* 作者建议开发者不要过度关注处理器层面的优化，而是应该专注于编写通用、高效的代码。
* 对于大多数应用，跨平台的性能和兼容性比针对特定处理器的极致优化更重要。

优化的目标是改善正确程序的行为，使其满足客户对速度、吞吐量、内存占用、功耗等方面的需求。因此，优化对于开发过程来说与编码功能一样重要。对于用户来说，无法接受的糟糕性能与错误和缺少功能是同一类问题。

优化是有效的

通过调整代码实现超过10倍的加速是不太可能的。然而，选择一个更好的算法或数据结构可能是一个功能是否可以部署或慢得不可行的区别。

**实验是打破“普遍认知”的关键**。通过实验和性能测量工具，开发者可以验证优化是否有效，而不是盲目依赖“常识”。

1. 不同的编译器。

用符合C++11的编译器。C++11实现了右值引用和移动语义，消除了许多以前C++版本不可避免的复制操作。

2使用更好的算法

即使是在十几个项目的小数据集上，如果数据被频繁搜索，最佳搜索或排序也可以节省大量时间。第5章，优化算法包含一些关于优化算法的指导。

3使用更好的库

4减少内存分配和内存占用

由于字符串是许多C++程序中非常重要（且代价高昂）的一部分，因此我专门用了整整一章来介绍它们，作为优化的案例研究。第4章，优化字符串用途：案例研究在字符串处理的熟悉环境中介绍并激发了许多优化概念。

对缓冲区复制函数的单个调用也可能消耗数千个周期。因此，减少复制显然是加快代码速度的一种方法

5删除计算

**优化应聚焦于频繁执行的代码（热点代码），而不是过度关注单个语句的微优化**。

6使用更好的数据结构

第10章，优化数据结构探讨了C++标准库提供的数据结构的性能、行为和权衡。第9章，优化搜索和排序讨论了使用标准库算法在简单向量和C数组上实现表数据结构。

7提高并发

8优化内存管理

内存管理器是C++运行时库中管理动态内存分配的部分，在许多C++程序中经常被执行。C++有一个用于内存管理的广泛的API，尽管大多数开发人员从未使用过它