HZ BOOKS 华章IT

[PACKT] PUBLISHING

资深软件开发人员撰写，融合作者多年OpenCL使用经验和异构编程心得，系统讲述OpenCL的核心概念、技术及实用技巧

从软件开发者的角度，深入剖析OpenCL异构并行编程技术，通过50多个OpenCL案例及大量示例代码，帮助读者快速掌握异构并行编程技术并理解高性能计算

高性能计算技术丛书

OpenCL Parallel Programming Development Cookbook

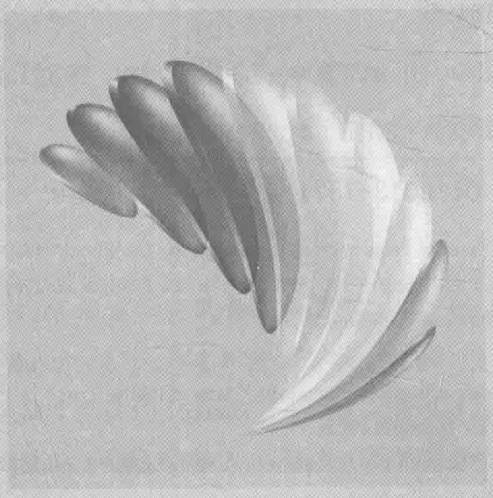
OpenCL异构并行编程实战

[美] 雷蒙德·泰（Raymond Tay）著

张立浩 译

机械工业出版社
China Machine Press

高性能计算技术丛书



OpenCL Parallel Programming Development Cookbook

OpenCL异构并行编程实战

[美] 雷蒙德·泰（Raymond Tay）著
张立浩 译

机械工业出版社
China Machine Press

图书在版编目（CIP）数据

OpenCL 异构并行编程实战 /（美）泰（Tay，R.）著；张立浩译 . —北京：机械工业出版社，2015.9
（高性能计算技术丛书）
书名原文：OpenCL Parallel Programming Development Cookbook

ISBN 978-7-111-51561-6

I. O… II. ① 泰… ② 张… III. 图形软件－程序设计 IV. TP391.41

中国版本图书馆 CIP 数据核字（2015）第 223179 号

本书版权登记号：图字：01-2013-7574

OpenCL 异构并行编程实战

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：谢晓芳　　责任校对：董纪丽
印　　刷：中国电影出版社印刷厂　　版　　次：2015 年 10 月第 1 版第 1 次印刷
开　　本：186mm × 240mm　1/16　　印　　张：15.25
书　　号：ISBN 978-7-111-51561-6　　定　　价：59.00 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

华章IT
HZBOOKS | Information Technology

About the Author 作者简介

Raymond Tay 过去 10 年来一直从事软件开发工作，他喜欢使用的编程语言包括 Scala、Haskell、C 和 C++。Raymond 于 2008 年开始接触 GPGPU 技术，最初使用的是 NVIDIA 提供的 CUDA 工具包和 AMD 提供的 OpenCL 工具包，然后使用的是 Intel 工具包。2009 年，他决定将正在进行的 GPGPU 项目提交给编委会，该编委会负责由 Morgan Kauffmann 出版的“GPU 计算宝典”的编撰工作。尽管其作品未入选最终的出版物，但他仍然很高兴能入围候选名单。在那之后，他参与了多个使用 CUDA 和 OpenCL 中所提供 GPGPU 技术与技巧的项目，同时对云计算中的函数编程范例和相关应用充满热情，这使得他开始研究通过使用 GPGPU 技术和函数编程范例加速云中应用程序的各种途径。他非常重视持续学习，并且希望能够长久坚持学习。

如果没有妻子和家庭成员的鼎力支持，本书不可能出版，因为我花费了大量周末和夜晚时间编写本书而没有时间陪伴他们，这样我才能完成本书并及时交稿。感谢 Packt Publishing 给我这个机会从事此项目，并且我从编辑和校对团队处获得了许多帮助。同时我也要感谢 Oracle 的高级首席软件工程师 Darryl Gove 和 NVIDIA 的 CPU 架构师 Oleg Strikov，他们通过卓尔不群的才智纠正了本书中的诸多错误。最后要感谢我的经理 Sau Sheong，他鼓励我开始此项目。再次谢谢各位！

审校者简介 *About the Reviewers*

Nitesh Bhatia 是拥有信息和通信技术（ICT）背景的技术专家，擅长计算和设计研究。他在 Infosys Design 担任用户体验设计人员，目前是位于印度班加罗尔的印度科学研究院的博士后。Bhatia 的研究领域包括视觉计算、数字人体建模和应用人机工程，并且喜爱研究不同的编程语言、计算平台、嵌入式系统等。他是几家社交媒体创业公司的创始人，而在闲暇时间，他则是狂热的摄影师和艺术爱好者，在其博客 Dangling-Thoughts（http://www.dangling-thoughts.com）中不断更新他的创造性作品。

Darryl Gove 是 Oracle Solaris Studio 团队的高级首席软件工程师，致力于为目前和未来的处理器优化应用和基准。他还是如下书籍的作者：《Multicore Application Programming》《Solaris Application Programming》和《The Developer's Edge》。Gove 的博客地址为 http://www.darrylgove.com。

Seyed Hadi Hosseini 是软件开发人员和网络专家，他在 16 岁时就获得了 MCSE、CCNA 和 Security+ 等证书，从此开始了他的职业生涯。他决定从事开源技术领域的工作，而 Perl 编程就是起点。Hadi 近 10 年来一直专注于 Web 技术和软件开发，他也是开源课程的导师。目前，Hadi 获得了 Linux Professional Institute、Novell 和 CompTIA 的 Linux 专家认证（LPI、LINUX+、NCLA 和 DCTS）。他的主要研究领域之一是高性能编程。他发表的第一篇科学论文在 2012 年举办的“第四届伊朗生物信息学大会”上获得最佳文章奖。在此论文中，他为基因组和蛋白质组中的 SSR 制定了超快速处理算法，其中就使用 OpenCL 作为 C 语言中的 GPGPU 编程框架，并且借助了 GPU 的大规模计算能力的优势。

Kyle Lutz 是软件工程师和纽约 Kitware 公司科学计算团队的一员。他拥有加利福尼亚大学生物科学学士学位，毕业后多年从事以 C++ 和 OpenCL 编写科学模拟、分析和可视化软件的工作。他还是 Boost.Compute 库的首席开发人员，这是基于 OpenCL 的 C++ GPU/ 并行计

算库。

Viraj Paropkari 于 2004 年毕业于印度普纳大学计算机科学系，并于 2008 年在美国乔治亚理工学院获得计算机科学硕士学位。目前他是 AMD 公司的高级软件工程师，致力于使用 OpenCL 对 CPU 和 GPU 的相关应用进行性能优化。他还致力于应对在大规模分布式系统上运行的大数据和高性能计算（HPC）应用所面临的新挑战。他之前在美国国家能源研究科学计算中心（NERSC）担任了两年的系统工程师，致力于建立世界上最大的超级计算机之一，该计算机运行并优化科学应用。在此以前，Paropkari 是伊利诺伊大学香槟分校计算机科学系的并行编程实验室（PPL）访问学者，同时也是美国橡树岭国家实验室的研究学者，美国橡树岭国家实验室是美国主要的研究实验室之一。他还在印度孟买的印度理工学院和印度塔塔基础研究所（TIFR）中从事关键任务型飞行模拟器的相关软件开发工作。Paropkari 是被授予高性能计算创新优秀奖的团队的主要贡献成员，该团队提升了 CFD 代码的运行速度，并且第一次实现了对现实燃油喷雾相关应用的模拟。模拟此问题的能力有助于将设计周期缩短至少 66%，并且为相关物理现象提供全新的见解，从而提供具有增强属性的喷雾。

前　言 *Preface*

欢迎阅读本书。本书并不是浅尝辄止式的入门书籍，而是由开发人员编写且面向开发人员的专业图书。有些读者可能会对本书感到熟悉，而另一些读者则会感到陌生。本书浓缩了我使用 OpenCL 的经验，但更重要的是对异构计算环境进行编程的心得。我希望与读者分享我所掌握的知识，并且决定采取按照主题分类问题的组织方式。我力求让这些主题保持简洁，但不得不承认，其中一些主题有点长。这样做的原因在于选择的问题多种多样，而向读者展现这些问题是因为本书中的章节描述如何将相关技术应用于当前或未来的工作。本书有望成为一份有用的参考手册，随时供你查阅。我无疑希望这些问题的解决方案可以像帮助我一样协助你。

本书从软件开发人员的角度进行编写，面向不仅希望知道如何以并行方式编程，还希望了解如何以并行方式思考的那些读者。依我看，后者比前者更为重要，但两者隔离都不能解决任何问题。本书通过代码加强读者对每个概念的理解，并通过介绍更多主题对这些概念进行扩展。

本书的组织形式可以帮助你熟悉 OpenCL 的核心概念，从而轻松进入 OpenCL 领域。然后，我们会深入讨论这些概念，具体方式是将新获得的知识应用于各个主题以及在工作中会遇到的一般性并行计算问题。

为最有效地利用本书，强烈建议读者是软件开发人员或嵌入式软件开发人员，并且有兴趣了解并行软件开发，但并不真正知道从何处及如何开始学习。理想情况下，你应该了解一些 C 或 C++ 知识（可以选择 C，因为它相对简单），并且熟悉使用跨平台的生成系统，如 Linux 环境中的 CMake。CMake 的优点是它允许为熟悉使用 Microsoft Visual Studio、Apple XCode 或其他一些集成开发环境的开发人员建立生成环境。必须承认，本书中的示例没有使

用这些工具。

本书内容

第 1 章通过介绍使用 OpenCL 的目的和动机来为后续内容做好铺垫。在相应主题中概述了核心概念，这些主题涉及设备的内在本质和它们之间的交互；此外也通过真正可正常运行的代码介绍这些概念。读者将了解相关上下文和设备，以及如何创建在这些设备上运行的代码。

第 2 章讨论 OpenCL 中的缓冲区对象以及划分数据的策略。随后，读者将学习工作项的定义以及如何利用 OpenCL 抽象化实现数据划分。

第 3 章解释 OpenCL 提供的两种常规数据类型，即标量和向量数据类型，介绍如何使用这些数据类型解决不同的问题，以及 OpenCL 如何抽象化处理器中的原生向量架构。该章也会展示如何通过 OpenCL 产生可编程的向量。

第 4 章讨论 OpenCL 为解决日常问题而提供的各种函数，例如几何、置换和三角函数。该章还介绍如何使用对应的向量化函数加快执行速度。

第 5 章给出典型 OpenCL 开发的生命周期。该章也讨论一些数据划分策略，这些策略依赖于对所讨论算法的认知程度。读者会在不经意间发现，并非所有算法或问题都需要相同的处理方式。

第 6 章将指导你使用索贝尔的方法构建边缘检测滤波器。同时还会介绍一些数学概念，包括一维和二维中的卷积理论以及相应的代码。最后，该章介绍如何在 OpenCL 中进行性能分析及其在本主题中的应用。

第 7 章通过研究矩阵乘法的并行化形式以及应用从串行到并行的转换来讨论矩阵乘法的并行化。然后，该章将通过讨论如何增加计算吞吐量和提升缓存利用率来优化矩阵乘法。

第 8 章讨论并行计算的环境以及用于解决该问题的传统方法，即通过充分的数学计算实现共轭梯度。一旦读者对共轭梯度有直观了解，该章将介绍稀疏矩阵的各种存储格式如何影响并行计算，然后具体讨论 ELLPACK、ELLPACK-R、COO 和 CSR 格式。

第 9 章介绍各种排序算法，其中重点讨论并行排序网络，也称为双调排序。就像在其他所有章节中所做的那样，该章完成相关主题的讨论，具体包括给出相应理论及其串行实现，通过转换实现并行化，然后开发最终的并行版本。

第 10 章介绍基于排序算法（如快速排序）的非比较形式的经典示例，在这些示例中，此类算法更适合 GPU 架构。该章也会介绍另一种核心并行化编程技术，称为归约，帮助读者

直观了解归约如何有助于基数排序更好地执行。基数排序的相关主题也演示了多种内核编程，突出介绍了它们的优缺点。

学习本书的具体要求

读者需要能够在 Linux 环境下轻松工作，因为本书中的示例是针对 Ubuntu 12.10 64 位操作系统进行测试的。下面是相关要求：

- GNU GCC C/C++ 编译器版本 4.6.1（最低版本要求）
- AMD、Intel 和 NVIDIA 提供的 OpenCL 1.2 SDK
- AMD APP SDK 版本 2.8，带有 AMD Catalyst Linux 显卡驱动程序版本 13.4
- Intel OpenCL SDK 2012
- CMake 版本 2.8（最低版本要求）
- Clang 版本 3.1（最低版本要求）
- Microsoft Visual C++ 2010（如果你在 Windows 上工作）
- Boost Library 版本 1.53
- VexCL（由 Denis Demidov 提供）
- AMD 提供的 CodeXL Profiler（可选）
- 保证每天 8 小时睡眠
- 开放的思想和虚心的态度
- 一杯特浓咖啡或其他提神的饮料

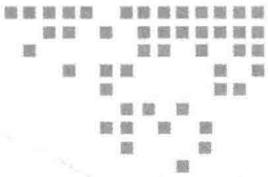
本书读者对象

本书面向特定的软件开发人员，这些开发人员希望了解可以用他们新购买的 CPU 或 GPU 做些什么，他们购买这些硬件的目的并不是用来玩计算机游戏。话虽如此，本书并不打算介绍仅可运行在家庭工作站上的算法。本书非常适合于有 C/C++ 实操经验的开发人员，以及希望学习如何使用 OpenCL 编写在异构计算环境中执行的并行程序的开发人员。

示例代码下载

本书中示例的代码可以从 http://www.packtpub.com/support 上注册并下载。

Contents 目　录



第 1 章 Chapter 1

使用 OpenCL

本章将介绍如下主题：

- 查询 OpenCL 平台
- 查询平台上的 OpenCL 设备
- 查询 OpenCL 设备扩展
- 查询 OpenCL 上下文
- 查询 OpenCL 程序
- 建立 OpenCL 内核
- 建立命令队列以及对 OpenCL 内核排队

1.1 引言

我们首先回顾一下计算的发展历史，并从以下角度探究 OpenCL 的重要性所在：OpenCL 旨在统一异构设备的软件编程模型。OpenCL 的目标是为个人计算机、服务器和手持设备 / 嵌入式设备中现代处理器的跨平台、并行编程制定完全免费的标准。Khronos 组织负责完成 OpenCL 标准的制定工作，许多知名公司也参与其中，如 Intel、ARM、AMD、NVIDIA、QUALCOMM、Apple 等。借助 OpenCL，只需要编写一次软件，就可以顺利地在各种支持此标准的设备上执行软件。OpenCL 在此方面类似于 Java，其优势在于现在可以采用统一的方法在这些设备上进行软件开发，具体实现方式是：通过各种数据结构揭示硬件，这些数据结构借助可编程应用接口（Application Programmable Interface，API）与硬件

交互。目前，OpenCL 支持包括 x86、ARM、PowerPC 在内的各种 CPU 以及 AMD、Intel 和 NVIDIA 提供的各种 GPU。

开发人员已深刻体会到开发跨平台兼容软件的重要性，因为这样他们就可以在任意舒适的平台上开发应用程序，更不用说这也提供了一种一致的模型，借助该模型，可以将想法在程序中明确表述出来，而程序可以在支持此标准的任何设备上执行。然而，跨平台兼容性也意味着异构环境的存在，长期以来，开发人员都不得不学习并努力克服在为异构设备（范围从执行模型到存储系统）编写软件时产生的问题。在这些异构设备上开发软件时经常引发的另一项任务是：人们还期望开发人员能够清晰表述和提取这些设备中的并行项。在 OpenCL 出现之前，人们建立了各种编程语言及其基本原理，希望借助它们来清晰表述其执行任务的设备上的并行项；这些语言包括 Fortran、OpenMP、MPI、VHDL、Verilog、Cilk、Intel TBB、Unified parallel C、Java 等。虽然开发人员可能认为有必要掌握这些工具，因为这样可以给他们的简历添砖加瓦，但这些工具都是为同构环境而设计的。让我们回过头来看一下，上述情况表明了并没有一种统一的方法来清晰表述异构环境中的并行项。开发人员需要借助这些工具来提高工作效率，但通常会涉及并行分解，因为该过程在很大程度上与硬件相关，从而增加了工作时间。更为严重的是，许多开发人员过去只需要在同构计算环境中处理任务，但在过去几年中，对异构计算环境的需求正在不断增长。

人们对异构设备需求的不断增长在一定程度上是因为需要性能更高、反应更灵敏的系统，并且随着虚拟现实技术的逐渐流行，提升性能的一种可行方法是增加专门的处理单元来提取异构设备中的每个并行项，因为这是达到所需功率效率的唯一方式。人们向混合运算转变的主要推动力可以追溯到 Anantha P. Chandrakasan 进行的一项研究，名为“使用变换优化功率”（Optimizing power using Transformations）。该研究得出如下结论：从根本上说，多核芯片（其运行频率稍微低于同时代的 CPU）实际上更加节能。未采用统一开发方法（例如 OpenCL）进行异构计算的问题在于，开发人员需要掌握多种类型的 ISA，并且存在多种并行级别及其存储系统。此处有必要提及 NVIDIA 开发的 GPGPU 计算工具包 CUDA，这不仅是因为它在很大程度上类似于 OpenCL，还因为它在学术界和业内被广泛采用。遗憾的是，CUDA 只能驱动 NVIDIA 的 GPU。

从异构环境中提取并行项的能力非常重要，这仅仅是因为计算应该是并行的，否则它就会违背 OpenCL 的整个初衷。幸运的是，主流的处理器公司都是 Khronos 组织领导的联盟体的一员，它们通过相应的机构主动实现了此标准。故事并未就此结束，不过好消息是开发人员意识到需要理解并行项及其在同构和异构环境中的工作方式。OpenCL 的设计目标就是清晰地表述异构环境中的并行项。

长时间以来，开发人员基本上忽视了这一事实：他们的软件需要利用身边的多核机器的性能。因此，他们继续在单线程环境中开发软件，但情况已经发生变化（前面讨论了这种

变化）。在多核领域中，开发人员需要掌握并发性的概念。并发性的优点是：通过合理地在线程之间分配资源，它可以最大化资源利用率。

使用多个处理单元并发执行软件，从而多个线程可以同时运行，这就是并行计算。开发人员面临的挑战是找出可以同时运行的线程并实现这种并发性。在 OpenCL 中，我们主要关注两个并行编程模型：任务并行和数据并行。

任务并行意味着开发人员可以创建并操作多个并发的任务。开发人员开发 OpenCL 的解决方案时，他们需要将问题分解为不同的任务，其中一些任务可以并发运行，这些任务会对应到并行环境的不同处理单元（PE），由其执行。另一方面，一些任务不能并发运行，甚至可能彼此独立。在任务之间共享数据的状况也带来了额外的复杂性。

在尝试实现数据并行时，开发人员需要重新调整考虑数据的方式以及如何并发读取和更新数据。并行计算中的一个常见问题是计算任意值数组中所有元素的和，同时存储中间的合计值。图 1-1 给出了一种可能的问题解决方法，其中应用了运算符，即任意二元结合运算符⊕。从概念上来说，开发人员可以使用任务来执行所输入的两个元素的加法，从而推导合计值。

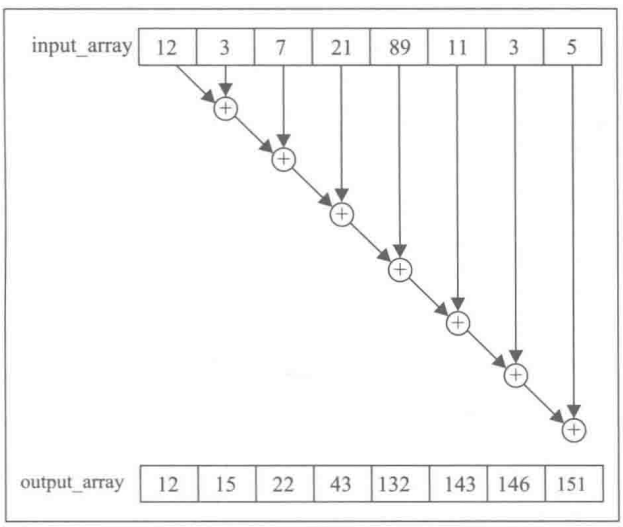


图 1-1

开发人员根据具体的问题来选择使用任务并行或数据并行，例如在遍历图表时应选择使用任务并行。无论开发人员更加倾向于使用何种模型，在通过 OpenCL 将程序对应到具体的硬件时，他们都需要面对相应的一组问题。在 OpenCL 面世之前，开发人员需要开发将在所需设备和通信器材上执行的模块，并且该模块需要能够与驱动程序进行 I/O 操作。此模块的一个示例是图形渲染程序，其中 CPU 初始化数据并建立所有对象，然后将渲染操作交给 GPU 完成。OpenCL 旨在利用检测到的所有设备，从而实现资源利用最大化，在此