<<C++多核高级编程>>

图书基本信息

书名：<<C++多核高级编程>>

13位ISBN编号：9787302222743

10位ISBN编号：7302222746

出版时间：2010年4月第1版

出版时间：清华大学出版社

作者：Cameron Hughes,Tracey Hughes

页数：561

译者：齐宁,董泽惠

前言

多核革命即将到来。
并行处理不再是超级计算机或集群的专属领域，入门级服务器乃至基本的开发工作站都拥有硬件级和软件级并行处理的能力。
问题是这对于软件开发人员意味着什么？
对软件开发过程会有怎样的影响？
在谁拥有速度最快的计算机的竞争中，芯片生产商更倾向于在单独的芯片上放置多个处理器，而不是提高处理器的速度。
迄今为止，软件开发人员尚能依赖于新的处理器，在不对软件做出任何实际改进的情况下提高软件的速度，但是这样的情况将成为过去。
为了提高总体系统性能，计算机供应商已经决定增加更多的处理器，而不是提高时钟频率。
这意味着如果软件开发人员希望应用程序从下一个新的处理器受益，就必须对应用程序进行修改以利用多处理器计算机。
尽管顺序编程和单核应用程序开发仍会有一席之地，但软件开发将向多线程和多进程转变。
曾经仅被理论计算机科学家和大学学术界所关注的并行编程技术，现在正处于为大多数人所接受的过程中。
多核应用程序设计和开发的思想如今成为人们关注的主流。
学习多核编程　　本书使用一般软件开发人员能够理解的术语介绍多核编程的基本知识。
为读者介绍了为多处理器和多线程体系结构进行编程的基础知识，对并行处理和软件并发的概念进行了实用的介绍。
本书介绍的是深奥的、不易理解的并行编程技术，但将使用一种简单、可理解的方式来介绍它们。
我们介绍了并发编程和同步的缺陷与陷阱以及应对之策，对多处理和多线程模型进行了直截了当的讨论。
本书提供了大量的编程实例，示范了如何实现成功的多核编程。
本书还包含了调试及测试多核编程的方法与技术。
最后，我们示范了如何使用跨平台技术来利用处理器的具体特性。
不同的视角　　本书的内容是为对多核编程和应用程序开发有着不同切入点的广大读者设计的。
本书的读者包含但不限于：　　·库及工具制作人员　　·操作系统程序员　　·内核开发人员　　·数据库服务器及应用服务器的设计人员及实现人员　　·科学应用程序员以及使用计算密集型应用程序的用户　　·应用程序开发人员　　·系统程序员　　每组人员都会从不同的视角来了解多核计算机。
有些人关心的是使用自底向上的方法，需要开发利用特定硬件和特定供应商的特性的软件。
对于上述人员而言，他们希望更加详尽地介绍多线程处理的知识。
其他人员可能对自顶向下的方法感兴趣，他们不希望为并发任务同步或线程安全的细节费心，而是倾向于使用高级库和工具来完成工作。
还有一些人需要混合使用自底向上和自顶向下的方法。
本书提供了对多核编程的多种视角的介绍，涵盖了自底向上和自顶向下的方法。
解决方案是多范型方法　　首先，我们承认不是每个软件解决方案都需要多处理或多线程。
有些软件解决方案通过使用顺序编程技术能够更好地实现(即使目标平台是多核的)。
我们的方法是以解决方案和模型作为驱动。
首先，针对问题开发出模型或解决方案。
如果解决方案要求某些指令、过程或任务并发地执行，那么就决定了最好使用哪组技术。
这个方法同强迫解决方案或模型去适合一些预先选择的库或开发工具的方法相反。
技术应当遵从解决方案。
尽管本书讨论库和开发工具，但并不偏向任何具体生产商库或工具集。
尽管书中包含了利用特定硬件平台的实例，但我们依赖跨平台方法，使用POSIX标准操作系统调用和

库，并且仅使用国际化C++标准所支持的C++特性。
面对多处理和多线程中的挑战和障碍，我们倡导组件方法。
主要的目的是利用框架类作为并发的构建块。
框架类被面向对象互斥量(mutex)、信号量(semaphore)、管道(pipe)、队列(queue)和套接字(socket)所支持。
通过使用接口类，显著降低了任务同步和通信的复杂度。
在我们的多线程和多处理应用程序中，控制机制主要是agent驱动。
这意味着在本书中，您将看到应用程序架构支持软件开发的多范型(multiple-paradigm)方法。
我们对组件实现使用面向对象编程技术，对控制机制主要使用面向agent编程技术。
面向agent编程的思想有时被逻辑编程技术支持。
随着处理器上可用内核数目的增加，软件开发模型将会越发地依赖面向agent编程和逻辑编程。
本书包含了对软件开发的这种多范型方法的简介。

<<C++多核高级编程>>

内容概要

为了提高系统总体性能，计算机厂商已经选择增加更多的处理器，而不是提高时钟频率。
相应地，如果您希望应用程序能够通过使用下一代处理器提高性能，就必须为了利用多处理器计算机而对应用程序进行改写。
这本非常实用的书教读者如何从顺序编程技术转移到并行和多线程编程技术，同时介绍了为多处理器和多线程架构编程的所有必备知识。
　本书的两位作者具有丰富的经验，虽然是并行处理和软件并发这些复杂的主题，但是采用了清晰、易于理解的方式来讲述它们。
通过他们进行多处理和多线程模型编程的实际方法，借助大量有用的实例，演示如何成功地完成多核编程，从而使读者能够充分利用新一代多核处理器的能力。
　本书主要内容　并发编程和同步带来的各种缺陷、陷阱和挑战　调试和测试多核编程的方法与技术　如何使川跨下台技术米利用处理器的特定特性　操作系统在多核编程中的任务　将框架类作为并发构建块加以利用的方法　如何通过使用接口类宋降低任务同步和通信的复杂性　本书适合于希望从事多核编程和多核应用程序开发的开发人员

作者简介

Cameron Hughes是一名专业的软件开发人员。
他是CTEST实验室的软件工程师，同时还是Youngstown州立大学的编程人员/分析师。
Cameron Hughes有着超过15年的软件开发经验，参与过各种规模的软件开发工作，从商业和工业应用到航空设计和开发项目。
Cameron是Cognopaedia的设计者，

<<C++多核高级编程>>

书籍目录

第1章 新的体系结构 1.1 什么是多核 1.2 多核体系结构 1.3 软件开发人员眼中的多核体系结构 1.4 总线连接 1.5 从单核到多核 1.6 小结 第2章 4种有影响的多核设计 2.1 AMD Multicore Opteron 2.2 Sun UltraSparc T1 多处理器 2.3 IBM Cell Broadband Engine 2.4 Intel Core 2 Duo处理器 2.5 小结 第3章 多核编程的挑战 3.1 什么是顺序模型 3.2 什么是并发 3.3 软件开发 3.4 C++开发人员必须学习新的库 3.5 处理器架构的挑战 3.6 小结 第4章 操作系统的任务 4.1 操作系统扮演什么角色 4.2 分解以及操作系统的任务 4.3 隐藏操作系统的任务 4.4 小结 第5章 进程、C++接口类和谓词 5.1 多核是指多处理器 5.2 什么是进程 5.3 为什么是进程而不是线程 5.4 使用posix_spawn( ) 5.5 哪个是父进程，哪个是子进程 5.6 对进程的详细讨论 5.7 使用ps实用工具监视进程 5.8 设置和获得进程优先级 5.9 什么是上下文切换 5.10 进程创建中的活动 5.11 进程环境变量的使用 5.12 使用system( )生成新的进程 5.13 删除进程 5.14 进程资源 5.15 异步进程和同步进程 5.16 wait( )函数调用 5.17 谓词、进程和接口类 5.18 小结 第6章 多线程 6.1 什么是线程 6.2 线程和进程的比较 6.3 设置线程属性 6.4 线程的结构 6.5 简单的线程程序 6.6 创建线程 6.7 管理线程 6.8 扩展线程接口类 6.9 小结 第7章 并发任务的通信和同步 7.1 通信和同步 7.2 对并发进行同步 7.3 线程策略方法 7.4 工作的分解和封装 7.5 小结 第8章 PADL和PBS：应用程序设计方法 8.1 为大规模多核处理器设计应用程序 8.2 什么是PADL 8.3 谓词分解结构 8.4 小结 第9章 对要求并发的软件系统进行建模 9.1 统一建模语言 9.2 对系统的结构进行建模 9.3 UML与并发行为 9.4 整个系统的可视化 9.5 小结 第10章 并行程序的测试和逻辑容错 10.1 能否跳过测试 10.2 测试中必须检查的5个并发挑战 10.3 失效：缺陷与故障导致的结果 10.4 如何对并行程序实现缺陷排除 10.5 什么是标准软件工程测试 10.6 小结 附录A 并发设计使用的UML 附录B 并发模型 附录C 线程管理的POSIX标准 附录D 进程管理的POSIX标准

章节摘录

除了CPU外，主板上最重要的部件是芯片组（chipset）。
在图2-8中，芯片组是被设计为连接CPU和主板上其他部件的一组集成电路。
它是主板的集成部分，因此不能被移走或升级。
它用于和特定类别的CPU或CPU系列共同工作，以优化CPU性能和系统性能。
芯片组将数据在CPU和主板上其他部件之间来回移动，这些部件包括内存、显卡、I／O设备，如图2.8所示。
所有到CPU的通信都经由芯片组。
芯片组由两个芯片组成：北桥（Northbridge）和南桥（southbridge）。
之所以如此命名它们，是由芯片在主板上的位置以及它们的用途决定。
北桥位于北部区域，南桥位于南部区域。
两者都是作为设备之间的桥梁或连接，它们为部件提供桥接，以确保数据到达期望的位置。
·北桥，也被称为内存控制中心（memory controller hub），直接通过前端总线（FSB）与CPU通信。
它将CPIJ同高速设备连接起来，如主存。
它还通过一条内部总线将CPU同PCI.E插槽及南桥连接起来。
数据在到达南桥之前，首先要经过北桥。
·南桥，也被称为I／O控制器，它的速度要比北桥慢。
由于它不是直接连接到CPU，因此它负责主板上较慢的部分，如音频、磁盘接口等I／O设备。
南桥通过串行外设接口（Serial Peripheral Interface，SPI）、6个PCI-E插槽、图中未显示的其他I／O设备连接到BIOS支持。
SPI使用主从配置来允许南桥与BIOS支持之间的数据交换（每次1比特）。
它使用全双工方式，意味着数据可以双向传送。
2.4.2 InteI的PCI Express PCI.E或PCI Express是计算机扩展卡接口。
该插槽是作为在主板上与声卡、显卡和网卡的串行连接。
串行连接速度较慢，每次发送1比特。
PCI-E是高速串行连接，其工作方式更像是网络，而不是总线。
它使用一个开关来控制很多被称为lane的点到点的全双工f同时双向通信）串行连接。
每个插槽可以有4、8或16个lane。
每个lane有两对从开关到设备的线，一对发送数据，一对接收数据。
这决定了数据的传送速度。
这些lane从开关直接扇出到数据要去的设备。
PCI-E是PCI的替代产品，并且提供了更多的带宽。
设备不共享带宽。
加速图形端口（Accelera,ted Graphics Port，AGP）被PCI.E x16（16 lane）插槽所替代，它能够提供更高的数据传送速度（8GB／s）。

<<C++多核高级编程>>

版权说明