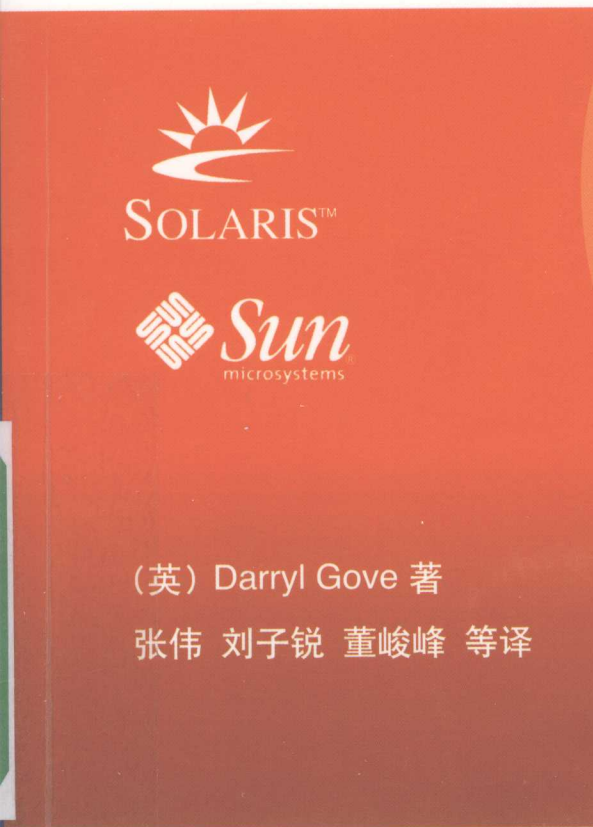
HZ BOOKS
PRENTICE HALL

Sun 公司核心技术丛书

Solaris 应用程序设计

Solaris Application Programming



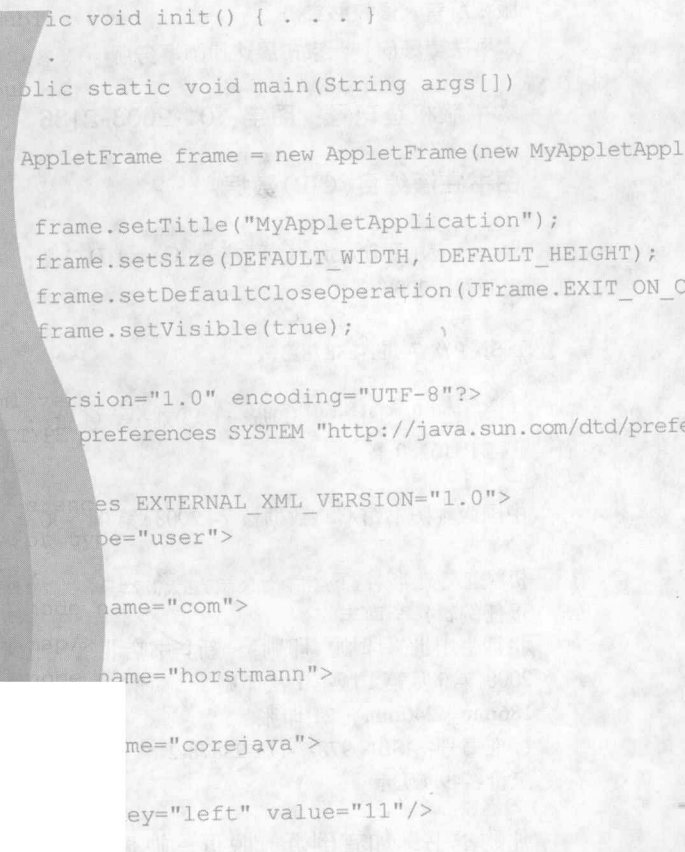
（英）Darryl Gove 著
张伟 刘子锐 董峻峰 等译

机械工业出版社
China Machine Press

Sun公司核心技术丛书

Solaris 应用程序设计

Solaris Application Programming



(英) Darryl Gove 著
张伟 刘子锐 董峻峰 等译

机械工业出版社
China Machine Press

本书介绍如何在 Solaris 操作系统上获得程序的最佳运行性能。全书共分五部分 13 章，包括常规处理器、SPARC、x64 处理器、信息工具、编译器、浮点数优化、库与链接、性能分析工具、校正与调试、性能计数器度量、源代码优化、多核多进程多线程以及性能分析。实用性强是本书的最大特点，全书覆盖了目前所有在 Solaris 上可以使用的开发工具，并介绍了它们的用法，而且书中还提供了大量实例工具的用法并解释其输出数据的含义。

本书内容丰富，层次分明，不仅适合软件开发新手阅读，也可供有一定经验的开发者参考。

本书版权登记号：图字：01-2008-2136

图书在版编目(CIP)数据

Solaris 应用程序设计/(英)格夫(Gove, D.)著；张伟，刘子锐，董峻峰译. —北京：机械工业出版社，2008.5

ISBN 978-7-111-23878-2

Ⅰ.S…　Ⅱ.①格…　②张…　③刘…　④董…　Ⅲ.操作系统(软件)，Solaris－程序设计　Ⅳ.TP316.89

中国版本图书馆 CIP 数据核字(2008)第 047390 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：李南丰
北京牛山世兴印刷厂印刷 · 新华书店北京发行所发行
2008 年 5 月第 1 版第 1 次印刷
186mm×240mm · 21 印张
标准书号：ISBN 978-7-111-23878-2
定价：49.00 元

凡购本书，如有倒页、脱页、缺页，由本社发行部调换
本社购书热线(010)68326294

译 者 序

由 Prentice Hall 出版的《Solaris™应用程序设计》(作者 Darryl Gove)是 Solaris 系列丛书中的一本。

本书的多语言翻译工作正在进行中，包括荷兰语、法语、德语、印度语、日语以及韩语等。当然，本书的中文版也即将与广大读者见面。

此书之所以受到如此的追捧，主要原因是 Solaris 操作系统正在被越来越多的用户所接受。随着 Solaris 操作系统的开源，Open Solaris 依靠其优异的性能和稳定的特性在全球迅速升温，官方网站的下载量节节攀升，各种有关 Open Solaris 的社区如雨后春笋般涌现出来，Solaris 操作系统越来越受到人们的青睐。用户在享受 Solaris 操作系统带来的种种好处的同时，还热衷于为 Solaris 的发展贡献自己的力量：积极参与社区活动，为扩展 Solaris 的功能贡献源代码。

但问题也随之而来：如何在 Solaris 上编写出性能出众而且稳定的应用程序呢？本书的推出恰逢其时地解决了这个棘手问题。引用一位读者的话，“这本书对 Solaris 平台上的编程技术做了详细的总结和分类，并告诉读者如何使用这些技术。我非常喜欢这本书并把它作为最新的 Solaris 参考手册，如果你对应用程序性能十分挑剔，是性能至上的狂热份子，那么这将是一本必备图书。”

由此可见，本书的适用读者对象十分广泛。无论是对应用程序性能分析一无所知的初学者，还是性能分析的专家，本书都同样具有指导和参考意义。即使在非 Solaris 平台上优化应用程序性能，本书的许多概念和方法仍然可以被借鉴和使用。

本书的翻译团队成员全部来自 Sun 中国工程研究院，包括：董峻峰、段少婷、焦梦葳、胡惠文、李志宏、刘子锐、吕华锋、孙勇、王秀燕、王宇博、许闽、薛东、薛晓舟、颜强、姚延栋、于继军、张伟、张扬眉、张羽和郑艺丁。为了使本书以最快的速度呈现在广大中文读者面前，翻译团队的每位成员都付出了艰苦的劳动。借此机会也要特别感谢翻译项目的发起方 Sun 中国技术社区和发起人蒋清野先生。

由于本书的翻译进度要求高，涉及技术面广，翻译难度较大；加之译者水平有限，书中难免有疏漏或错误，欢迎广大读者批评指正。

张 伟

2008 年 4 月

前 言

关于本书

本书旨在介绍如何在 Solaris 操作系统上获得程序的最佳运行性能。本书适用的读者是那些对 Solaris 上的可用工具感兴趣的开发者和软件架构师，以及那些关心如何充分利用系统中每一种性能的人员。

本书不仅适合性能分析和优化的新手阅读，也适合在这个领域有一定经验的开发者阅读。为了满足不同层次读者的需求，本书在介绍各种工具并进行深入讨论之前，首先综述了处理器的基本概念。

与其他的同类书籍相比，本书是一本实用性很强的入门书籍。人们在开发过程中经常会遇到两类问题。第一类问题是应该使用什么工具。针对这一问题，本书覆盖了目前所有在 Solaris 上可用的开发工具，并且介绍了它们的常见用法。第二类问题是如何理解这些工具的输出。书中提供了大量的例子，来演示工具的用法并解释其输出数据的含义。

对于编译器可以自动执行的优化技术，本书将避免解释手动启用它的方法。本书着重关注于如何利用适当的工具来定位问题，以及如何利用最简单的方法来解决问题。有时候，解决的方法是使用不同的编译选项优化应用程序，从而消除导致问题产生的热点代码。而有时候，解决的方法是修改编译器无法执行优化的代码；我会详细地解释编译器无法优化这些代码的原因。

目标和前提

本书要实现的目标如下：

- 全面地介绍影响处理器性能的部件。
- 介绍用于性能分析和性能提高的工具，包括操作系统所提供的工具和编译器所提供的工具。
- 介绍编译器以及编译器支持的提高性能的优化技术。
- 讨论 SPARC 和 x64 处理器家族的特征，并示范如何利用这些特征来提高应用程序的性能。
- 介绍利用多处理器或多线程来提高性能的可能性，以及有效地利用计算机资源的可能性。

本书假定读者可以熟练地使用 C 编程语言。书中的大多数例子都采用了这种语言。本书还假定读者了解一些处理器的基本原理和指令集的相关知识。本书不会详细地阐述处理器的内部细节，但是会介绍一些现代处理器的特征，这些特征可能影响应用程序的性能。

本书假定读者有 Sun Studio 编译器和相关的工具。这些工具都是可以免费下载的。本书

使用 Sun Studio 12 编译大多数例子。新版本的编译器将产生相似的结果。编译器通常安装在/opt/SUNWspro/bin/目录下，并且假定该路径已经设置在读者的环境变量中。

本书是基于 Solaris 10 的。在此之前的操作系统支持书中涉及的大多数工具。如果某个工具需要更新版本的 Solaris 的支持，那么我将在书中给出说明。

本书章节综述

第一部分　处理器综述

- 第 1 章：常规的处理器
- 第 2 章：SPARC 家族
- 第 3 章：x64 处理器家族

第二部分　开发工具

- 第 4 章：信息工具
- 第 5 章：编译器的使用
- 第 6 章：浮点数优化
- 第 7 章：库与链接
- 第 8 章：性能分析工具
- 第 9 章：校正与调试

第三部分　优化技术

- 第 10 章：性能计数器度量
- 第 11 章：源代码优化

第四部分　线程化和吞吐量

- 第 12 章：多核、多进程与多线程

第五部分　总述

- 第 13 章：性能分析

致谢

在本书的编写过程中，很多人都做出了贡献。Ross Towle 提供了多线程编程的章节大纲，并且对这章的内容做出了有价值的评论。Joel Williamson 多次阅读了早期的草稿，每次都给出了详细的评论和改进方法。我的同事 Boris Ivanovski、Karsten Gutheridge、John Henning、Miriam Blatt、Linda Hsi、Peter Farkas、Greg Price 和 Geetha Vallabhenini 也阅读了草稿，并提出了修改意见。在此，我要特别感谢 John Henning 对本书做了许多细致的改进。

我还要感谢所有花时间阅读本书并提供有效建议的各个领域的专家，他们是链接方面的 Rod Evans，调试器方面的 Chris Quenelle，提供了一些有用工具名字和评论的 Brian Whitney，提供建议的 Brendan Gregg，对编译器和源代码优化技术部分进行审阅的 Jian-Zhong Wang，提供关于浮点优化技术详细建议的 Alex Liu，对性能分析和多线程章节进行评论的 Marty Izkowitz，对多线程相关章节提出建议的 Yuan Lin、Ruud van der Pas、Alfred Huang 和 Nawal Copty，

对 MPI 进行评论的 Josh Simmons，了解 SPARC 处理器历史的 David Weaver，对 x64 处理器相关章节进行审阅的 Richard Smith，以及对本书提供总体建议的 Richard Friedman 。

在本书的出版过程中，很多人都提供了帮助，他们是 Yvonne Prefontaine、Ahmed Zandi 和 Ken Tracton。尤其感谢 Richard McDougall 在项目最后阶段提供的帮助。

特别感谢 Prentice Hall 的工作人员，包括编辑 Greg Doench 和产品经理 Julie Nahil。还要感谢 Techne 小组的产品项目经理 Dmitri Korzh。

最重要的是，感谢我的家人对我的支持和鼓励。Jenny 的冷静以及有效的建议帮助我完成了书中困难部分的编写；Aaron 即使对最现实的问题也抱有丰富的想象力，他成功地感染了我；Timothy 对分享生活中的乐趣有很大的热情，这对我的影响是无处不在的；Emma 出生在我完成本书的时候，她是上天赐给我的最珍贵的礼物。

目　录

第四部分　线程化和吞吐量

第五部分 总 述

第一部分　处理器综述

第 1 章　常规的处理器

1.1　本章目标

简单来说，处理器的功能就是从内存中取出指令，并执行该指令，如果需要的话，还将从内存取出数据，或将结果送入内存。但是，这样的描述遗漏了决定应用程序性能的许多重要细节。本章描述的是“常规的”处理器；也就是说，本章所描述的是，通常情况下处理器是如何工作的以及它由哪些部件构成。阅读完本章，读者将会了解有关处理器的一些术语，并且会理解一些应用在处理器设计方面的方法。

1.2　处理器的组成

每一台计算机的“心脏”是一个或多个中央处理器(CPU)。图 1-1 是 UltraSPARC T1 CPU 的照片。CPU 是计算机中执行计算的部件，组成计算机的其他部件包括内存芯片、硬盘、电源、风扇(用来冷却计算机)，以及其他使得计算机可以与外界通信的芯片(如显卡芯片及网卡芯片)。CPU 的底部有成百的“引脚”；[⊖]如图所示，它们组成了交织状的纹样。每一个引脚都是 CPU 和系统之间的一个连接。

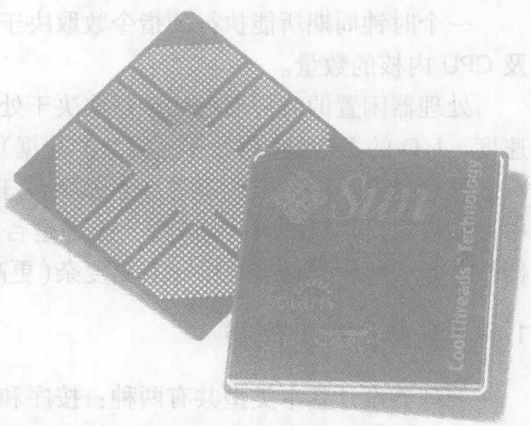


图 1-1　UltraSPARC T1 处理器

在 CPU 封装的内部是一个称为“内核”的小硅片。一个 CPU 包含一个或者多个用于计算的内核，本机上的或者是片上的一些存储器，这称为“高速缓存”(用于保存指令和数据)，以及系统接口(使得处理器可以和系统的其他部件进行通信)。

有些处理器只有一个内核。而图 1-1 中的处理器 UltraSPARC T1 有 8 个内核，每一个内核都可以同时运行 4 个线程。对于这个系统的用户来说，这看起来就像有 32 个虚拟的处理

⊖ 过去，引脚设计成连接在处理器基底上真正的插针。这种封装方式的问题是，引脚很容易被弄弯或者弄坏。近年来，芯片封装技术把引脚设计成球状或者片状。

器。每一个虚拟处理器对操作系统来说就像是一个完整的处理器，都能执行一条指令流。图1-2所示的是UltraSPARC T1处理器的核心，该图标示了CPU每个区域所执行的功能。

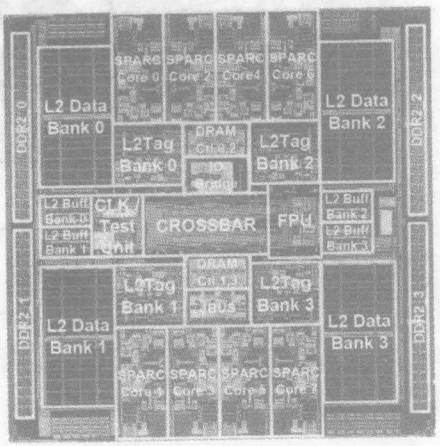


图1-2　UltraSPARC T1的核心

1.3　时钟速率

所有的处理器都以某种特定的时钟速率来执行。时钟速率可以从MHz到GHz[⊖]级别。时钟速率越高，通常意味着功耗越高。一条或者多条指令能够在一个时钟滴答内执行，所以，每秒被执行的指令数能够达到数百万条甚至数10亿条之多。每个时钟滴答称为一个“周期”。

时钟速率通常作为衡量一台处理器能力的显著特征，但时钟速率不足以衡量一台处理器的工作能力。

把时钟速率作为衡量标准常称为“兆赫神话”。处理器每秒能够处理的工作量取决于许多因素，时钟速率只是其中之一。其他因素还包括每个时钟周期处理器能够发出多少条指令，以及由于没有发出指令，而导致有多少个时钟周期被错失，这是一种令人惊讶的常见情况。处理器的性能还取决于处理器的设计以及当时它所负载的工作量。

一个时钟周期所能执行的指令数取决于可用的执行流水线的数量(将在第1.6节谈到)以及CPU内核的数量。

处理器闲置的时钟周期的数量取决于处理器的设计和所提供的高速缓存的容量、内存的速度、I/O的吞吐量(比如写入磁盘的数据)以及具体的应用。

处理器设计的一个关键选择点常关注于高速缓存是否需要增加(这将会减少等待从内存中读取数据所消耗的时钟周期)，或者是否将这些核心空间用于其他的功能，如更多的处理器核，是否每个处理器核中具有更复杂(更高的性能)的电路。

1.4　乱序执行处理器

处理器设计基本类型共有两种：按序和乱序执行处理器。乱序执行处理器一般在给定的时钟速率下能够提供更高的性能，但也更复杂并且消耗更多的能量。

在一个按序执行的处理器中，指令按照它所出现的顺序执行，如果前一条指令的结果还不可用，处理器将会等待(或说“停滞”)直到结果可用。这种方式依赖编译器来做好调度指令的工作以避免停滞。但这种方式并不是总有效，所以按序执行处理器会有一些停滞的时钟周期，不能执行新的指令。

一种减少停滞时钟周期数量的方法是允许处理器乱序执行指令。处理器会试图从当前的指令流中找到那些与当前停滞的指令相独立并能与之并行执行的指令。x64系列处理器都是乱序执行处理器。乱序执行的弊端是随着乱序程度的增加，处理器急剧地变得更加复杂。

⊖　兆赫(MHz) = 每秒100万个周期。千兆赫(GHz) = 每秒10亿个周期。

当指令流存在小间隙时，乱序执行是保持处理器利用率的好方法。但是，如果指令流中存在一个较大的间隙，比如在处理器等待数据从内存中取出时，那么乱序执行处理器将不会比按序执行处理器好多少。

1.5　芯片多线程

芯片多线程(CMT)是对复杂的乱序执行处理器的一种替代技术。按序执行处理器设计更简单并且比乱序执行处理器消耗更少的电能，但它存在更多等待前一条指令的结果变为可用的时间。CMT 被用在了 UltraSPARC T1 处理器中，有多个线程的指令在同一个核上执行。当一个线程被阻塞时，一个或更多个线程将准备好执行。因此，处理器的每一个核几乎在每一个时钟周期都可以执行一条指令——处理器的利用率提高了。

已往，强调的是尽可能地提高一个线程的性能，但 CMT 强调的是单位时间(吞吐量)能够做多少工作，而不是每份工作需要多长时间(响应时间)。

Web 服务器是一个很适合在 CMT 系统上运行的应用的例子。Web 服务器的性能通常以每秒钟它能够处理的页面数量来衡量，即以吞吐量为衡量标准。有多个可用的硬件线程来处理页面请求以提高系统的性能。另一方面，服务器的“响应性”通常取决于它花了多少时间将页面通过网络送出去，而不是它花了多少时间让 Web 服务器准备这个页面，所以处理器准备这个页面的响应时间和它将页面通过网络送出去的时间相比是很短的。

1.6　执行管道

为了能在每个时钟周期内执行大量的指令，处理器会有多个“管道”，每一个都可以处理一种特定类型的指令。这种类型的处理器被称为是超标量处理器。一般来说，它有内存管道(用来处理有关内存的操作，如装入和存储)，浮点管道(用来处理浮点算术)，整型管道(用来处理整型算术，如加法和减法)，以及分支管道(用来处理分支以及调用指令)。多执行管道的一个例子如图 1-3。

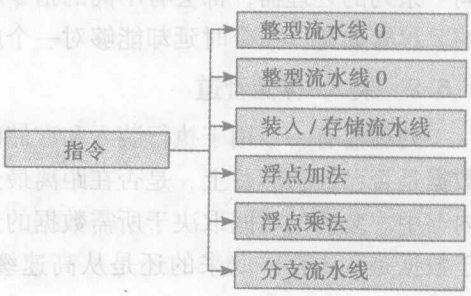


图 1-3　多执行管道示例

另外一种提高处理器时钟速率的方法是让指令执行流水线化，这意味着每条指令实际上花了多个时钟周期来完成，而且在每一个时钟周期内处理器只执行了整条指令的一小步。

流水线的一个例子是把执行一条指令的过程分为取指令(把下一条指令从内存中取出来)、解码(确定指令要处理器做什么)、执行(进行工作)和送回结果(提交指令执行的结果)，这是一个四级流水线，这种流水线如图 1-4 所示。这样做的优点是当一条指令正处于取指令逻辑阶段时，第二条指令可以处于解码逻辑阶段，第三条可以处于执行逻辑阶段，第四条可以处于送回逻辑阶段。流水线执行的速度

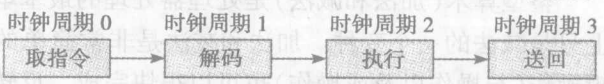


图 1-4　四级流水线