开发人员专业技术丛书

HZ BOOKS 华章科技

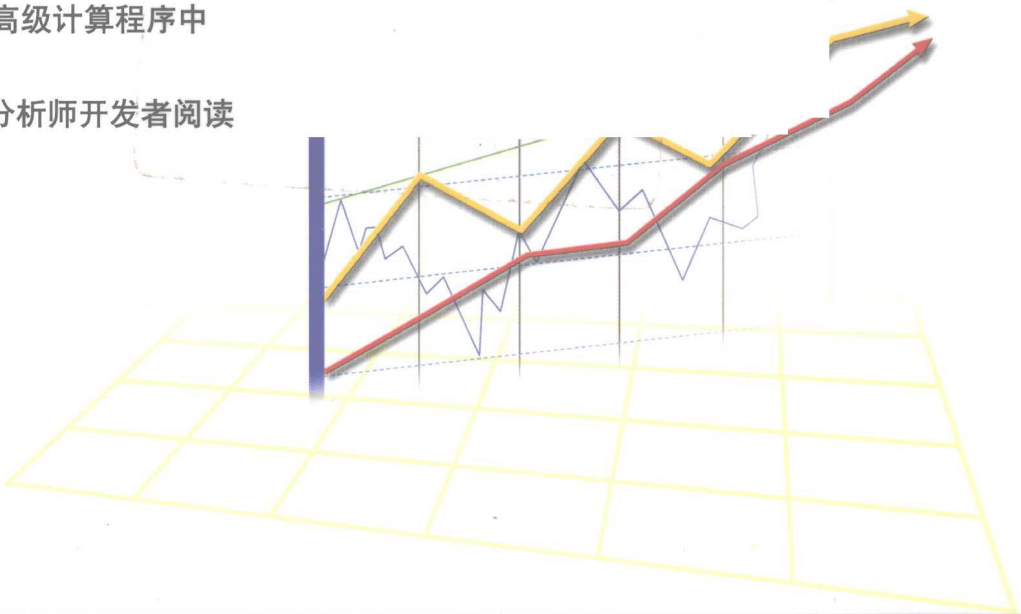
PEARSON

多核应用架构关键技术
软件管道与SOA

Software Pipelines
The Key to Capitalizing on the Multi-core Revolution

（美） Cory Isaacson 著 吴众欣 译

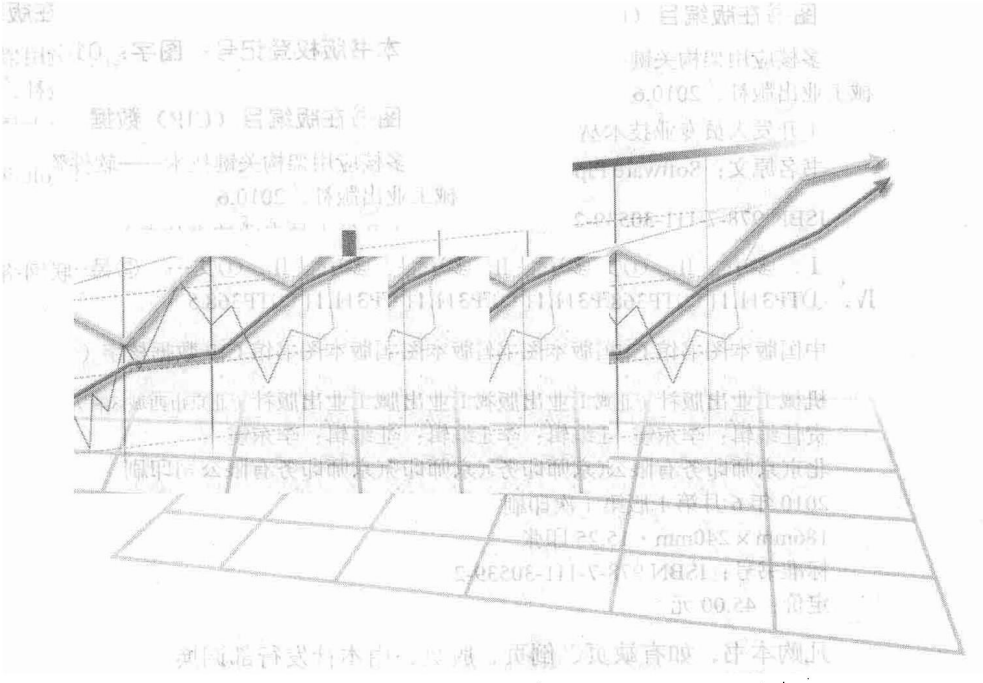
- 多核环境下的SOA解决方案。
- 解决应用程序的性能瓶颈，在任何SOA或高级计算程序中取得性能突破。
- 适合架构师、分析师开发者阅读



机械工业出版社
China Machine Press

开发人员专业技术丛书

多核应用架构关键技术
软件管道与SOA



机械工业出版社
China Machine Press

本书介绍软件管道如何工作，它们能完成什么样的任务，如何使用软件管道优化周期来应用它们。通过并行处理方法，扩展保证关键任务处理有序的应用程序。解决现存应用程序的性能问题，并且解决现存处理过程中的瓶颈问题。一个完整的、容易采用的管道参考框架。详细的代码示例反映了经过验证的管道模式。

本书适用于开发多核环境下软件的人员。

本书版权登记号：图字：01-2009-3514

图书在版编目（CIP）数据

多核应用架构关键技术——软件管道与 SOA /（美）艾萨克逊（Isaacson,C.）著；吴众欣译. —北京：机械工业出版社，2010.6
（开发人员专业技术丛书）
书名原文：Software Pipelines:The Key to Capitalizing on the Multi-core Revolution

ISBN 978-7-111-30539-2

Ⅰ. 多… Ⅱ. ①艾… ②吴… Ⅲ. ①并行程序—程序设计 ②互联网络—网络服务器—程序设计 Ⅳ. ①TP311.11 ②TP368.5

中国版本图书馆 CIP 数据核字（2010）第 078949 号

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：李东震
北京京师印务有限公司印刷
2010 年 6 月第 1 版第 1 次印刷
186mm × 240mm · 15.25 印张
标准书号：ISBN 978-7-111-30539-2
定价：45.00 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）88378991；88361066
购书热线：（010）68326294 ；88379649；68995259
投稿热线：（010）88379604
读者信箱：hzjsj@hzbook.com

译 者 序

SOA 带来的最大问题就是性能问题，restful 的概念也想解决其中的一部分，而从一个底层架构来说，软件管道当仁不让。ESB 与 BPEL 在 SOA 中扮演着公路与航运图的角色，而对于当今多核处理器的发展，SOA 的实践除了软件管道这个开源项目之外基本没有其他项目涉及。我的导师钱德沛教授首先提出数据驱动编程在多核领域会带来翻天覆地的变化，他引领我进入一个崭新的天地，随即我接触到释放多核处理性能的管道框架，加上以前关注过的 EIP 与 Java COPS，基本上 SOA 的天地已在胸中。无论是模式还是架构，从底至上，从顶至底，代码实现与测试用例，我都摸了个遍。

不过工作以后，又发现若干问题，学海无涯啊，学无止境，这些嘴上经常挂的话时时得以验证，将我的感觉也快磨迟钝了。不过我想一旦有所感受就要写出来与大家分享，否则我们就不能进步。以前与我合作翻译的朋友都转做其他事情，不过我还是愿意把信息从小受众转变成广域受众，因为我清楚自己做了些什么。

仅以此译稿献给我的老师、父母、爱人及可爱的女儿，她让我顺利升级，成为父亲！

我还希望我能把一本原创技术书籍在年底写出来，不知道我的孩子是否能给我充足的时间。

本书耗费了我大量的心力。翻译本书期间我体验了毕业、母亲住院、老婆怀孕与生产的若干经历，译者身心憔悴，若不是机械工业出版社陈冀康的督促与安慰，此书恐怕难以翻译成稿。

吴众欣

2010 年于北京

序

多核硬件这种新兴的产品也逐渐变得平常起来。由于能量消耗，高热量，以及量子物理学的不可预测性，半个世纪以来主要的计算机芯片厂商热衷于提升 CPU 时钟速度的活动基本停止了（套用爱因斯坦的话，CPU 不会赌博——不能掷骰子）。相反，厂商们采用多核架构，提供了更强的处理能力，而不是增加 CPU 的时钟速度。虽然这是一个合乎理性的行动，但对于现有的应用软件，它们很大一部分不能从多出的那部分核心的处理能力上获利。由于多核 CPU 的时钟速度降低了，应用软件运行起来会变得更缓慢，这种情况称为多核困境。

一般来说，多核困境在各种编程语言范围内都有出现，如 Java、C#、C++中等。这就是为什么各大技术厂商都投入大量的研究引领下一代的编程环境。但是对于那些已编写过的软件应该如何处理？任何软件应用的现实情况是，要从多核受益，应用程序必须编写为多线程处理方式，或者放置在能让它有效使用多线程的容器中。

虽然不存在“快捷”（随插随用）的解决办法，但在很多用例的情况下也有多个可用的开发工具和容器对多核困境有所帮助。然而，没有很多、很好的方法学来解决这一问题。在本书中，Cory Isaacson 勾勒出了一个系统的、逻辑的迁移到多核平台的规划和执行方法。

这样的硬件发展趋势，将带来数十亿行代码的软件产业在构造上的变化、迁移、优化或重写，来充分利用多核硬件。具有实践性与符合逻辑性的方法会让转型平稳变化。

现在，并行计算已经从一个冷门（边缘案例）变成企业软件的共同要求，让应用能够并行运行，不再是最有经验的程序员的工作。本书描述了进行并行计算软件开发的几种主流技巧。

举个例子，如同你将数据和 UI 层（用户界面层，也称人机界面，是指用户和某些系统进行交互方法的集合）与主要商业逻辑分开一样，一门创建并行计算的技术是通过您的整个开发小组将并发模型与应用逻辑分开。这样做使开发者关注应用的功能性，而不必在设计时担心流程是否是多线程的。此外，它可以成为将现有的单线程应用迁移到多核环境下的有效方法。

另外，本书讨论了面向服务的架构和多核之间的连接的方法。最基本的方法是把您的应用程序作为服务集合并部署在可以运行多个服务实例的容器中。

使用将 SOA 与多核关联起来的方法，服务可以成为你并发模型的重要组成部分。通过将并发与应用程序逻辑相分离的方法，可以很便捷地将现有的应用程序移植到多核环境中，并更有效地构建新的并行应用程序。这样更容易重新配置（而不是重新编码）你的应用程序，此后还能继续优化它们，并迁移到新一代的硬件环境中去——从 2 个核与 4 个核到 8、16、32 … 128 以及更多核。在面向服务架构中设计企业应用服务，可以更加容易地将并发与应用程序逻辑区分开来，就可以让它们一道工作。

如果你有一个单机应用程序，它仍起作用并且你不想重写代码，就需要做些工作。如果你打

算使用一个容器，需要确保它能满足您的业务应用需求，其中可能包括消息排序、插入与并入到业务流程中、人机交互和长期运行的需求。

在大多数并行计算方法中，传统的多线程编程方法是一种“线程级别”的方法。本书描述了一种“服务级别”的方法，它可以提供一种只花费较少的精力就能迁移到多核平台的方法，并且更容易配置。它只是补充而不是取代传统的线程级别的办法。

将你的工作迁移到多核环境下需要一些规划，但能完成比你想的还要多的工作。设计一种很棒的并发模型，让现有的应用程序可以继续为你多服务几年。本书为此尽力并提供一张漂亮的路线图。

Patrick Leonard
工程与产品策略副总裁
Rogue Wave Software
Pleonard@roguewave.com

前　言

我们现在正处在多核时代。出于消费者对计算能力的需求，我们都期望永无止境地提升计算能力，现在 CPU 制造商使用多核处理器来继续增加计算性能并维持这一长期趋势。如果我们要充分利用这巨大的能力，我们的业务应用程序必须学会“同时多干几件事”。然而，当出现特别是顶端的应用组件原本不是为并行计算设计的情况，传统的并行计算方法（如多线程、SMP 和集群）既受限制，开发起来又很艰难。

软件管道架构是一种新型架构，具体处理在多核时代使用并行处理方式所遇到的问题。它是一种新的解决问题方法。管道技术抽象出并行计算的复杂性，并使得利用新 CPU 处理能力解决商业应用成为可能。

我们主要是为软件架构师、应用开发人员以及需要高性能和可扩展商业应用的应用开发经理写本书。项目经理、软件质量保证专家和 IT 运营管理人员也会发现它很有用，但是本书的主要关注点是软件开发。我们的意图是使本书尽可能适合广大读者，提供开发工具让您可以快速学习，并将所学应用到自己的开发挑战过程之中。

本书分为 4 个部分，下面是我们在前言中要谈到的内容。

管道理论

管道理论部分是第 1 ~ 5 章，包括以下主题：

- 管道如何工作，软件管道所包含的基本概念和基本理论
- 管道能做什么
- 应用软件管道的方法
- 管道模式，包括适用于商业应用场景的软件管道的各种方案，为后面的章节中的示例打下基础

作为本书其余部分的理论基础，这部分的内容适用于所有读者。如果你是一个软件设计师或应用程序开发人员，你一定要学习第一部分的内容。如果你正在阅读管理角度的书籍，或者如果你的兴趣更为普遍不想涉及过多的技术细节，你也可以集中阅读本书的第一部分。

管道方法论

管道方法论部分，通过第 6 ~ 13 章的内容，演示了如何逐步使用软件管道优化周期（Software Pipelines Optimization Cycle，SPOC）方法来实现软件管道。为了说明该方法是如何工作的，我们用它来解决商业上的一些问题，用一个虚构的公司作为示例——管道银行公司（PBCOR）。每

一章都展示一个新的步骤，然后告诉您如何在 PBCOR 示例中使用该步骤进行处理。

这部分内容会让刚刚开始阅读本书的读者感兴趣，包括项目经理。PBCOR 示例给出了相当多的技术细节，因此，应用程序开发经理可能希望跳过这个更为复杂的示例。

管道示例

管道示例，通过第 14～22 章，包含我们为本书基于管道的参考框架开发的示例代码。我们已经为管道理论部分的每个主要管道模式开发了示例。你可以直接使用这些示例指导实际的应用程序。

这部分内容是针对软件架构师与应用程序开发人员的，他们直接参与管道实现。此外，IT 运营管理人员阅读了配置过程的章节也会觉得很有帮助，它展示了实际应用组件如何不做更改就能改变应用程序的可扩展性。我们推荐您仔细地阅读这部分的前三章。这些基础内容包括第 14 章、第 15 章与第 16 章。在这些之后你可能更想看第 17～22 章的高级示例内容，它们更加集中适合于你的特定应用场景。

软件管道的未来

在最后一部分，我们来谈一下对软件管道架构未来的设想。仍然有很多领域等待我们的开发，我们希望这一部分将激励读者帮助此技术进入主流方向并向前发展。

Web 站点

我们已经在 softwarepipe-lines.org 建立了一个软件管道技术的网站。该网站是为在使用或改进软件管道架构感兴趣的本书读者或其他人员建立的。您可以从网站下载下面各项：

- 软件管道优化周期方法论的工具和示例报告模板
- 管道参考框架的源代码
- 本书中所有示例的完整源代码
- 所有对管道技术与应用的讨论与文章。

我们希望您发现我们为您找到的软件管道技术是如此令人兴奋，你借此机会利用其能力，使用它来帮助克服自己系统在性能和扩展性方面的挑战。

致谢

有许多人为软件管道的发展作出过贡献，他们中的一些人也参与了这本书英文第 1 版的编写工作。我深深地感谢贡献创意、忘我投入、完成大量工作并对这个项目给予鼓励和支持的所有人。我不能说出每个人的名字，但正是他们所作的奉献才使得这项工作成为可能：

Ravi Palepu，是我们最好的朋友和商业伙伴，在讨论管道的构想过程中，陪我很长的时间，并帮助制定最初的概念，概括出现在的工作。

Rogue Wave Software 公司的 Patrick Leonard 与 David Haney 与我创建了管道的核心概念，

形成了当前工作的基础。

我还要感谢 Rogue Wave 软件公司在我任公司总裁期间及之后对我的支持。

Barbara Howell 是本书的插图作者还是令人难以置信的专业编辑，书中的插图都是她创作的；没有她我不会完成这个工作。她的执着、注重细节与天分让我能够通过书中的图表与图形就能将关键的概念介绍清楚，这真是太棒了。

Colin Holm 提供了大部分的代码示例，以及在第三部分对管道参考框架的介绍。对于本书最后的内容，Colin 的洞察力与工作是非常重要的。他有能力创建出从真实的商业应用程序员（他可以帮助将软件管道从概念转变到现实）的视角出发的示例。

最后，我要感谢我的家人，特别是我的妻子 Kim 为完成本书所作的巨大奉献。我还要感谢睿智的、善解人意的孩子们的理解与合作。事实上，我的大儿子，Devyn（一个真正的数学奇才），帮助我理清管道定律的公式；我的小儿子，Tyler，他是本书各种想法与如何展示的决策人。

作者简介

Cory Isaacson 是 Prelude Innovations 公司（一个特别关注先进软件技术产品孵化和推荐的公司）的 CEO。他积极参与领导信息技术工作 20 多年。Cory 担任 WebSphere Advisor 杂志的技术编辑与专栏作家，在数百场的公众活动和研讨会上发表演说，并撰写关于架构和实用性技术的大量文章。Cory 为数百个顶级架构师和专业开发人员（他们负责商业服务、娱乐、电信与软件行业的强大商业应用的开发与实现）提供指导。

最近 Cory 担任 Rogue Wave Software 公司的总裁，3 年来担任公司管理，建立新产品战略，并在 2007 年中期被一家私人控股公司成功收购。Cory 关注的重点是有效地解决诸如 SOA、可视化和支持实际业务应用的商品化资源的新技术开发和部署的挑战性问题。Cory 具有高性能的事务处理应用的专业知识，帮助先进 IT 企业对数据与业务数量上的急剧增长做出响应，同时还解除了经营成本的压力。最近，Cory 一直是在多核架构上为了改善应用程序性能而使用并行处理和可伸缩的数据库技术的积极倡导者。

Cory 在美国加州圣巴巴拉大学获得学士学位。

简介

纵观整个 IT 发展历史，专业开发人员一直在寻找提高关键业务应用性能的方法。计算机行业一直试图寻找答案，而且我们已经看到了很多解决方案、架构和方法。显然，这个问题不是小问题。在今天以信息为基础的经济结构中，往往由公司的软件性能来决定事业的成功与失败。有很多这样的案例：银行系统、交易系统、呼叫中心的运营部门、报告服务，以及许多其他行业也都依赖于高性能需求的应用。在这些行业中，公司开展业务的速度直接关系到该公司的生存能力。缓慢、不胜任或无响应的应用程序对公司运营与最终赢利与否造成破坏；最终，这些应用程序实际上可以让依赖于它的公司垮掉或受到损害有加剧。情况还不止如此，当我们做生意时越来越依赖于信息交流，性能会拖住需求（想进一步发展）的后腿。

还有另外一个问题。只让单个组件获得更快的性能是不够的。如果你的公司安装了一个新的

应用，如果业务进行了扩展，或如果处理的数据量向上攀升，你可能会突然需要性能逐次提高 5 倍、10 倍、20 倍，甚至更多。

其他因素也非常关键：你要如何让软件快速满足新的需求和应对竞争威胁？软件的推广及迅速采用面向服务的架构（SOA）是需要更加灵活的软件系统的确凿的证据。

SOA 是一种先进的技术。与早期的 IT 架构发展趋势相比，SOA 能更好地兑现自己的承诺。但它也向自己提出了挑战。出于下述原因，如果您使用 SOA 进行开发，性能和伸缩性显得更加重要：

- 一般来说，SOA 比以前单机或紧耦合系统的设计需要更高的计算能力。
- 需要注意的是，松散耦合服务实现的是以消息为中心的应用开发。开发者不仅要编写传统的处理逻辑，还必须处理信息传输、验证、解释、产生——所有这一切都是 CPU 与处理密集型的应用。
- 随着越来越多的公司使用 SOA 方法，我们可以预计消息传递量的增加和其对现有 IT 系统带来的巨大负担。潜在的不利影响将逐渐升级。

预测表明，在未来一两年内，使用 SOA 的公司将遇到性能问题。从历史上看，这并不是什么新鲜事，每当商业界采用新的软件架构，它便会经历痛苦的成长过程。过去二十年里，每个新的软件开发模式转变带来的震荡时期会历时 1 ~ 3 年的调整阶段（任何早期的 J2EE 用户都可以证明这一点）。在此期间，业务逐步采用了新的设计方法，当这样做时，它们仍面临与性能和伸缩性相关的问题。在许多情况下，软件开发者不能克服陡峭的学习曲线，部署的应用程序不能按预期运行，于是许多项目很快以失效告终。

直到最近，硬件成为这类不成熟架构的唯一可取之处。每当计算机业界取得了重大进展，主要在 CPU 性能方面，于是通过更快的芯片或使用一些其他机械式解决方案，可以消除性能瓶颈。现在这样的好情况已经不再存在。我们遇到微处理器技术的停滞期，主要由物理因素造成的，如电力消耗、发热与散热和量子力学效应。业界再也不能简单地增加单 CPU 的时钟速度。因此，从现在与能预见的未来，CPU 厂商正依赖于新的多核设计，增加处理能力。如果你想从多内核芯片获益，软件必须实现并行处理，它还不是今天主流应用普遍具有的能力。

让我们从企业的软件体系架构总结一下现今他们的真正需求：

- 对于性能和伸缩性，要有一个切实可行的并行处理办法
- 灵活性，让商业适应市场和其他的环境变化

特别是在使用传统的手段时，创建具有这些特性的应用是不容易的。此外，在现实世界中，做这样的模式转换工作需要具有商业知识和一定技术专长的专业开发人员。简言之，专业开发人员需要一系列为实现这些目标（针对业务应用在一个新的层面上进行并行处理）而设计的工具集。

因此，灵活、适当、实用的并行处理方法需要的是什么已成为一个问题。软件管道技术正是为了实现这个目的而开发出来的方法，它为专业的开发人员提供一套可用的开发工具和为应对今天的商业应用环境激烈的竞争提供可伸缩处理能力。

大家如何对待并行处理

作为本书中我们研究的一部分，我们希望找出软件社区对并行处理的一些想法，所以我们对

相关的Web文档进行了统计分析。我们的分析工具比较了在Web 上文件中使用的单词及其频率，来显示出某一特定主题的总趋势。结果很有意思，他们确认并行处理是现代计算挑战的重要解决方案。

为了分析，我们基于对“软件”这一主题词的查询，并查阅参考资料来找出该主题下的相关术语。我们是从下面的术语开始的：

- 多核
- 多线程
- 并行处理
- 并行编程

我们在本节中用若干个图表显示结果。图 0.1 显示了人们使用术语的频率。正如你所看到的，并行编程是最流行的术语，其次是多核，然后是并行处理。它给我们一个好想法来了解软件社区对该话题的讨论。

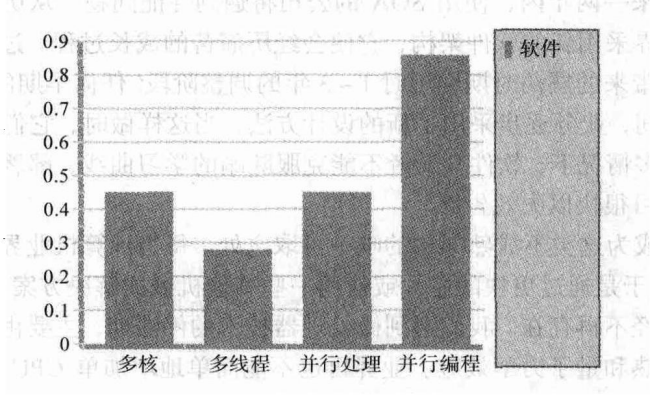


图 0.1　软件查询结果

要获得更详细的查询，我们按照以下特点将每个术语交叉结合起来分析：

- 复杂
- 困难
- 重要
- 知识
- 有用

在图 0.2 中，你可以看到并行处理的每个特点间的关系。并行处理因为具有两个关系最紧密的特点，被认为是“有用的”和“重要的”。

图 0.3 显示了并行编程和它的一些特点。并行编程肯定是“重要的”，但文档中有很高的比例提到，它是“困难的”。有趣的是，“知识”这个特点具有较高的排名，这并不奇怪，因为对于缺乏该技术经验的普通人来说并行编程很困难。

图 0.4 和图 0.5 显示多核心与多线程的特点。这两个图反映了与并行编程类似的信息。

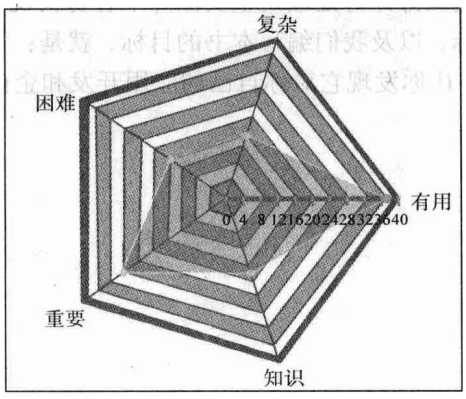


图 0.2　并行处理的特点

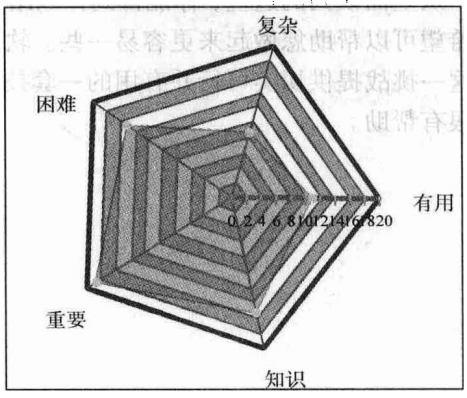


图 0.3　并行编程的特点

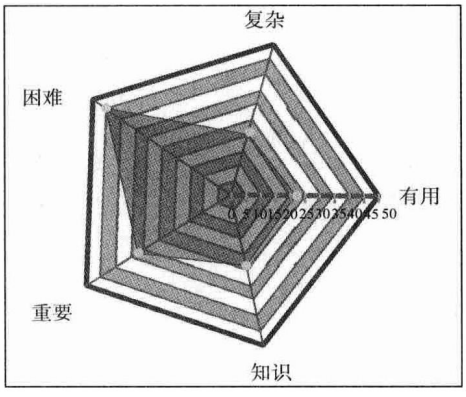


图 0.4　多核的特点

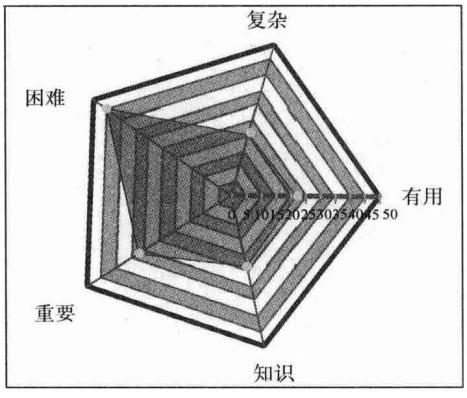


图 0.5　多线程的特点

图 0.6 给出包括所有术语和特点，它显示了每个组合的相对强度。你可以看到，并行处理被认为是“有用的”，而并行编程是“重要的”和“困难的”。

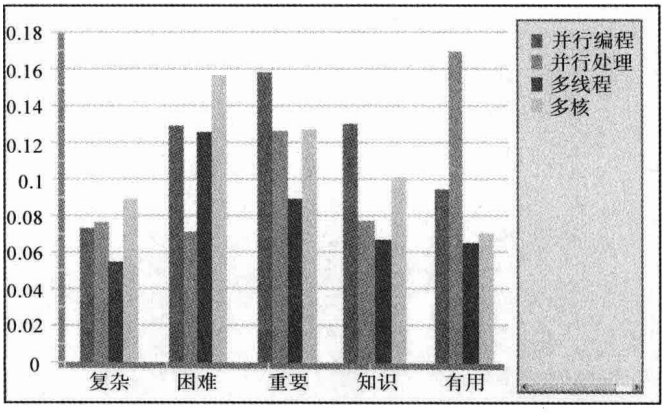


图 0.6　所有术语的特点

从上面谈到的这些，你能得到什么结论？看来人们认为并行处理是重要但并不容易实现。我们希望可以帮助您做起来更容易一些。软件管道的目标，以及我们编写本书的目标，就是：为应对这一挑战提供切实可行并有用的一套技术，我们决心让你发现它对你自己的应用开发和企业管理很有帮助。

目　录