《Java并发编程实战》笔记

**Brian Goetz、Tim Peierls、Joshua Bloch、Joseph Bowbeer、David Holmes、Doug Lea著**

**童云兰 等译**

**机械工业出版社**

# 前言

当开发Java并发程序时，所要面对的挑战之一就是：平台提供的各种并发功能与开发人员在程序中需要的并发语义并不匹配。

相关书籍：

并发的入门介绍可参考《The Java Programming Language》

并发的百科全书《Concurrent Programming in Java》

# 简介

要想充分发挥多处理器系统的强大计算能力，最简单的方式就是使用线程。

## 并发简史

之所以在计算机中加入操作系统来实现多个程序的同时执行，主要是基于以下原因：

**资源利用率**。在某些情况下，程序必须等待某个外部操作执行完成，例如输入操作或输出操作等，而在等待时程序无法执行其他任何工作。

**公平性**。不同的用户和程序对于计算机上的资源有着同等的使用权。一种高效的运行方式是通过粗粒度的时间分片（Time Slicing）使这些用户和程序能共享计算机资源，而不是由一个程序从头运行到尾，然后再启动下一个程序。

**便利性**。通常来说，在计算多个任务时，应该编写多个程序，每个程序执行一个任务并在必要时相互通信，这比只编写一个程序来计算所有任务更容易实现。

但凡做事高效的人，总能在串行性和异步性之间找到合同的平衡，对于程序来说同样如此。

【批注】：了解进程和线程的区别

//TODO zhangqi

线程允许在同一个进程中同时存在多个程序控制流。

线程会共享进程范围内的资源，例如内存句柄和文件句柄，但每个线程都有各自的程序计数器（Program Counter）、栈以及局部变量等。

线程还提供了一种直观的分解模式来充分利用多处理器的硬件并行性，而在同一个程序中的多个线程也可以被同时调度到多个CPU运行。

线程也被称为轻量级进程。在大多数现代操作系统中，都是以线程为基本的调度单位，而不是进程。如果没有明确的协同机制，那么线程将彼此独立执行。由于同一个进程中的所有线程都将共享进程的内存地址空间，因此这些线程都能访问相同的变量并在同一个堆上分配对象，这就需要实现一种比在进程间共享数据粒度更细的数据共享机制。如果没有明确的同步机制来协同对共享数据的访问，那么当一个线程正在使用某个变量时，另一个线程可能同时访问这个变量，这将造成不可预测的结果。

## 线程的优势

如果使用得当，线程可以有效地降低程序的开发和维护等成本，同时提升复杂应用程序的性能。线程能够将大部分的异步工作流转换成串行工作流，因此能更好的模拟人类的工作方式和交互方式。此外，线程还可以降低代码的复杂度，使代码更容易编写、阅读和维护。

### 1.2.1 发挥多处理器的强大能力