HZ BOOKS
华章IT

国内顶尖的并行计算领域知名专家风辰多年实践经验总结，兼具深度和高度

基于大量示例，揭秘如何在主流硬件平台上通过向量化和并行化技术优化代码性能，涵盖线性代数、偏微分方程求解、分子动力学和机器学习领域常见算法优化案例

高性能计算技术丛书

Parallel Optimization of Scientific Computing and Enterprise Applications

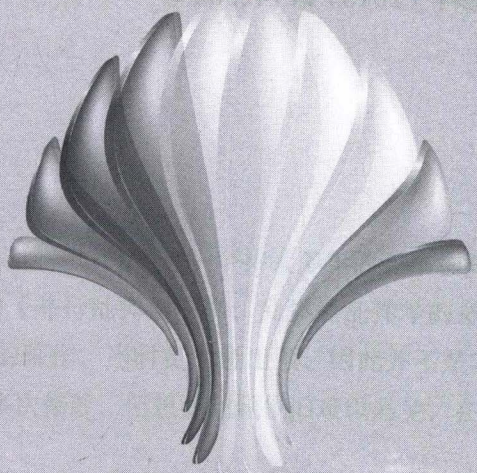
科学计算与企业级应用的并行优化

刘文志◎著

机械工业出版社
China Machine Press

科学计算与企业级应用并行优化

高性能计算技术丛书



Parallel Optimization of Scientific Computing and Enterprise Application

科学计算与企业级应用的并行优化

刘文志◎著

机械工业出版社
China Machine Press

图书在版编目（CIP）数据

科学计算与企业级应用的并行优化 / 刘文志著 . —北京：机械工业出版社，2015.7
（高性能计算技术丛书）

ISBN 978-7-111-50628-7

I. 科… II. 刘… III. ①科学计算 – 研究 ②并行算法 – 算法设计 IV. ① N32 ② TP301.6

中国版本图书馆 CIP 数据核字（2015）第 142902 号

科学计算与企业级应用的并行优化

出版发行：机械工业出版社（北京市西城区百万庄大街 22 号 邮政编码：100037）
责任编辑：高婧雅 责任校对：董纪丽
印 刷：北京市荣盛彩色印刷有限公司 版 次：2015 年 7 月第 1 版第 1 次印刷
开 本：186mm × 240mm 1/16 印 张：12.75
书 号：ISBN 978-7-111-50628-7 定 价：49.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
客服热线：（010）88379426 88361066 投稿热线：（010）88379604
购书热线：（010）68326294 88379649 68995259 读者信箱：hzit@hzbook.com

版权所有 • 侵权必究
封底无防伪标均为盗版
本书法律顾问：北京大成律师事务所 韩光 / 邹晓东

Forword 序

到这里，终于可以松一口气了，一个持续多年的工作总算可以告一段落了。本系列起源于我 2012 年想写的《并行乱弹》一书，乱弹是乱弹琴的意思。按我的本意，并不想把它写成一本非常严谨的著作，当时更无意出版，因此并不是非常注意全书逻辑的严密性，虽然经过我和编辑的多次修改，但想必问题依旧难以避免，在此诚恳地请求读者谅解。

本系列的 3 本书相互之间有联系，也有其独立性：《并行算法设计与性能优化》介绍常见的串行代码优化方法和并行算法的设计；《并行编程方法与优化实践》介绍常见的向量化和并行编程环境及一些实例；《科学计算与企业级应用的并行优化》则介绍领域相关的算法与应用的性能优化。

如果说要写一本简短的武侠小说来描写主角是如何学习这 3 本“秘笈”的话，我想故事是这样的：在 2015 年，主角是某位内向宅男码农 hpc，因为程序速度太慢天天被产品经理骂，受项目经理白眼，每天工作到晚上 10 点，遭到家人埋怨。在某个月黑风高的晚上，某条街道上那位江湖人称“风辰”的“HPC 帮”护法长老收 hpc 为不记名弟子，并传授给 hpc《并行算法设计与性能优化》《并行编程方法与优化实践》及《科学计算与企业级应用的并行优化》3 本“秘笈”，“风辰”临走时收了传功费用 1024 万元。hpc 如获至宝，休假一月潜心修炼。一月后出关，容光焕发，程序速度大幅度提升，产品经理天天请吃大餐，项目经理忙着加项目奖金。此后，hpc 每天下午 5 点即下班回家，修身养性，陪家人一起吃饭、购物。

愿这 3 本书能够真正成为改变读者生活的良师益友！

风辰

2015 年 5 月 17 日于深圳

试读结束：需要全本请在线购买： www.ertongbook.com

前　言 *Preface*

IT 行业急需这本书

和本系列的前两本书一样，在解释为什么笔者认为软件工程师需要这本书之前，笔者先来介绍并行、并发和代码性能优化这 3 个概念，因为理解这 3 个概念是阅读本系列 3 本书的基础。

- 并行对应的英文单词是 parallelism，是指在具有多个处理单元的系统上，通过将计算或数据划分为多个部分，将各个部分分配到不同的处理单元上，各处理单元相互协作，同时运行，以达到加快求解速度或者提高求解问题规模的目的。
- 并发对应的英文单词是 concurrency，是指在一个处理单元上运行多个应用，各应用分时占用处理单元，是一种微观上串行、宏观上并行的模式，有时也称其为时间上串行、空间上并行。
- 代码性能优化是指通过调整源代码，使得其生成的机器指令能够更高效地执行，通常高效是指执行时间更少、使用的存储器更少或能够计算更大规模的问题。

从大的方面来说，并行和并发都是代码性能优化的一种方式，但是今天并行和并发已经是如此重要，以至于需要“开宗立派”。为了明晰并行、并发和代码性能优化的边界，在本书中，代码性能优化特指除了并行和并发以外的代码优化方法，比如向量化和提高指令流水线效率。在本书中，笔者将向量化独立出来解说。

2003 年以前，在摩尔定律的作用下，单核标量处理器的性能持续提升，软件开发人员只需要写好软件，而性能的提升就等待下次硬件更新来解决，在 2003 年之前的几十年

里，这种“免费午餐”模式一直在持续。2003 年后，主要由于功耗的原因，这种“免费午餐”已经不复存在。为了生存，各硬件生产商不得不采用各种方式提高硬件的计算能力。目前最流行的 3 种方式如下：

1）让处理器在一个周期处理多条指令，多条指令可相同可不同。如 Intel Haswell 处理器一个周期可执行 4 条整数加法指令、两条浮点乘加指令，而访存和运算指令也可同时执行。

2）使用向量指令，主要是 SIMD 和 VLIW 技术。SIMD 技术将处理器一次能够处理的数据位数从字长扩大到 128 或 256 位，也就提升了计算能力。

3）在同一个芯片中集成多个处理单元，根据集成方式的不同，相应地称为多核或多路处理器。多核处理器是如此重要，以至于现在即使是手机上的嵌入式 ARM 处理器都已经是四核或八核了。

目前绝大部分应用软件都是串行的，因为串行执行过程符合人类的思维习惯，易于理解、分析和验证。由于串行软件只能在多核 CPU 中的一个核上运行，和 2003 年以前的 CPU 没有多少区别，这意味着花多核 CPU 的价钱买到了单核的性能。通过多核技术，硬件生产商成功地将提高实际计算能力的任务转嫁给软件开发人员，而软件开发人员没有选择，只有直面挑战。

标量单核的计算能力没有办法继续大幅度提升，而应用对硬件计算能力的需求依旧在提升，这是个实实在在的矛盾。在可见的将来，要解决这个矛盾，软件开发人员只有代码性能优化和并行可以选择。代码性能优化并不能利用多核 CPU 的全部计算能力，它也不要求软件开发人员掌握并行开发技术，另外通常也无需对软件架构做改动，而且串行代码优化有时能够获得非常好的性能（如果原来的代码写得很差的话），因此相比采用并行技术，应当优先选择串行代码性能优化。一般来说，采用并行技术获得的性能加速不超过核数，这是一个非常大的限制，因为目前 CPU 硬件生产商最多只能集成十几、几十个核。

从 2006 年开始，可编程的 GPU 越来越得到大众的认可。GPU 是图形处理单元（Graphics Processing Unit）的简称，最初主要用于图形渲染。自 20 世纪 90 年代开始，NVIDIA、AMD（ATI）等 GPU 生产商对硬件和软件加以改进，GPU 的可编程能力不断

提高，GPGPU（General-purpose computing on graphics processing units）比以前容易许多。另外由于GPU具有比CPU强大的峰值计算能力，近年来引起了许多科研人员和企业的兴趣。

近两三年来，在互联网企业中，GPU和并行计算越来越受到重视。无论是国外的Google、Facebook，还是国内的百度、腾讯、阿里和360，都在使用代码性能优化、并行计算和GPU来完成以前不能完成的任务。

10年前，并行计算还是实验室里教授们的研究对象，而今天多核处理器和GPU的普及已经使得普通人就可以研究它们。对于软件开发人员来说，如果不掌握并行计算和代码性能优化技术，在不久的将来就会被淘汰。

作为本系列的压轴之作，本书专注于领域相关的算法和应用的并行与性能优化。笔者介绍了如何优化线性代数、偏微分方程求解、分子动力学和机器学习领域的一些重要算法。

如果以武侠中的功夫来比喻的话，本系列的第一本书《并行算法设计与性能优化》专注于理论基础和实践的结合，是“内功”和“心法”的修炼；本系列的第二本书《并行编程方法与优化实践》专注于程序设计语言的核心内容的应用，是“招式”的学习；而本书则是“内功心法”和“招式”的具体运用。“内功心法”好意味着基础好、潜力大，以后发展空间广泛；“招式”好则能够通过“奇招”“怪招”解决问题；而只有将“内功心法”和“招式”完美融合，做到“心中无招而手中有招，无招胜有招，无招即有招”，才能成为异构并行计算这个领域真正的集大成者。愿读者和笔者一起向这个目标前进。

本书完全是“干货”，是一本真正将业界最佳实践和“只可意会，不可言传”的领域知识简洁明了地贡献出来的著作。为了帮助读者理解，本书使用了大量的示例。开发人员通常比较忙，因此本书力求简洁明了，点到为止。

读者对象

由于多核处理器和GPU已经非常便宜，而代码性能优化、向量化和并行已经深入IT行业的骨髓，所有IT行业的从业者都应当阅读本书。如果非要列一个读者清单，笔者

认为下列人员应当阅读本书：

- 互联网及传统行业的 IT 从业者，尤其是希望将应用移植到多核向量处理器的软件开发人员；
- 对向量化和并行化感兴趣的职业工作者；
- 线性代数、偏微分方程、分子动力学和机器学习相关领域的科技工作者；
- 大中专院校及研究所的学生、教师；
- 关注异构并行计算和高性能计算的人们。

如何阅读本书

本系列包括 3 本书，本书是此系列的第三本。本书重点介绍如何利用目前主流的 C 语言的各种特定硬件或平台的向量化扩展、并行化库，来设计性能优良的向量化和并行代码。而本系列的第一本《并行算法设计与代码优化》关注并行优化和并行计算相关的理论、算法设计及高层次的实践经验；本系列第二本《并行编程方法与优化实践》关注 C 程序设计语言的向量化和并行化扩展及算法到硬件的映射；本书则关注如何将线性代数、偏微分方程求解、分子动力学和机器学习领域的常见算法优良地实现出来。

本书不但包括如何使用 SSE/AVX 向量化扩展、OpenMP 编译制导语句来优化运行在 X86 多核处理器上的代码性能，还包括使用 NEON 向量化扩展、OpenMP 编译制导语句优化运行在移动处理器（ARM）的代码性能，以及使用 CUDA 和 OpenCL 优化运行在图形处理器（GPU）的代码性能。笔者希望通过这种方式能够让阅读本书的软件开发人员了解和掌握如何将常见算法映射到具体硬件上以获得高性能，以及如何依据硬件和算法的特点进行代码性能优化。

本书分为以下几章：

第 1 章 介绍常见的并行编程基于的多核 / 众核向量处理器的架构、OpenCL 程序如何映射到这些平台上执行及 OpenCL 程序在这些硬件上运行时具有哪些不同。先介绍 Intel Haswell、ARM A15、Intel MIC、AMD GCN GPU 和 NVIDIA Kepler/Maxwell GPU 的架构。然后介绍 OpenCL 程序如何映射到 Intel Haswell 处理器、AMD GCN GPU 和 NVIDIA Kepler/Maxwell GPU 上执行。最后介绍 OpenCL 程序在这些处理器上运行时的

细微区别。

第 2 章 介绍如何在 X86、ARM 和 GPU 上优化常见的线性代数运算，如计算稀疏矩阵向量乘法，求解下三角线性方程组，计算矩阵乘法等。对于电子电路模拟、计算流体力学相关的领域来说，稀疏矩阵向量乘法是非常重要的。本章还介绍如何在主流的 X86、ARM 和 NVIDIA GPU 上优化稀疏矩阵向量乘法运算。

第 3 章 介绍如何在 X86 和 GPU 处理器上优化偏微分方程的求解，主要介绍如何求解热传递问题和三维 Stencil 问题。

第 4 章 介绍如何在 X86 处理器和 GPU 上优化常见的分子动力学算法，如邻居搜索、范德华力计算、键长伸缩力计算和径向分布函数计算。

第 5 章 介绍如何在 X86、ARM 和 GPU 上优化常见的机器学习算法，如 k-means、KNN、二维卷积和四维卷积的计算性能。最后介绍多 GPU 并行卷积神经网络的常见方法，以及如何使用数据并行来使用多 GPU 并行优化 Caffe。

对于对并行和代码性能优化不太了解的人员，笔者建议先阅读本系列的第一本书《并行算法设计与性能优化》[⊖]和第二本书《并行编程方法与优化实践》[⊜]。对于对并行或代码性能优化非常了解的人员，可按照自己相关的领域选择对应的章节阅读。

勘误和支持

由于笔者的水平有限、工作繁忙、编写时间仓促，而向量化、并行和代码性能优化又是一个正在高速发展的、和硬件及算法密切相关的、影响因素非常多、博大精深、兼具个人特色的领域，许多问题还没有统一的解决方案，虽然笔者已经努力确认很多细节，但书中难免会出现一些不准确的地方，甚至是错误，恳请读者批评指正。你可以将书中的错误或写得不好的地方通过邮件发送到 ly152832912@163.com，微博联系“异构并行计算 - 风辰”或微信联系“风辰”，以便再版时修正，笔者会尽快回复大家。如果有更多的宝贵意见，也欢迎发送邮件，期待能够得到读者朋友们真挚的反馈。

⊖ 本书由机械工业出版社华章公司出版，书号 978-7-111-50102-2。

⊜ 本书由机械工业出版社华章公司出版，书号 978-7-111-50194-7。

致谢

首先要感谢我的老婆，她改变了我的人生轨迹，让我意识到人生有如此多的乐趣。如果不是她容忍我晚上 10 点下班回家后还要接着写书，此系列的出版估计得晚个两三年。

感谢我的母校中国地质大学（武汉）的图书馆，那是我对并行计算产生兴趣的地方。感谢我的母校中国科学院研究生院和中国科学院图书馆，在那里我奠定了从事并行计算事业的基础。

感谢我的朋友陈实富、高洋等，如果没有你们，我还需要更多时间来提升水平。感谢我的老板王鹏、吴韧和汤晓欧，在这些技术大佬和“人生赢家”的指导下，我才会成长得如此迅速。

感谢我在英伟达（NVIDIA）时的实习生、百度时的实习生和我现在的同事及下属，如果不是你们的努力工作，我没有时间来写这个系列。

感谢机械工业出版社华章公司的高婧雅和杨福川老师，我本无意出版此书，是你们引导我将这本书付梓成书，是你们帮我修改书稿，让它变得可读、可理解，是你们帮我修正错误，是你们的鼓励和帮助使得我顺利完成全部书稿。

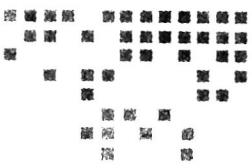
最后感谢我的爸爸、妈妈、姥姥、姥爷、奶奶、爷爷，感谢你们将我培养成人，并时时刻刻为我提供精神力量！

谨以此书献给我最爱的家人，以及众多热爱代码性能优化、向量化、并行计算的朋友们！愿你们快乐地阅读本书！

风辰

试读结束：需要全本请在线购买： www.ertongbook.com

目　录 *Contents*



第 1 章 Chapter 1

多核向量处理器架构

多核向量处理器的出现是为了更好地提供程序所需的性能，而处理器的峰值性能、程序代码能够实现的性能则与硬件的特性、组织结构紧密相关。

硬件的性能最终由程序代码实现。代码中的各个部分（如访存、计算）如何在硬件上执行，在硬件上执行时，其具体行为如何；这些问题的答案都影响了代码的最终性能。目前看来只有 OpenCL 才有可能在主流的多核向量处理器（CPU、MIC、GPU、FPGA 等）上都得到支持（即一份代码可以在这些平台上运行），故本章只介绍 OpenCL 程序如何映射到 CPU 和 GPU 上执行。

处理器和程序是两个相互独立、而又相互联系的事物。在计算机编程的发展史上，有些看起来相互矛盾的几个现象一直在共同发展，即：

1）程序要独立于硬件，不能过分依赖硬件。为了更好地达到这一目标，有解释型编程语言、运行时编译（Just In Time，JIT）等技术支持，如 CUDA 的 JIT 驱动能够编译 PTX 为可执行代码，Python 和 Bash 会解释执行代码，Java 虚拟机和 C# 的 CRT 则上升到另外一个层面。

2）依据硬件的特性设计程序。为了发挥硬件的性能，许多软件开发人员不得不

依据硬件的特性来设计程序，比如 X86 处理器是否支持 popcount（用来计算二进制表示的数据中 1 的数量）就可能导致完全不同的算法设计；在基于 GPU 的异构并行计算平台上，PCI-E 的带宽在很大程度上影响了算法的设计策略。

3）依据程序来设计硬件。为了让程序在自家的处理器上运行得更快，硬件生产商会依据一些典型的应用的计算特点来设计处理器，让处理器在这些典型的应用上性能非常好。这类典型的应用有 Stencil、矩阵乘等。

从某种程度上说，这些现象是相互矛盾的，一方面要求独立于硬件，另一方面又要求依赖硬件；从另外一方面来说，这些现象又是相互融合的，软件开发人员依据硬件设计程序和硬件生产商依据典型应用程序来设计硬件都是为了让应用的计算性能更优；而不同的编程语言希望独立于硬件则是为了让使用该语言编写的应用能够运行在更多的平台上。

本章将介绍目前市场上的主流多核向量处理器（如 AMD GPU、NVIDIA GPU、Intel MIC、ARM A15 CPU 和 Intel Haswell）的硬件组织架构，以及 OpenCL 程序如何映射到这些硬件上执行。为简单起见，本章只介绍和性能优化相关的部分。

1.1 众核系统结构

众核是指处理器核数比较多，通常指处理器核心数量在几十个或以上。目前主流的众核处理器都是多核向量处理器，如 NVIDIA 和 AMD 的 GPU、Intel 的 MIC 等。

目前的众核处理器都是通过 PCI-E 总线连到主板上，作为 CPU 控制的一个硬件加速器使用。Sandy Bridge 已经完全集成了内存控制器和 IOH，这意味着连接到不同的 CPU 上的众核处理器会连接到不同的 IOH（I/O Hub）。如图 1-1 所示为一种简单的两个 GPU 连到两个处理器的情形。

在目前的 Intel 和 AMD 的 X86 处理器之间，分别通过 QPI 总线和 HT 总线连接多个 X86 多核向量处理器。由于处理器集成了内存控制器，因此在多路处理器上编

试读结束：需要全本请在线购买： www.ertongbook.com