科学计算及其软件教学丛书/石钟慈主编

数值并行算法与软件

李晓梅　吴建平◎编著

科学出版社
www.sciencep.com

科学计算及其软件教学丛书

数值并行算法与软件

李晓梅　吴建平　编著

科学出版社

北京

内 容 简 介

本书为“科学计算及其软件教学丛书”之一，从计算数学的要求出发，系统介绍国内外新发展的数值并行计算方法，并进行可扩性与复杂性分析. 主要内容包括：并行计算基础理论，数值并行计算方法和并行计算的编程环境与编程实例. 全书深入浅出，串行、并行算法相结合，并行算法与实际编程例子相结合，易于理解和掌握. 每章附有习题，可供练习.

本书可作为应用数学、信息与计算科学专业高年级大学生和研究生的教材和参考书，也可供从事计算机科学、计算机软件、计算机应用方面的研究人员和学生参考使用.

图书在版编目(CIP)数据

数值并行算法与软件/李晓梅，吴建平编著. —北京：科学出版社，2007
(科学计算及其软件教学丛书)
ISBN 978-7-03-019461-9

Ⅰ. 数… Ⅱ. ①李…②吴… Ⅲ. 数值并行 Ⅳ. TP301.6

中国版本图书馆 CIP 数据核字(2007)第 113504 号

责任编辑：李鹏奇 李晓鹏/责任校对：曾 茹
责任印制：张克忠/封面设计：陈 敬

科学出版社 出版
北京东黄城根北街 16 号
邮政编码：100717
http://www.sciencep.com
铁成印刷厂 印刷
科学出版社发行 各地新华书店经销
*
2007 年 8 月第 一 版 开本：B5(720×1000)
2007 年 8 月第一次印刷 印张：18 1/4
印数：1—3 500 字数：340 000
定价：29.00 元
(如有印装质量问题，我社负责调换〈铁成〉)

《科学计算及其软件教学丛书》序

随着国民经济的快速发展，科学和技术研究中提出的计算问题越来越多，越来越复杂. 计算机及其应用软件的迅猛发展为这些计算问题的解决创造了良好的条件，而培养一大批以数学和计算机为主要工具，研究各类问题在计算机上求解的数学方法及计算机应用软件的专业人才也越来越迫切.

1998 年前后，教育部着手对大学数学专业进行调整，将计算数学及其应用软件、信息科学、运筹与控制专业合并，成立了 “信息与计算科学专业”. 该专业成立之初，在培养目标、指导思想、课程设置、教学规范等方面存在不少争议，教材建设也众说纷纭. 科学出版社的编辑曾多次找我，就该专业的教材建设问题与我有过多次的讨论. 2005 年 11 月在大连理工大学召开的第九届全国高校计算数学年会上，还专门讨论了教材编写工作，并成立了编委会. 在会上，编委会就教材编写的定位和特色等问题进行了讨论并达成了共识. 按照教育部数学与统计学教学指导委员会起草的 “信息与计算科学专业教学规范” 的要求，决定邀请部分高校教学经验丰富的教师编写一套教材，定名为 “科学计算及其软件教学丛书”. 该丛书涵盖信息与计算科学专业的大部分核心课程，偏重计算数学及应用软件. 丛书主要面向研究与教学型、教学型大学信息与计算科学专业的本科生和研究生. 为此，科学出版社曾调研了国内不同层次的上百所学校，听取了广大教师的意见和建议. 这套丛书将于今年秋季问世，第一批包括《小波分析》、《数值逼近》等十余本教材. 选材上强调科学性、系统性，内容力求深入浅出，简明扼要.

丛书的编委和各位作者为丛书的出版做了大量的工作，在此表示衷心的感谢. 我们诚挚地希望这套丛书能为信息与计算科学专业教学的发展起到积极的推动作用，也相信丛书在各方面的支持与帮助下会愈出愈好.

石钟慈

2007 年 7 月

前　言

作为利用计算机对物理应用问题进行研究的一种手段，数值模拟已经成为与理论分析、实验研究相并列的第三种科学研究手段. 由于数值模拟代价低，只要计算程序或软件研发成功，就可以用来对不同的工况进行反复模拟，同时所得结果不随主客观因素的改变而改变，所以在科学与工程应用领域越来越受重视. 而且，数值模拟不仅仅是一种研究手段，更已经成为衡量一个国家核心竞争力的关键指标，对国民经济的发展与国家安全起着重要的基础性作用，在数值天气预报、高新材料研究与设计、汽车设计制造、航空航天器设计制造、油汽资源勘探、地震资料处理、生命科学等领域中已成为必备工具. 全面核禁试后，数值模拟更是成为核武器唯一可行的研究手段.

虽然人们越来越认识到数值模拟这种研究手段的重要性，但是，其有效性有赖于两方面的研究成果. 计算机硬件设备是提供可用计算能力的物理载体，各种软件与相应计算方法是实现充分利用计算资源进行高效计算的基石. 只有在计算能力强的计算机上，设计高效的并行算法，开发出相应的并行计算软件，充分利用计算资源，才能实现高效的数值模拟.

在可用的计算能力方面，从 20 世纪四五十年代以来，世界各国就一直在投入大量的人力物力进行研究，不仅使单处理器速度大幅提升，并且研制出了阵列机、向量机、MPP 等许多种类的并行计算机. 特别是进入 21 世纪以后，国内外高性能并行计算机的发展更取得了重大进展，每秒万亿次、十万亿次乃至百万亿次的高性能并行计算机已相继研制成功并投入使用，从而使得以前科学计算中无法解决的问题成为可能.

相比之下，高性能并行数值模拟方法的研究与软件开发却比较滞后，由于缺少可靠的基础并行算法和软件，许多应用还无法真正实现高性能计算. 同时，为设计高效的并行计算方法与开发高效的应用软件，有必要对并行算法的性能进行评价. 正是基于这种考虑，在高性能并行计算机在我国的发展和应用日益普及，但许多应用部门、研究单位和高等院校却严重缺少并行计算方面人才的大背景下，本书专门对并行算法的性能评价、基本设计技术、常用的基础数值并行算法与并行编程技术等内容进行了介绍，为解决这种急需提供了一本入门教材.

本书分为 3 部分，共 11 章.

第一部分由 3 章组成，主要介绍并行计算基础理论与技术. 第 1 章介绍并行计算机，包括：单处理机体系结构、并行计算机的基本概念与分类、并行计算机

体系结构以及并行计算机未来的发展趋势. 第 2 章讲述并行算法性能度量方法，包括：基本概念与性能参数、并行算法加速比模型、并行计算的可扩展性以及实用例子的并行计算可扩展性分析. 第 3 章讲述并行算法的设计基础，包括：应用问题的基本求解过程、并行计算模型以及并行算法的设计方法与设计技术.

第二部分由 5 章组成，主要介绍数值并行算法. 第 4 章介绍矩阵并行计算，包括：稠密矩阵向量乘并行计算、稠密矩阵乘并行计算、稀疏矩阵向量乘并行计算、稀疏矩阵乘并行计算、经典软件以及 MPI 程序实例. 第 5 章介绍线性方程组的并行求解，包括：稠密矩阵的并行 LU 分解、三角形线性方程组的并行求解、三对角线性方程组的并行求解、带状线性方程组的并行求解、经典迭代法并行计算、预条件 Krylov 子空间迭代法、经典软件以及 MPI 程序实例. 第 6 章介绍矩阵特征值问题的并行计算，包括：几个重要概念、矩阵标准特征值问题的并行计算、矩阵广义特征值问题的并行计算、经典软件与 MPI 程序实例. 第 7 章介绍偏微分方程并行求解，包括：区域分解方法、多重网格法的并行计算、交替方向方法的并行计算、经典软件以及 MPI 程序实例. 第 8 章介绍离散变换的并行计算，包括：一维 DFT 的并行计算、二维及多维 DFT 的并行计算、并行多项式变换算法、实序列 DFT 的并行计算、离散余弦变换的并行计算、经典软件介绍以及 MPI 程序实例.

第三部分由 3 章组成，主要介绍编程环境和程序实例. 第 9 章介绍并行编程环境，包括：消息传递编程模型、消息传递接口 MPI、高性能 FORTRAN 数据并行编程 HPF 以及共享存储并行编程 OpenMP. 第 10 章介绍并行编程中需要研究的问题，包括：应用问题的分解、并行程序的性能优化、存储层次的管理以及并行 I/O. 第 11 章针对矩阵乘法，给出各种编程方式下的并行编程实例，包括：HPF 编程实例、OpenMP 编程实例、MPI 编程实例以及 MPI/OpenMP 混合编程实例.

本书主要关注并行算法的设计方法与技术、并行算法的性能度量方法和具体实例的编程，在介绍并行算法时，则侧重于科学计算领域中一些共性问题的并行计算方法. 因此，在学习本书之前，读者应先了解相关的数值方法与并行计算机体系结构，最好具有消息传递、共享存储以及高性能 FORTRAN 编程的实践经验. 这样，通过本书的学习，可以迅速掌握并行算法的设计方法、并行计算机的使用方法，并能迅速用于解决实际问题.

本书由李晓梅教授和吴建平副教授共同编写，其中第 1、2、3、9、10 章和第 6 章中 MPI 程序实例以外的部分由李晓梅教授编写，第 4、5、7、8、11 章和第 6 章中的 MPI 程序实例由吴建平副教授编写，全书由李晓梅教授与吴建平副教授进行统一校订.

本书的目的是为本科生和研究生的教学服务. 它的适用范围比较广泛，既可

作为应用数学、信息与计算科学专业学生的教材，也可作为计算机科学、计算机软件与应用专业学生的教材或这些方面研究人员的参考书.

在本书编写过程中，我们参阅了国内外相关的专著和教材，在此向各位作者表示感谢. 由于我们的学识水平、工作经验和能力有限，不当之处在所难免，敬请各位专家、学者和广大读者赐教，共同为促进我国高性能计算的深入发展而努力.

编　者

2007 年 4 月于北京

目　录

第 1 章　并行计算机

并行是一种提高计算机系统性能的有效方法. 并行计算机的出现，一方面是为了解决单处理机中存在的问题和约束，另一方面也是各领域大型应用问题的推动和技术进步的结果. 本章首先简要介绍单处理机体系结构，它是理解并行计算机体系结构的基础；然后转入并行计算机体系结构以及并行体系结构中一些重要问题的描述；最后讨论今后并行计算机的发展趋势. 通过这些介绍与讨论，读者可以对并行计算机系统有一个初步了解，为进一步深入学习并行机和并行算法奠定基础.

1.1　单处理机体系结构

众所周知，并行计算机的主要目标是获取更高的性能，并解除单处理机在同一时刻只能执行一个任务的限制. 为了更好地理解并行机的体系结构，本节简单介绍单处理机的两个主要部件：中央处理器 CPU(用于执行程序)和存储器(保存正在执行的程序及其操作数据).

1.1.1　中央处理器 CPU

CPU 是计算机的核心，它负责所有的计算并监控计算机的其他部件. 一个典型的 CPU 包含以下几部分：

(1) 算术逻辑部件 ALU(arithmetic logic unit)：执行计算功能，如加法与比较等.

(2) 浮点部件 FPU(floating-point unit)：执行浮点数的各种操作.

(3) 加载/存储(load/store)部件：实现对数据的加载和存储操作.

(4) 寄存器：快速存储器，用于存储程序的中间结果. 通常可细分为浮点寄存器 FPR(floating point register)和通用寄存器 GPR(general purpose register)两类.

(5) 程序计数器 PC(program counter)：存储下一条要执行的指令地址.

(6) 存储接口：提供对存储系统的访问. 一般，一级 Cache 在 CPU 芯片内为寄存器提供数据.

CPU 在时钟控制下进行操作，即每个时钟周期 CPU 执行一个操作(这里所指的操作意味着 CPU 在一个时钟周期内既可以执行少于一条的指令，也可以在多

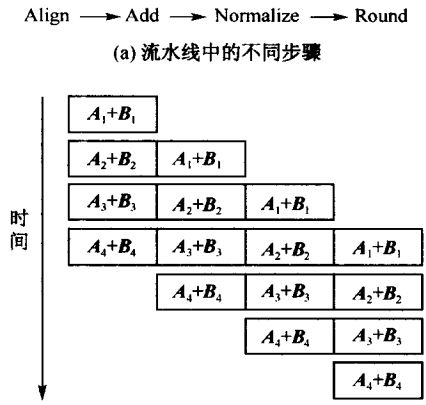
功能部件存在的情况下执行多条指令). CPU 的主频一直在持续大幅度增长,目前台式机的处理速度已达到每秒几十亿次.

目前 CPU 按基本操作集的不同分为复杂指令集计算机 CISC(complex instruction set computer)和精简指令集计算机 RISC(reduced instruction set computer)两大类. 它们能完成相同功能的操作,但 CISC CPU 需要更多指令才能完成这个操作,而 RISC CPU 需要的指令少且简单,因而可更快完成每条指令.

由 CPU 执行的指令组成了指令集. 指令集体系结构 ISA(instruction set architecture)体现了 CPU 相关指令集的设计. 采用 FORTRAN 和 C 语言编写的程序,经编译器编译后形成指令. 一般情况下,多数程序都要经过编译之后才能执行,故 ISA 对编译器的辅助或妨碍作用将极大地影响处理器的有效性. 在设计 ISA 时,重点要考虑内存的访问方式. 一般有"内存到内存"(memory-to-memory)和"加载—存储"(load-store)两种操作方式. 对于某些复杂操作(如浮点除),其内存访问频率相对较低,在 ISA 中可能包含读取内存多项数据的指令,ISA 执行这些指令,并将结果写回内存,称之为"内存到内存"的操作. 而内存访问频率相对较高时,则要求 ISA 为"加载—存储"体系结构. 在这种操作方式下,CPU 操作的所有数据均保存在快速寄存器中,内存中的数据在使用之前也必须先加载到寄存器,操作的结果数据送回寄存器,有时也称这种操作为"寄存器到寄存器"的操作. 存储操作将数据写回内存(通常,这个操作要通过多级缓存完成),加载操作和存储操作经常由一个加载/存储操作部件来处理.

由 CPU 执行的程序存放于内存中. 程序计数器指定下一条要执行的指令的存储地址,该指令取自内存,并由 CPU 进行译码. 在指令的执行过程中,PC 寄存器改变为要执行的下一条指令的地址. 程序中的控制流程(如 if、while 函数调用等)则通过将 PC 寄存器设置一个新地址来实现.

CPU 的复杂性部分是由各种指令的复杂性不同引起的. 某些指令(如按位逻辑或)容易用硬件实现,但某些指令(如浮点除)则难以实现,访存操作也很复杂,CPU 通常无法预测何时能完成一次访存操作. 幸运的是,人们已经提出了多种办法来解决这些问题,如在浮点操作中使用流水线技术. 该技术是将一个复杂操作分割成多个步骤,每个步骤可由 CPU 同时执行,只是 CPU 处理的数据是不同的. 也就是说,在一个时钟周期内,一旦启动了一个浮点操作,那么在下一个时钟周期内,即使该操作还没有结束,也可以开始进行下一个浮点操作. 图 1.1 描述了一个浮点加指令的流水线. 随着技术的发展,CPU 的速度在提高,流水线的深度也在加大(即有更多的步骤),流水线技术已广为采用. 在现代 CPU 中,不仅是浮点操作,而且许多其他的操作也使用了流水线技术.



(a) 流水线中的不同步骤

(b) 在7个时钟周期内，对4组数求和. 在3个时钟周期以后，每个周期返回一个结果，如果不采用流水线，则4组数求和需要16个时钟周期.

图 1.1 浮点加流水线

1.1.2 存储器

当计算机在运行一个程序时，程序和数据保存在各级存储器中. 存储器的设计需要考虑下面的一些问题：

(1) 存储器的大小. 对用户来说，计算机的存储容量是永远不够的. 由此提出了“虚存”的概念，以使程序可以使用足够大的内存.

(2) 访存延迟与分级. 由于存储器的访问时间与 CPU 的速度极不匹配，从而提出了多级存储结构，它是在计算机性能与成本之间进行折中的产物.

(3) 存储带宽. 存储器与 CPU 以及其他设备之间的传输速率与 CPU 的发展速度始终不匹配.

(4) 存储器保护装置. 许多计算机体系结构都有专门的硬件来保护存储器，目的是为了防止应用软件有意或无意地修改其他应用程序所使用的各级存储器.

上面四个问题中，第一个问题实际上是一个成本问题，存储带宽可以通过增加存储器的通路数以及使用交叉存取技术来解决. 但访存延迟是一个难于解决的问题，它取决于物理约束，难于降低. 另外，高延迟还降低了存储访问的有效带宽. 例如，假设某个存储器互联网在传输 32b 的数据块时，带宽为 1Gb/s，则传输 32b 的时间需要 32ns，如果访存延迟也是 32ns，则传输 32b 数据块总共需要 64ns，从而使有效带宽由 1Gb/s 降低到 500Mb/s. 通常在高延迟下，改善带宽的方法是增加每次传输的数据量，即以大数据量来降低延迟所占用总时间的比例. 当然，只有当所传输的数据均有用时，该办法才有效.

为使访存速度与 CPU 相匹配，在计算机体系结构中引入了容量更小、速度更

快的存储层，即 Cache，也称为高速缓冲存储器. Cache 一般分为三级，一级 Cache 嵌入在 CPU 芯片内，为寄存器提供数据，它的容量最小，一般为 16-128kb. 多数系统有二级或三级 Cache，它们的容量为 4-8Mb. 典型的 DRAM(D)存储器，通常容量为 256Mb-4Gb，约为 Cache 容量的 1000 倍.

层次化存储器需要考虑两个重要问题：(1)降低 Cache 的失效率；(2)Cache线大小的选择. 由于 Cache 的容量远小于主存的容量，因此不可能将一个进程所使用的数据全部驻留在一级或二级 Cache 中. 所以程序在执行过程中，必须由存储硬件来确定哪些存储单元上的数据要复制到 Cache 中. 如果 Cache 已经写满，但还需要复制其他存储单元上的数据，则必须从 Cache 中删除一些数据项(必要时还要写回主存). 若 CPU 请求一个数据，该数据不在 Cache 中，这时产生一次 Cache 失效事件，失效事件发生的概率称为 Cache 失效率. 存储系统设计的主要目标就是使 Cache 失效率达到极小. 同时，失效率还依赖于程序的行为和对应的算法. 从编程和算法设计的角度来看，为了降低失效率，需要在程序中开发“时间局部性”，即在一个短时间内重复使用同一个数据，也就是说，在数据被剔出 Cache 之前，充分使用该数据. 这就要求编程和算法设计人员认真考虑数据的使用方式.

关于 Cache 线大小的选择问题. 数据在 Cache 和主存中通常以 64b、128b 和 256b 为单位进行成组传输，其传输单位称为 Cache 线. 在一个时刻移动的是整个 Cache 线，从而允许主存更有效地提供一组数据. 如果程序访问第一个数据后紧接着访问相邻的数据，它将发现该数据已在 Cache 中，对于这样的程序，较大的 Cache 线能提高程序的性能. 如果程序中采用了非结构化的访存方式，那么数据读入 Cache 将占据其大部分时间，而实际上这些数据并不被使用. 对于这样的程序，较大的 Cache 线反而会降低程序的性能，这时应选择相对较小的 Cache 线.

除上面两个问题外，还有一些其他需要考虑的问题，如关联度(主存地址到 Cache 的映射)、替换机制(旧的数据如何丢弃，新的数据如何分配空间)以及 Cache 大小等. 同时，在考虑时间局部性时，还要考虑空间局部性. 所谓空间局部性是指存储器中的数据不是以单个对象(如整数、字符、浮点数等)进行加载，而是以较大的单位进行加载. 关于这部分内容的进一步介绍可参阅文献[1].

1.2　并行计算机的基本概念及其分类

本节主要介绍并行计算机的基本概念以及并行计算机的分类方法.

1.2.1　并行计算机的基本概念

什么是并行计算机？简单地说，并行计算机是指两台或两台以上的处理机，通过高速网络连接起来而成的并行计算机系统. 处理机间相互通信与协作，以高效、

快速地求解大型应用问题.

并行计算机的出现、发展以及广泛应用是基于人们在两方面的认识:第一,单处理机性能不能满足大规模科学与工程问题的计算需求,而并行计算机是实现高性能计算、解决挑战性计算问题的唯一途径;第二,同时性和并行性是物质世界的一种普遍属性,如对几十个常规应用软件的统计表明,90%左右的串行计算均可以并行化[1]. 实际上,针对某一具体应用问题,我们可以利用它们内部的并行性,设计并行算法,将其分解为相互独立、但彼此又有一定联系的若干子问题,分别将各子问题交给各台处理机进行处理,而所有处理机按所设计的并行算法相互通信与协调,共同完成对给定应用问题的求解.

这里需要说明的是,在以后各章内容的介绍中,我们将并行计算机、并行处理机或多处理机几个称呼混合使用.

1.2.2 并行计算机的分类方法

对并行计算机的分类可以帮助我们识别并行计算机的重要特征. 它有多种分类法,如:按指令流和数据流的 Flynn 分类法[2]、按并行度和流水的 Händler 分类法[3]、按指令流和执行流的 Kuck 分类法[4]等. 由于任何计算机均在数据上执行指令的运算,指令流告诉计算机每步做什么运算,并通过它影响数据流. 因此,下面我们仅介绍 1966 年 Flynn 提出的按指令流和数据流的不同组合的计算机分类方法,其中指令流是指机器所执行的指令序列,数据流则指指令流所调用的数据序列. Flynn 分类法将计算机系统分为 4 类:

(1) 单指令流单数据流(SISD-Single Instruction stream Single Data stream)计算机.

(2) 单指令流多数据流(SIMD-Single Instruction stream Multiple Data stream)计算机.

(3) 多指令流单数据流(MISD-Multiple Instruction stream Single Data stream)计算机.

(4) 多指令流多数据流(MIMD-Multiple Instruction stream Multiple Data stream)计算机.

在这四类计算机中,SISD 计算机就是传统的单处理机,也称串行计算机,1.1 节对其作了介绍. 因此,按照指令流和数据流的不同,可将并行计算机分为三类:

(1) 单指令流多数据流(SIMD)并行机:在同一时刻,各处理机(或处理器)执行相同的指令,处理不同的数据. 这类并行机的典型代表是 70 年代的向量机(如 Cray-1 等)和 80 年代初期的阵列机(如 CM-2 等). SIMD 类并行机对并行计算机的发展起到了重要的推动作用,但由于微处理器芯片技术的发展,单处理机性能非常强大,同时,90 年代后并行机均朝着 MIMD 方向发展,所以目前这类并行计算

机已退出历史舞台.

(2) 多指令流单数据流(MISD)并行机:直到目前为止,我们在国内外还没有见到该类计算机产品出现.

(3) 多指令流多数据流(MIMD)并行机:各处理机(或处理器)可同时执行不同的指令,处理不同的数据. 该类并行机按照存储方式的不同又可分为共享存储多处理机(如对称多处理机 SMP—symmetric multiprocesser)、分布存储多处理机(如大规模并行处理机 MPP—massively parallel processer、机群(cluster))和分布共享存储 DSM(distributed shared memory)多处理机三种类型.

并行计算机的其他分类法见相应的参考文献.

1.3　并行计算机体系结构

本节首先简要介绍现代并行计算机的几种体系结构模型及其特征,然后讨论并行计算机系统结构中的几个主要问题,包括互联网络、CPU 并行和存储并行,最后介绍并行计算机今后的发展趋势.

1.3.1　现代几种并行计算机的体系结构模型及其特征

本小节主要介绍共享存储对称多处理机 SMP、大规模并行处理机 MPP、机群 cluster 和分布共享存储多处理机 DSM 等几种类型机器的体系结构及其特征.

1. 共享存储对称多处理机 SMP

图 1.2 描述了共享存储对称多处理机 SMP 的体系结构模型图,其存储模块和处理器对称地分布在互联网络的两侧,将商用微处理器通过高速总线(或交叉开关)与存储器相连. SGI Power Challenge 系列并行机、DEC Alpha 8400 服务器、IBM RS6000 等均属于这种类型的机器.

SMP 体系结构具有如下特征:

(1) 对称性. 系统中任何处理器均可访问任何存储模块中的任何存储单元和 I/O 设备,且访问成功的概率是一致的.

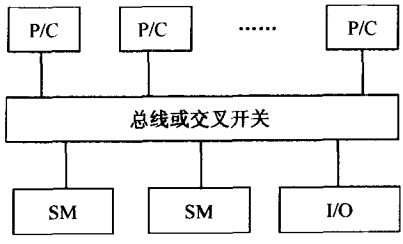


图 1.2　SMP 体系结构示意图

(2) 单一的操作系统(OS)映像. 系统中只有一个 OS 驻留在共享存储器中,它可以根据各处理器负载情况动态地在多个处理器上调度进程,从而可保证各处理器间的负载平衡.

(3) 局部高速缓存及其数据一致性. 每个处理器具有片上或外置高速缓存 Cache,多级