代码时，其他人能够读懂和理解代码。

另一方面，在很多情况下，让程序运行得快也是一个重要的考虑因素。如果一个程序要实时地处理视频帧或者网络包，一个运行得很慢的程序就不能提供所需的功能。当一个计算任务的计算量非常大，需要执行数日或者数周，那么哪怕只是让它运行得快20%也会产生重大的影响。本章会探讨如何使用几种不同类型的程序优化技术，使程序运行得更快。

编写高效程序需要做到以下几点：第一，我们必须选择一组适当的算法和数据结构。第二，我们必须编写出编译器能够有效优化以转换成高效可执行代码的源代码。对于这第二点，理解优化编译器的能力和局限性是很重要的。编写程序方式中看上去只是一点小小的变动，都会引起编译器优化方式很大的变化。有些编程语言比其他语言容易优化。C语言的有些特性，例如执行指针运算和强制类型转换的能力，使得编译器很难对它进行优化。程序员经常能够以一种使编译器更容易产生高效代码的方式来编写他们的程序。第三项技术针对处理运算量特别大的计算，将一个任务分成多个部分，这些部分可以在多核和多处理器的某种组合上并行地计算。我们会把这种性能改进的方法推迟到第12章中去讲。即使是要利用并行性，每个并行的线程都以最高性能执行也是非常重要的，所以无论如何本章所讲的内容也还是有意义的。

在程序开发和优化的过程中，我们必须考虑代码使用的方式，以及影响它的关键因素。通常，程序员必须在实现和维护程序的简单性与它的运行速度之间做出权衡。在算法级上，几分钟就能编写一个简单的插入排序，而一个高效的排序算法程序可能需要一天或更长的时间来实现和优化。在代码级上，许多低级别的优化往往会降低程序的可读性和模块性，使得程序容易出错，并且更难以修改或扩展。对于在性能重要的环境中反复执行的代码，进行大量的优化会比较合适。一个挑战就是尽管做了大量的变化，但还是要维护代码一定程度的简洁和可读性。

我们描述许多提高代码性能的技术。理想的情况是，编译器能够接受我们编写的任何代码，并产生尽可能高效的、具有指定行为的机器级程序。现代编译器采用了复杂的分析和优化形式，而且变得越来越好。然而，即使是最好的编译器也受到*妨碍优化的因素*(optimization blocker)的阻碍，妨碍优化的因素就是程序行为中那些严重依赖于执行环境的方面。程序员必须编写容易优化的代码，以帮助编译器。

程序优化的第一步就是消除不必要的工作，让代码尽可能有效地执行所期望的任务。这包括消除不必要的函数调用、条件测试和内存引用。这些优化不依赖于目标机器的任何具体属性。

为了使程序性能最大化，程序员和编译器都需要一个目标机器的模型，指明如何处理指