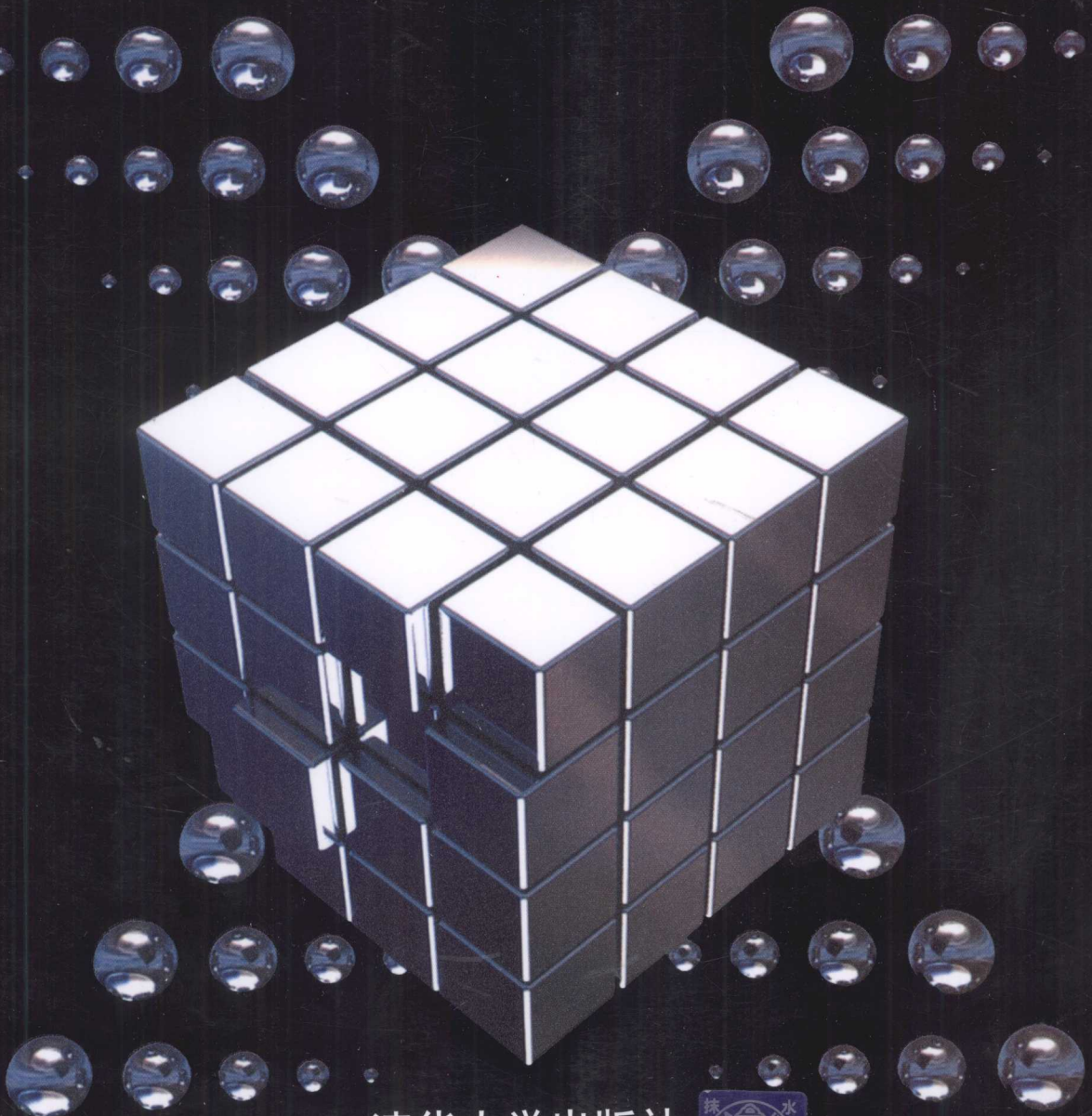
多核时代，从串行到并行，你的编程思想和开发技术必须升级！

释放多核潜能

英特尔Parallel Studio并行开发指南

英特尔亚太研发有限公司 北京并行科技公司 编著



清华大学出版社

释放多核潜能——英特尔 Parallel Studio 并行开发指南

英特尔亚太研发有限公司　北京并行科技公司　编著

清华大学出版社
北　京

内 容 简 介

本书采用工程理论、工具详解和实际案例分析相结合的方式，全面介绍了英特尔 Parallel Studio 工具集的使用。全书分三部分：基础部分（第 1、2 章）介绍了多核架构、并行编程的关键理论，Parallel Studio 的特点以及一些简单案例；中级部分（第 3～12 章）详述了 Parallel Studio 各个组件的使用，是本书的重点；提高部分（第 13 章）选取了来自英特尔线程挑战赛的 4 个算例和 1 个商业软件并行优化案例，提供了从工程实际角度解决并行问题的视角。

本书适合所有对并行开发技术感兴趣的人员，包括具备一定编程经验的程序员、调试人员，计算密集型行业的高性能计算架构师、性能优化分析师，并行开发的研究人员，对英特尔 Parallel Studio 感兴趣的技术决策者等。此外，本书也可作为高等院校计算机专业并行开发相关课程的培训及社会实践参考用书。

本书封面贴有清华大学出版社防伪标签，无标签者不得销售。
版权所有，侵权必究。侵权举报电话：010-62782989　13701121933

图书在版编目（CIP）数据

释放多核潜能：英特尔 Parallel Studio 并行开发指南 / 英特尔亚太研发有限公司，北京并行科技公司编著. —北京：清华大学出版社，2010.9
ISBN 978-7-302-23503-3

Ⅰ. ①释… Ⅱ. ①英… ②北… Ⅲ. ①并行程序－程序设计－软件工具－指南 Ⅳ. ①TP311.11-62②TP311.56-62

中国版本图书馆 CIP 数据核字（2010）第 153988 号

责任编辑：王峰松　张为民
责任校对：徐俊伟
责任印制：孟凡玉
出版发行：清华大学出版社　　**地　　址**：北京清华大学学研大厦 A 座
http://www.tup.com.cn　　**邮　　编**：100084
社　总　机：010-62770175　　**邮　购**：010-62786544
投稿与读者服务：010-62795954，jsjjc@tup.tsinghua.edu.cn
质　量　反　馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 装 者：北京嘉实印刷有限公司
经　　销：全国新华书店
开　　本：185×260　**印　张**：19　**字　数**：475 千字
版　　次：2010 年 9 月第 1 版　　**印　　次**：2010 年 9 月第1 次印刷
印　　数：1～5000
定　　价：39.80 元

产品编号：037536-01

序　一

随着 CPU 技术进入多核时代，如何有效利用多核并发运算优势，提升设备的处理能力，满足用户对高性能的需求，已成为业界关注的焦点。

在单核 CPU 时代，处理器芯片厂商提升 CPU 运算能力的主要途径是提高主频。作为 CPU 的主要性能指标，主频标志着每单位时间内 CPU 能够执行运算指令的数量。实际上，在单核 CPU 时代，处理器已经实现了多线程运算，通过在逻辑上模拟出多个 CPU 内核，以实现多任务调度和并发处理。然而，这些处理过程始终由单个 CPU 以线程切换的方式完成的，运算负载由单个 CPU 承担。而多核 CPU 则在真正意义上实现了内核级并行，与传统的单核 CPU 相比，多核 CPU 带来了更强的并行处理能力、更高的计算密度和更低的时钟频率，并大大减少了散热和功耗。

目前，越来越多用户的 PC 系统都具备了双核、四核的处理器。随着处理器数的增加，程序的并行执行度可以更高，但是目前不少用户觉得多核没有带来很明显的性能提升，这是因为现在针对多核开发和优化的应用程序还比较少。主要原因之一就是开发并行执行程序的难度非常大，程序员面临的巨大挑战就是如何把需要执行的任务并行化，而编写并行度超过 4 路以上的高效率程序。程序员没有经过系统的专业学习和长期并行编程的实践经验，编写的程序就很难充分利用多核处理器带来的并行计算优势。即使是并行编程经验丰富的程序员在编写并行度较高的程序时的效率，相比他编写并行度低的程序时的效率也要低得多。

“工欲善其事，必先利其器”。为此，英特尔公司推出了 Parallel Studio 并行程序开发套件，旨在为基于 Microsoft Visual Studio 的 C 与 C++程序开发各阶段提供简单、高效的并行开发工具，显著提高应用程序在英特尔多核处理器上的性能。英特尔还同时推出了该开发套件的新插件英特尔 Parallel Advisor Lite，它可以帮助架构师确定哪些源代码最适合并行化并提供详尽的并行化实施建议。

作为简捷的端到端并行化开发工具，英特尔 Parallel Studio 旨在帮助 Windows 开发人员提升基于英特尔多核处理器的并行软件开发流程的效率和体验，进一步优化 Windows 串行或并行应用软件，帮助软件开发商增强基于英特尔多核处理器平台的核心竞争力，进一步释放英特尔平台的效能。它具有下列功能：

- 端到端的并行化产品套件：易于实施，且覆盖软件开发的整个周期，即设计、编码、调试、测试。
- 前向扩展：可以有效地运行于未来任何英特尔处理器平台。基于标准的线程解决方案，显著提升企业在 C/C++应用软件和 C/C++开发人员方面的投资效益。
- 轻松实现并行化：快速提升生产效率，提高硬件和软件的投资回报率，同时使学习曲线达到最优化。
- 多种并行程序开发方案：针对数据和任务并行化编程提供相应的开发方案。

为了帮助在桌面 PC、笔记本计算机和 Windows 工作站上用 Visual Studio 开发 C/C++

程序的广大开发人员早日进入多核并行编程的行列，适应多核平台时代的必然要求，英特尔软件与服务事业部组织了长期从事高性能计算的并行应用优化、开发的工程师和北京并行科技有限公司的技术专家，共同编写了这本“工程师写，工程师读”风格的《释放多核潜能——英特尔 Parallel Studio 并行开发指南》。相信这本取材于英特尔高性能支持团队大量真实并行优化案例的厚积薄发之作，在并行计算的推广和人才培养方面将发挥重要的作用。

梁兆柱 博士
英特尔亚太研发有限公司 总经理
英特尔公司软件与服务事业部 中国区总经理

此为试读，需要完整PDF请访问：www.ertongbook.com

序　二

这是一本对程序员非常有帮助的书。我们都需要认识和使用并行计算，而本书是有助于充分了解并行编程的最好工具。

目前多核处理器无处不在，并行编程可以充分发挥现有多核的计算能力，而这一技术的推广依赖两个方面：程序员的专业知识和并行编程工具。

本书通过对英特尔 Parallel Studio 的介绍，能帮助所有程序员获得并行编程的专业技能。

英特尔 Parallel Studio 是一套有利于并行编程的、无与伦比的工具集，展现了其在这一领域的显著进展。在实现高可扩展、可靠、可持续性的并行应用中，这些工具能有效地解决关键性技术难题。

这足以证明英特尔为什么能在并行编程工具领域领跑世界。英特尔引领了处理器的发展，从超线程技术到多核处理器，这些技术使得并行计算得到普遍应用，引发了计算机体系结构的变革，而现在是软件面临着变革。

为了帮助软件开发人员，英特尔推出了非常受欢迎的 Intel Threading Building Blocks（TBB，英特尔线程构建模块，参见 http://threadingbuildingblocks.org），它在 C++并行编程领域的普及地位已不可替代。TBB 被移植到多种处理器和操作系统上，有开源版和商业版两个版本，是当前最为广泛使用的 C++编程模型。此外，英特尔还致力于创建教授和老师们的社区（参见 http://intel.com/thinkparallel），以及开发者社区（参见 http://intel.com/software）。

英特尔 Parallel Studio 展现了在并行编程工具方面的重大进展以及英特尔的引领地位。它囊括了并行编程中的许多重要解决方案。正是这些完美解决方案真正使 Parallel Studio 与众不同，它涉及设计、实现、纠错和性能调优的各个环节。这是其他产品所不可比拟的。

我们都需要“并行思维”，没有比这本书和英特尔 Parallel Studio 工具集更好的起点了。

在此，我非常赞赏本书所有作者在完成本书过程中对“并行思维”模式所做出的贡献。

我们衷心希望本书对您有所帮助并期待您的反馈！

仁达敬（James Reinders）*

---

* 仁达敬是英特尔软件开发产品部的负责人，编写了很多并行编程方面的文章和书籍，其中包括近期 O'Reilly 出版社出版的《Intel Threading Building Blocks》一书。

前　言

本书开始策划的时候，哥本哈根世界气候大会还没有召开。写作组怀着一颗纯粹的让多核潜能完全释放的心，也就是最初的“绿色计算”的想法，召集了并行开发界的工程师、专家，准备编写一本介绍并行开发的书。伴随着哥本哈根会议的召开，全球开始轰轰烈烈地提倡节能、低碳，《释放多核潜能——英特尔 Parallel Studio 并行开发指南》一书便应运而生了。

1. 本书缘起

摩尔定律的发展，把计算机业界带入前所未有的多核时代。过去只在高端服务器和高性能计算领域关注的并行编程，业已成为桌面 PC 开发者必须逾越的门槛。而不断发展的硬件复杂性和处理器微架构的创新，使得并行软件开发模式和相应的软件开发工具，包括建构在其上的开发者的思维方式，都面临新的挑战。

我们知道，面向高性能服务器集群和企业级服务器，英特尔公司很早就提供了一整套并行软件开发工具，包括 C/C++/FORTRAN 编译器、Vtune 性能分析器、英特尔数学核心函数库 MKL、英特尔集成性能原件 IPP、英特尔 Thread Profiler、英特尔 Trace Analyzer/Collector 等，这一系列的开发工具，旨在实现从微架构并行、线程并行、进程并行到数据级并行的最大并行性能，而开发者也需要具备一定的并行编程和 Linux 下程序开发的经验。

随着多核平台的普及，桌面 PC 甚至笔记本计算机都具备了过去小型服务器的计算能力，为此，英特尔公司结合超过 25 年的并行软件开发和高性能计算的专门技术，将基于 Linux 高性能服务器机集群上的并行开发下移到基于 Windows 的多核桌面 PC，为广大使用 Microsoft Visual Studio 的应用开发人员提供了端到端并行开发工具 Parallel Studio，为多核时代在 Windows 上使用 C/C++开发具备多核扩展能力的串行或并行多线程应用提供了高效的开发工具。

使用 Microsoft Visual Studio 的广大开发人员将在熟悉的开发模式下通过 Parallel Studio 解决下列新的挑战：

- 代码的哪些部分最适合并行化以获得最佳性能并避免内存使用冲突？
- 哪类并行编程模式和特定的多线程实现适合于您的应用？
- 由于多线程程序在运行时的线程执行顺序是非确定的，如何检测和改正难以复现的程序线程错误和内存错误？
- 如何在多核平台上提升多线程应用的性能并使其性能随核数扩展？

为帮助在 Windows 平台上编程的开发人员早日进入并行编程的行列，特别是理解在并行开发工具后面的方法论、并行编程模式和性能优化的方法，我们编写了这本 Intel Parallel Studio 应用指南。

2. 读者对象

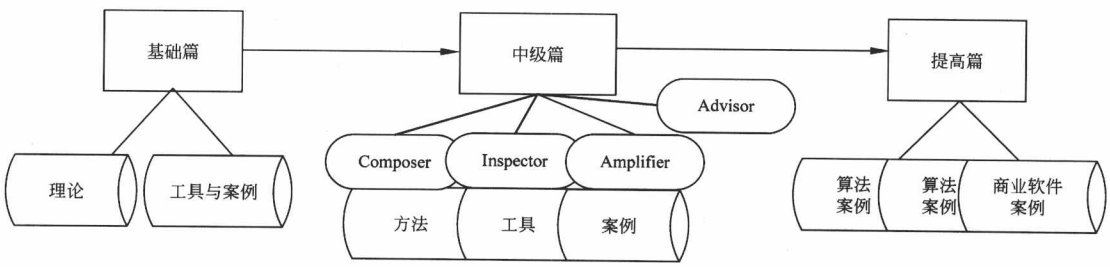
只要是基于 Windows 的开发人员，都适合使用英特尔 Parallel Studio 全面提高程序的性能。他们包括但不局限于以下几类：多线程编程人员；高性能计算编程人员；性能优化分析师；软件调试工程师；垂直领域科研开发工作者，如数值算法研究，编解码研究；互联网计算密集型应用开发者；还有以上领域的技术决策者；等等。此外，科研院校相关专业的学生、老师、培训师也可以将其作为参考资料进行学习。

读者如果要学习具体的并行开发技术，还需要具备一定的 C/C++编程能力，熟悉 Microsoft Visual Studio 的使用，同时最好了解一些并行编程的基本知识，如 OpenMP、WinThread 编程等。

3. 内容组织

从事本书写作的是英特尔软件与服务事业部和北京并行科技有限公司相关技术专家组成的并行专题写作组。英特尔软件与服务事业部高性能计算团队自 2004 年起就专注于国内高性能计算应用在 Intel 平台上的并行开发和性能优化，积累了丰富的工程实际经验，体现在本书的行文风格和大量抽象自实际案例的代码上。本书不仅可作为学习 Intel Parallel Studio 的指南，更可以为读者带来开发和优化并行程序的方法，流程和实际工程案例参考，为从事并行应用开发的读者提供了从工程实际角度解决并行问题的视角。

本书内容采取工程理论、工具介绍和实际案例分析三位一体的叙述结构，分为基础部分、中级部分和提高部分三块，如下所示。



在基础部分（第 1、2 章），介绍并行编程和技术的基本概念，多核架构和多核编程的关键理论，Parallel Studio 产品的定位和特点，并用快速上手的案例突出工具的使用模式。在中级部分（从第 3～12 章）分别详述 Parallel Studio 的各个组件的使用，其理论基石和相应的案例分析，这是本书的重点。在提高部分（第 13 章）选取了来自英特尔线程挑战赛的 4 个算例，综合使用 Parallel Studio 全部组件，贯穿并行算法的挑选、数据结构的构建，并行纠错和并行性能优化整个开发周期，从问题描述、串行实现到并行优化的角度加以分析。最后，采用 Parallel Studio 对并行科技基于 QT 构建的商业软件 Paraview 进行了并行优化，突出方法和工具在完整工程优化项目中的使用。

本书具体章节概括如下：

第 1 章介绍并行编程和技术的基本概念，多核架构和多核编程的关键理论。

第 2 章介绍 Parallel Studio 产品的定位和特点，以及快速上手的案例。

第 3 章介绍英特尔 Parallel Composer 组件中的英特尔 C/C++编译器、英特尔并行调试

器、英特尔线程构建模块 TBB、英特尔集成性能基元 IPP 的使用。

第 4 章介绍几种不同的并行化方法的理论概念和实际并行化开发中的一些问题，并通过蒙特卡洛法计算 π 值的并行优化案例加以说明。

第 5 章通过 4 个典型案例对 Parallel Composer 的各个组件在实际开发中作深入介绍，分别是浮点计算的向量化和自动并行，并行调试器的使用，使用英特尔 TBB 进行字符串并行查找和调用英特尔 IPP 进行压缩解压缩。

第 6 章介绍 Intel Parallel Studio 软件包中 Inspector 组件的基本功能和用法，体现了 Inspector 的简单、方便、容易上手的特点。

第 7 章介绍软件纠错的一些基本概念和理论，特别说明了并行软件纠错的特点。

第 8 章通过一些简单明了的案例来介绍使用 Inspector 模块来如何发现并行软件开发中碰到的线程间死锁、竞争或对内存的错误操作等问题，并说明在基于线程或 OpenMP 的并行软件开发中，如何避免或解决这些问题。

第 9 章关于 Parallel Amplifier 的详细使用方法和介绍，相当于参考手册。

第 10 章介绍性能优化方法论，包括通用优化方法论以及并行优化方法论。

第 11 章使用 Amplifier 的优化案例分析，从各个层面介绍优化技术点。

第 12 章介绍 Parallel Advisor 辅助用户定位串行程序中可并行的代码段，并分析选定代码的加速效果以及潜在的数据共享错误。

第 13 章介绍英特尔线程挑战赛的 4 个算例以及商业软件 Paraview 的完整工程过程。

4．阅读建议

（1）按章节顺序从前至后顺次阅读。

并行开发不仅是一个工具的使用，更是一种编程思维。本书在编写时将方法论和工具融合起来，不同的工具对应到不同的方法论，再用案例进行诠释，由浅入深，由易到难，环环相扣。所以入门读者最好按照章节顺序从前至后顺次阅读，不要忽略任何一个章节。

（2）理解并实验每一段案例代码。

有编程经验的读者都有过看别人的程序时总觉得心领神会，轮到自己写的时候就脑子空空的经历。本书编写了大量案例，在数量有限的讲解并行开发的书籍中属凤毛麟角，建议读者能够用心体会并动手在机器上实验这些代码。特别是最后一章，给出了一些典型算法的并行案例以及真实商业软件的并行过程，相信对读者大有裨益。

（3）单独抽看方法论部分。

已经有过一定并行开发经验的读者可以先跳过英特尔 Parallel Studio 套件的讲解，抽看方法论部分。本书的并行编程方法论也能自成体系，从代码并行化，并行查错，再到并行优化，相信对读者在并行思维的层面上有很大提高。

（4）单独抽看英特尔 Parallel Studio 各组件的详解部分。

想快速了解英特尔 Parallel Studio 如何使用的读者，可以单独抽看个组件的详解部分。这部分也可以作为英特尔 Parallel Studio 套件使用的技术白皮书。

（5）反复阅读案例部分或着手优化自己的程序。

通读本书后，读者不妨经常回过头来分析一下案例部分，或者着手将自己的程序进行优化。程序的优化和性能提高是永无止境的，本书的案例中也还存在有待提高的地方，希

望读者能够在这个过程中始终贯彻并行思维。

5. 致谢

本书的第 2、3、4、5、10、11、12 章由英特尔高性能计算支持团队编写，其中第 2、10、11 章由乔楠编写，第 3、4、5 章由金君编写，第 12 章由孙相征编写；第 1、9、13 章由北京并行科技有限公司编写，其中第 1、9 章由颜伟编写，第 13 章由邓辉、徐海涛编写；第 6、7、8 章由英特尔高性能计算支持团队的王哲和北京并行科技有限公司的颜伟共同编写；全书由北京并行科技有限公司的贺玲统稿。在此对他们及支持他们在家里编写本书的家人们表示感谢！

苏轼在《稼说》一文中提出学习的主张“博观而约取，厚积而薄发”是我们多年并行程序优化工作的共鸣，其精髓就在勤于积累和精于应用。从一个个并行性能优化的项目中，通过不断地遭遇问题、解决问题和总结问题的过程，我们积累和抽象出相应的方法和工具使用技巧，希望能通过本书用工程语言恰当地描述面对的问题、方法、工具和解决方案，藉由实践案例来阐述 Parallel Studio 各功能模块的运用。如果各位读者能够从中不仅学会“Know-how”，而且能够进一步学习和探寻工具背后的“Know-why”，则我们的目的就达到了。

何万青　博士
英特尔软件与服务事业部高性能计算工程经理

目　录

第 1 章　并行开发理论基础 ……1

1.1　并行相关概念 ……1
1.1.1　并发与并行、并行度 ……1
1.1.2　粒度 ……1
1.1.3　加速比及其定律 ……2
1.1.4　可扩展性与并行效率 ……4
1.1.5　负载均衡 ……4
1.1.6　吞吐量与延迟 ……4
1.1.7　热点与瓶颈 ……4
1.2　多核并行 ……4
1.2.1　多核软硬件现实 ……5
1.2.2　多核架构 ……5
1.2.3　多核并行手段 ……6
1.2.4　多核并行设计方法 ……7
1.2.5　多核多线程系统 ……8
1.2.6　多核多线程同步 ……9
1.2.7　多核多线程实现的问题 ……11
1.3　小结 ……11

第 2 章　英特尔 Parallel Studio 基础 ……12

2.1　英特尔 Parallel Studio 介绍 ……12
2.1.1　英特尔 Parallel Studio 背景 ……12
2.1.2　英特尔 Parallel Studio 的组成 ……12
2.1.3　英特尔 Parallel Studio 的特色 ……13
2.1.4　英特尔 Parallel Studio 的使用者 ……14
2.2　英特尔 Parallel Studio 快速上手 ……14
2.2.1　英特尔 Parallel Studio 的下载安装 ……14
2.2.2　选择案例 ……14
2.2.3　实践动手第一步：采用 Parallel Studio 运行串行程序 ……15
2.2.4　实践动手第二步：选用合适的实现对代码并行化 ……17
2.2.5　实践动手第三步：定位错误 ……18
2.2.6　实践动手第四步：性能优化 ……20
2.3　小结 ……23

此为试读，需要完整PDF请访问：www.ertongbook.com

第 3 章 英特尔 Parallel Composer 详解……24

3.1 Composer 概述……24
3.2 英特尔 C/C++编译器……25
3.2.1 自动并行和 OpenMP 并行……25
3.2.2 过程间优化……29
3.2.3 档案导引优化……29
3.2.4 编译器向量化……30
3.3 英特尔并行调试器……32
3.3.1 英特尔并行调试器概述……32
3.3.2 线程数据共享侦测……32
3.3.3 可重入函数调用侦测……34
3.3.4 SSE 寄存器窗口……34
3.3.5 OpenMP 多线程调试……35
3.3.6 并行区域的串行执行……36
3.4 英特尔 TBB 线程构建模块……36
3.4.1 英特尔 TBB 概述……37
3.4.2 功能模块分类与介绍……37
3.4.3 编译和运行 TBB 多线程程序……37
3.5 英特尔 IPP 性能基元……38
3.5.1 英特尔 IPP 概述……38
3.5.2 主要函数及其功能……39
3.5.3 编译和运行……45
3.6 小结……45

第 4 章 并行化方法……46

4.1 基本概念……46
4.1.1 Amdahl 定律……46
4.1.2 进程与线程……47
4.2 并行化方法……48
4.3 并行化设计……49
4.3.1 任务划分……49
4.3.2 功能划分……51
4.3.3 并行化开发中的一些思考……51
4.4 案例分析：用蒙特卡罗方法计算 π 值……55
4.5 小结……58

第 5 章 英特尔 Parallel Composer 案例分析……59

5.1 案例 5-1：Composer 的使用——向量化和自动并行化……59

5.2 案例 5-2：并行调试器的使用 ……62
5.3 案例 5-3：通过 TBB 进行字符串查找 ……66
5.4 案例 5-4：IPP 压缩和解压缩案例介绍 ……69
5.5 小结 ……71

第 6 章 英特尔 Parallel Inspector 详解 ……72

6.1 Inspector 概述 ……72
6.2 启动 Inspectort ……73
6.2.1 工作流程 ……73
6.2.2 启动 ……73
6.3 配置查找错误的类型和粒度 ……75
6.3.1 基于线程的相关错误及粒度 ……76
6.3.2 基于内存的相关错误及粒度 ……77
6.4 定位和解决发现的错误 ……79
6.4.1 检查错误 ……79
6.4.2 查看和分析错误 ……80
6.5 小结 ……81

第 7 章 软件纠错方法 ……82

7.1 基本概念 ……82
7.1.1 软件查错或纠错 ……82
7.1.2 白盒测试 ……82
7.1.3 黑盒测试 ……83
7.2 并行软件的纠错 ……84
7.3 线程并行的常见错误 ……84
7.3.1 线程间死锁 ……85
7.3.2 线程间竞争 ……86
7.3.3 内存泄露 ……87
7.4 小结 ……89

第 8 章 并行软件纠错案例 ……90

8.1 案例 8-1：线程间相互作用导致的死锁问题 ……90
8.2 案例 8-2：线程竞争 ……94
8.3 案例 8-3：内存泄露 ……99
8.4 小结 ……101

第 9 章 英特尔 Parallel Amplifier 详解 ……102

9.1 Amplifier 概述 ……102
9.1.1 如何开始 Amplifier ……102

9.1.2 如何使用符号信息 ……103
9.1.3 环境和对象 ……106
9.2 Amplifier 的几个概念 ……107
9.3 Amplifier 的分析运行 ……109
9.3.1 分析运行的几个选项 ……109
9.3.2 选择分析模式 ……110
9.3.3 如何选择分析模式 ……110
9.3.4 如何在命令行下运行分析模式 ……110
9.3.5 热点：分析程序哪里耗时 ……111
9.3.6 并行度：展现并行程序的另外一个特点 ……111
9.3.7 锁和等待：分析程序在哪里等待 ……112
9.3.8 选择数据采集的时段 ……112
9.4 Amplifier 中浏览性能数据结果 ……112
9.4.1 总览 ……112
9.4.2 在 Bottom-up 和 Top-down 中切换 ……113
9.4.3 选择和管理栈类型 ……114
9.4.4 选择颜色方案 ……114
9.4.5 按照不同类型划分组 ……114
9.4.6 在命令行模式下查看性能数据 ……114
9.5 Amplifier 解释性能数据结果 ……115
9.5.1 总览 ……115
9.5.2 解释热点分析结果 ……115
9.5.3 解释并行度分析结果 ……116
9.5.4 解释锁和等待分析结果 ……116
9.6 Amplifier 中的源代码 ……117
9.7 Amplifier 中对比性能数据结果 ……117
9.8 Amplifier 中管理结果文件 ……119
9.9 小结 ……120

第 10 章 性能优化方法 ……121

10.1 性能优化概述 ……121
10.1.1 性能和性能优化是计算机领域不变的主题 ……121
10.1.2 性能优化的定义 ……121
10.2 性能优化通用方法 ……122
10.2.1 性能优化的顺序 ……122
10.2.2 系统级别的性能优化 ……122
10.2.3 应用级别的性能优化 ……123
10.2.4 微架构级别的性能优化 ……123
10.2.5 性能优化工作循环 ……123

10.2.6 性能优化循环的常见问题……123
10.3 并行应用性能优化方法……125
10.3.1 概述……125
10.3.2 减少关键路径上的时间……126
10.3.3 检查是否选择最优的并行方法……126
10.3.4 检查是否选择合适的层级开始并行……127
10.3.5 Amdahl 定律的检查：减少串行部分的比例……127
10.3.6 检查程序的负载均衡问题……127
10.3.7 检查程序的粒度问题……128
10.3.8 采用合适的线程库……128
10.3.9 检查同步性能问题……129
10.3.10 检查硬件导致的扩展性问题……130
10.4 小结……130

第 11 章 性能优化案例……131

11.1 IO 并行：系统级优化案例……131
11.2 锁的实现：锁优化案例……133
11.3 同步与负载均衡：生产消费类型的优化案例……134
11.4 优化临界区：WinThread 循环计算型优化案例……144
11.5 负载均衡与归约：OpenMP 循环计算型优化案例……148
11.6 线程数，桶数与锁：Hash 表与 TBB 优化案例……153
11.7 选择合适的层级并行：任务与数据并行优化案例……164
11.8 避免硬件性能瓶颈：内存与高速缓存优化案例……165
11.9 算法选择：排序优化与 TBB 案例……168
11.10 内存操作 TBB 优化案例……172
11.11 小结……174

第 12 章 英特尔 Parallel Advisor 详解……175

12.1 Advisor 基础……175
12.1.1 Advisor 总览……175
12.1.2 如何开始 Advisor……175
12.2 Advisor 工作流程……177
12.3 Annotations……178
12.4 Advisor 工具……179
12.4.1 Survey……179
12.4.2 Suitability……180
12.4.3 Correctness……181
12.5 使用案例……183
12.5.1 SpMV 并行化……183

12.5.2 DGEMM 并行化 ……188
12.6 小结 ……194

第 13 章 总体系统化案例 ……195

13.1 数独 ……195
13.1.1 串行算法 ……196
13.1.2 并行优化 ……211
13.1.3 小结 ……212
13.2 最短路径 ……213
13.2.1 串行算法 ……213
13.2.2 并行优化 ……232
13.2.3 小结 ……235
13.3 基数排序 ……236
13.3.1 串行算法 ……237
13.3.2 并行优化 ……245
13.3.3 小结 ……247
13.4 骑士巡游 ……248
13.4.1 串行算法 ……249
13.4.2 并行优化 ……265
13.4.3 小结 ……271
13.5 商业软件 Paraview ……271
13.5.1 问题描述 ……272
13.5.2 并行优化 ……272
13.5.3 小结 ……284

附录 A 英文术语表 ……285

第 1 章　并行开发理论基础

为了更好地发挥英特尔 Parallel Studio 并行开发工具的功能，在介绍这一套件的使用前还需要有一定的理论基础，本章介绍最基本的理论和技术。

1.1　并行相关概念

使用英特尔 Parallel Studio 的核心是发挥应用的并行性。因此，并行相关概念显得尤为重要。

1.1.1　并发与并行、并行度

并行计算中常见的两个名词：并行与并发。二者有微妙的不同。

并发是指从广义上讲只要同时多任务处理，就是并发；但是并发并不能保证在每一个时刻都有多个任务（线程，进程）同时工作。例如，一个单核处理器平台并且这个平台没有或者没有打开硬件超线程或者多线程功能，在上面运行一个多线程程序，虽然这个程序是并行程序，但是由于处理器资源有限，在任意一个时刻最多只有一个线程是工作的。

并行就不同。并行是指在同一时刻，有多个任务同时运行，才称之为并行。从性能上讲，它体现了并行的计算能力。例如，在一个双核处理器的平台上运行多线程程序，如果并行程序设计得好，在某些时刻，同时运行的线程数应该可以达到二，从而充分利用了处理器多核资源。因此，并行是指无论从微观还是宏观上看，都是多个线程或者进程同时执行，而并发在微观上不是同时执行的，只是把时间分成若干段，使多个进程或线程快速交替地执行。

接下来还有并行度这个概念：在某个时刻同时工作的任务（线程，进程）的数量，称之为并行度。并行度越高，说明同时运行的任务数越多，如果相应的核数合适，可以提高整体的计算能力，充分发挥并行程序的效率。当然有时候也叫并发度，但是基本上都是在微观上表示同时进行的线程/进程的个数。

1.1.2　粒度

粒度是衡量计算与通信大小的度量。具体分为应用的粒度与机器的粒度。一般我们讲的粒度都是针对应用而言。

应用的粒度定义为：应用的计算量与通信量的比。通信量指的是线程之间或者进程之间进行数据共享或者消息传递的开销。

机器的粒度定义为：这个机器的计算能力与通信能力的比。

此为试读，需要完整PDF请访问：www.ertongbook.com