第38卷第7期 中国科学技术大学学报 Vol. 38, No. 7
2008年7月 JOURNAL OF UNIVERSITY OF SCIENCE AND TECHNOLOGY OF CHINA Jul. 2008

文章编号:0253-2778(2008)07-0841-07

分层并行计算模型

陈国良[1,2],苗乾坤[1,2],孙广中[1,2],徐 云[1,2],郑启龙[1,2]

(1. 中国科学技术大学计算机科学与技术系,安徽合肥 230027;2. 安徽省高性能计算重点实验室,安徽合肥 230027)

摘要:传统单一的并行计算模型变得越来越复杂,对并行计算各阶段针对性不强、指导能力差的特点,为此提出了对并行计算模型分层研究的思想,依此把并行计算模型分为并行算法设计模型、并行程序设计模型、并行程序执行模型三个层次,分别给出了各个模型的特点及研究内容. 理论分析结果表明,通过分层,每个阶段的模型分工明确,目标单一,指导性强.

关键词:分层并行计算模型;并行算法设计模型;并行程序设计模型;并行程序执行模型

中图分类号:TP302,TP338.6 **文献标识码**:A

Layered models of parallel computation

CHEN Guo-liang[1,2], MIAO Qian-kun[1,2], SUN Guang-zhong[1,2], XU Yun[1,2], ZHENG Qi-long[1,2]

(1. *Department of Computer Science and Technology, University of Science & Technology of China, Hefei* 230027, *China*; 2. *Anhui Province Key Laboratory of Computing and Communication Software, Hefei* 230027, *China*)

Abstract: The conventional unified parallel computation model becomes more and more complex, which has weak pertinence and little guidance for each parallel computing stage. A layered idea for parallel computation model research was proposed, in which the layered parallel computation model was composed of parallel algorithm design model, parallel programming model and parallel execution model. The properties of each model were described, and research spots were also given. Theoretical analysis results show that the layered parallel computation model is clear and single-goaled for each parallel computing stage.

Key words: parallel computation layered model; parallel algorithm design model; parallel programming model; parallel execution model

0 引言

当今几乎所有学科都在向定量化和精细化的方向发展,为此必须使用计算,比如:计算生物,计算物理,计算化学等. 通常这些计算都是非常耗时的,单个PC的计算能力远远不能满足人们的需求,提高计算速度的有效方法是采用并行计算[1]. 并行计算现在已经广泛应用于气象、能源、生物、制药、航天、抗灾、国防、制造、科研等关系到国计民生等重要领域[2~4],成为一个国家科技水平的标志. 近年来,国内外并行计算机的发展不断地迈向新台阶,采用国产高性能通用处理器芯片龙芯2F的国产万亿次高性能计算机KD-50-I[5]的研制成功是我国并行计算机国产化的重大突破. 不久的将来,并行计算机将会广泛出现在人们的日常生活中.

并行计算就是在并行/分布式计算机上所做的计算. 并行计算的过程可分为如下几个步骤[1,6]:对于一个给定的问题,计算科学家首先要将其描述为

收稿日期:2008-05-22;**修回日期**:2008-06-23

基金项目:国家自然科学基金重点项目(60533020)资助.

作者简介:陈国良(通讯作者),中国科学院院士. E-mail: glchen@ustc.edu.cn

待求解的数值或非数值计算问题;然后要为该计算问题设计一个并行算法,并选择某种并行程序设计语言编程来实现它;最后领域专家/程序使用者要在具体的并行机上编译和运行程序(软件)最终求解出此问题.从以上并行计算的一般过程可以看出,并行计算以并行算法为理论基础,以并行计算机为硬件平台,以并行程序设计为软件支撑.

计算模型是为了计算目的将真实体(计算机软/硬件)进行一定的抽象而成的[1,7].并行计算模型是算法设计者所看到的参数化了的并行机,是提供给编程者的计算机软/硬件接口,是程序执行时的系统软/硬件支撑环境.并行计算模型是算法设计者、程序设计者和系统运行者在实现并行计算时所看到的虚拟并行计算机,是他们共同执行任务的桥梁.传统的并行计算模型一般指的是并行算法设计模型,为并行算法的研究者提供一个独立于具体并行机体系结构的抽象的并行机.一般模型中通常定义了机器参数、计算行为、开销函数,它们被称为计算模型三要素.随着并行计算机体系结构的飞速发展,为了使计算模型能够反映体系结构的变化,人们不断地向该单一模型中加入旨在反映机器特性的新参数,调整计算行为,修改开销函数.因此,单一模型变得越来越复杂,在实际的算法设计中难以使用.

根据并行计算的一般过程,我们提出将并行计算模型的下一步研究分为并行算法设计、并行程序设计、并行程序执行三个层次来考虑.通过分层,在并行计算的各个阶段,模型的职能各有侧重,分工明确,目标单一,从而适合不同阶段的设计人员关注主要的问题.按照分层的观点,并行计算模型可定义为如下的三元组

并行计算模型 ={并行算法设计模型,
并行程序设计模型,并行程序执行模型}

并行算法设计模型,从算法的角度,将不同的并行机抽象为一种通用虚拟并行机,算法设计者在其上设计和分析并行算法;并行程序设计模型,从编程的角度指导程序员,按照程序的执行流程,选用某种并行语言,正确编程实现某并行算法;并行程序执行模型,从性能效率的角度指导程序运行者,将不同的并行语言实现的程序,在具体的并行机上编译和优化运行并行程序.

1 研究背景

1.1 并行计算模型相关工作

传统的并行计算模型,主要是针对算法研究者,比较抽象、理论性强,不实用,只关心并行算法的设计与分析,不考虑算法的具体实现和执行.文献[8]介绍了并行计算模型的发展历史,总结了各阶段中一些典型的并行计算模型的特点.并行计算模型的演变历史,首先是模型以 CPU 计算为中心,再以网络通信为中心,最后逐步过渡到以存储访问为中心.

根据并行计算模型的发展历史,可以把并行计算模型分为三代[8].早期的并行机为共享存储器的 SIMD 和 MIMD 的计算机,相应地,提出了以计算为核心的所谓第一代并行计算模型 PRAM[9,10] 和 APRAM[11] 等.在此期间,还出现了不少各种不同互连结构的 SIMD 机器,相应地也提出了 SIMD-IN 模型,并在其上开发了不少优秀的并行算法.后来出现了分布存储的大规模并行机 MPP,相应地提出了以网络通信为核心的所谓第二代并行计算机模型 BSP[12],LogP[13],NHBL[14] 等.最近分布共享的并行机成为主流并行机,CPU 和主存之间的速度差异越来越大,存储系统逐渐成为影响系统性能的主要瓶颈,为此提出了以存储访问为核心的所谓第三代并行计算机模型 UMH[15],DRAM(h)[16] 等.另外,在第三代并行计算机模型的基础上,又提出了同时考虑层次存储和层次并行的所谓第三代半并行计算机模型 HPM[17] 等.

从并行计算模型的发展演变过程可以看出,并行计算模型沿着丰富、强化单一模型的路线发展.在单指令流共享存储模型 PRAM 的基础上,扩展为多指令流共享存储模型 APRAM 等.在多指令流共享存储模型的基础上,扩展为多指令流分布存储模型 BSP,LogP 等.在多指令流分布存储模型的基础上,扩展为考虑不同层次存储模型 UMH,DRAM(h) 等.在层次存储模型基础上,又进一步扩展为考虑层次并行模型 HPM 等.

1.2 模型分层的基本出发点

利用并行计算所能获得的性能综合体现在算法、实现、编译器、操作系统、处理器体系结构、网络互连技术等因素,因此实际的计算行为相当复杂.一味追求单一模型的功能强和多目标,致使单一模型越来越复杂,最终导致模型不实用和不可操作性.单一模型中,反映机器不同特性的参数过多,会使单一模型描述越来越困难,最终导致成本函数过于复杂而无法求解.当在单一模型中又考虑到机器的底层特性和不同的硬件并行度时,使得模型很难建立.对算法设计者而言,单一模型太复杂,设计算法时考虑的因素过多,

影响设计低时空开销的优秀算法. 所以,并行计算模型应按算法的设计、算法的实现和算法的执行进行分层,即模型可分为: 并行算法设计模型,并行程序设计模型和并行程序执行模型,前者面向并行算法研究者,中者面向并行程序设计者,后者面向程序运行者.

并行算法设计模型是算法设计者和计算机体系结构之间的桥梁,主要用来指导并行算法的设计与分析. 并行程序设计模型是程序设计者与计算机软/硬件的接口,主要负责把算法转换成可在并行机上正确执行的程序. 并行程序执行模型是编程人员在并行机软/硬件支撑下,编译和运行机器目标代码,以优化程序的性能. 分层计算模型可将单一模型中的功能按要求分配到模型不同的层次中,缓解了单一计算模型的精确性与可使用性之间的矛盾. 分层后,各层次模型职能不同,目标单一,各负其责,易于设计与实现. 分层并行计算模型中的并行算法设计模型和并行程序设计模型目前相对比较成熟,从而可集中精力重点研究并行程序执行模型. 三层并行计算模型功能如图 1 所示. 三层并行计算模型从几何形状上看,呈现哑铃形状:从不同的并行计算机来(抽象计算参数建立模型),经过统一的加工后(用语言编程实现算法),又回到不同的并行计算机中去(运行代码,求解问题). 文献[18]从编程者的角度把并行程序的编写过程分为若干层次,本文将并行计算的整个过程分为三个层次,而并行编程是我们提出的三层模型中的一个层次.

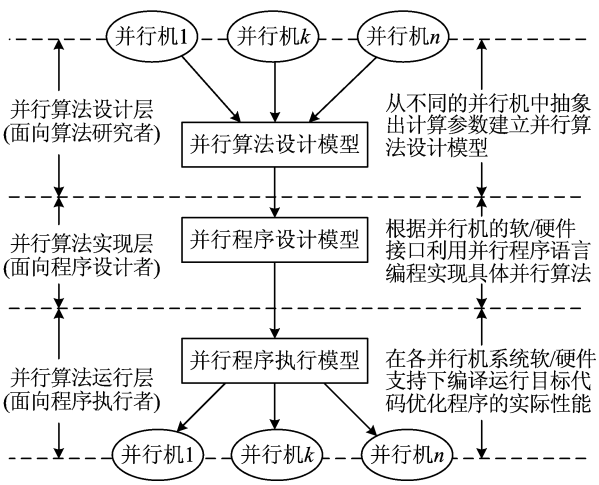

图 1　并行计算的分层模型示意图

Fig. 1　Overview of layered models for parallel computing

2　分层的并行计算模型

从给定的计算问题开始,首先用伪代码设计并行算法,然后用源语言算法的时空复杂度实现算法,最后编译成机器代码在实际并行机上运行,整个并行计算过程表现出来的性能取决于一系列的因素. 算法的伪代码数和通信次数决定算法的时空复杂度和通信复杂度;程序设计语言、编译器、运行环境的选择决定每条源级语句转换成机器指令的数量和优化的程度;操作系统、处理器、存储系统、体系结构决定每条机器指令实际执行的速度;I/O 系统决定整个程序 I/O 操作的数量. 为了在实际并行机上获得高的编程和运行效率,并行计算的各个阶段都要在各自模型的限制和约束下进行,即并行算法设计模型、并行程序设计模型、并行程序执行模型. 下面我们分别给出每个模型的定义、参数、行为描述以及当前已有的相关模型.

2.1　并行算法设计模型

算法设计模型需要反映机器硬件关键特性,同时又必须非常简单易于算法描述和表达,是算法设计者与机器结构之间的桥梁,面向算法的研究者. 算法设计模型重点关注算法的设计原理,确保设计出来的算法的正确性和较低的时间、空间复杂度. 算法设计模型主要体现在如下三要素:机器参数(抽象出的 CPU,Memory,I/O 网络参数等)、执行行为(算法的同步,异步执行等)、成本函数(算法的复杂度函数,它是机器参数的函数),其功能特性如图 2 所示. 长期以来,计算机科学家研究并行算法的设计模型,提出了各种反映当时并行机体系结构的模型[8].

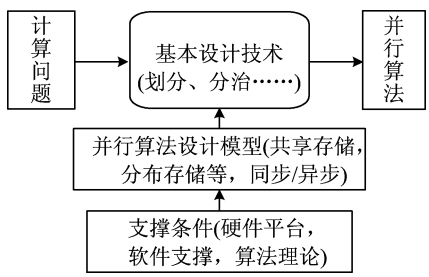

图 2　并行算法设计模型示意图

Fig. 2　Overview of parallel algorithm design model

并行算法设计模型理论性较强,计算机科学家们为此做出了诸多的贡献. 随着并行机的发展,不少计算机结构科学家,参与了所谓第二代并行计算模型的研究. 第三代并行计算模型,涉及并行机底层的很多硬件知识,所以参与研究该模型的科学家越来越多. 并行算法设计模型的出现,首先使得并行算法的研究不再仅限于针对某台具体的并行计算机专门设计和分析并行算法,更重要的是,可在一类抽象的

并行机上研究算法,从而使得并行算法本身更趋普适化,可广泛适用于某一类并行机;其次,在并行算法设计模型上设计出的并行算法,可以利用数学工具严格定量地分析其性能,这样使得并行算法学科本身更加定量化,更趋成熟;最后,因为有了统一的并行算法设计模型,这样在其上设计出的很多不同的并行算法,可以根据算法的复杂度统一标准对其优劣进行相互比较. 按照我们的研究思路,并行算法设计模型是我们提出的三层计算模型中最基础的部分,其他两层模型(并行程序设计模型和并行程序执行模型)都尽量参照该模型进行研究.

2.2 并行程序设计模型

并行算法设计出来后,我们需要考虑如何正确、方便、快速地在某种并行程序设计模型下用编程语言实现[19]. 并行程序设计模型也可以称为并行程序编程模型,是提供给程序员使用的,为程序员提供了一些计算机软/硬件的编程接口,隐藏了通信和任务调度的细节,在一定程度上简化了并行程序的编写. 程序设计模型主要确保并行算法使用某种程序设计语言可以正确地在并行计算机上编程实现. 并行程序设计模型应该尽量缩减与传统串行编程模型的差异,让串行程序设计人员可以快速地掌握并行程序编写的方法,降低大规模并行程序设计的难度. 一个成功的并行程序设计模型需要具有可编程性、可移植性、可扩展性、通用泛化性. 通过提高并行程序设计模型和相应并行程序设计系统的抽象层次,屏蔽并行系统中的底层实现细节,支持通用的并行数据结构和并行程序的开发,最终提高程序的开发效率. 程序设计模型需要有自己的高级语言或者对当前语言进行扩展,用语言对程序员进行约束,使他们在该语言下实现算法,其功能特性如图 3 所示. 当今流行的并行程序设计模型主要有大粒度的进程级的消息传递模型,如 MPI[20,21];中、细粒度的线程级共享变量模型,如 OpenMP[22],Pthread[23];细粒度进程级的数据并行模型,如 HPF[24].

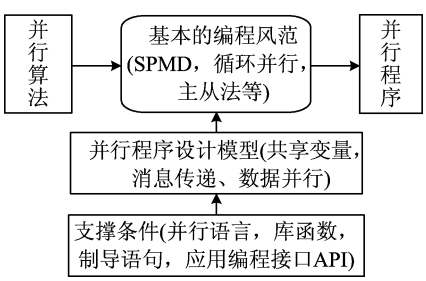

图 3 并行程序设计模型示意图

Fig. 3 Overview of parallel programming model

并行计算机体系结构的快速发展[2~4],多核桌面计算机的普及[25~27]及更大规模并行计算机的设计与研制,给并行程序设计模型的研究带来新的机遇,也提出了挑战. 要充分利用这些并行计算资源,就需要设计更加易于使用的并行编程模型,以满足普通程序员开发并行程序的需求. 由于并行程序要比串行程序复杂的多,加之并行执行的不确定性、易出错性,并行程序设计一般由并行领域的专业人员来完成. 多个线程同时运行时,通常有对共享数据的访问,产生数据竞争问题,最终的执行结果跟多个线程对共享数据区读写的先后顺序有关. 并行程序执行时,每次执行的指令流的顺序有差别,执行中遇到的错误往往很难重现,这就导致了对并行程序执行的分析、调试、优化非常困难. 一般来说,不同的并行算法会有一些共同的结构,可以抽取出这些结构,把它们放到编程模型中. 这样程序员编程时可以集中在应用问题本身的实现上,考虑如何把算法映射到适合的编程模型,其他并行处理相关的细节,诸如通信如何完成、任务怎么映射到各个线程、如何进行同步等,在编程时不需要显式考虑;利用编程模型提供的一些接口,并行相关的细节都交给运行环境处理. 一些常见的并行编程风范有 fork/join, pipeline, work-farms, meshes 等模型. 这些模型可以解决很大范围的问题,并且它们都是经过仔细检验和多次验证的、保证可以开发出来的正确的并行程序. 模型中定义了程序的控制流,所有并行相关的部分都放在编程模型中解决,应用相关的一些代码就可以直接使用已有的串行程序,程序员只需要把这些应用相关的代码填入到控制流的各个阶段,一个并行程序就可以快速地设计出来了. 研究者近年来基于并行编程模型的思想开发了许多简化并行程序设计的支持系统,如 MapReduce[28],Dryad[29],Hadoop[30],CO_2P_3S[31]. 这些系统在保证程序正确性的前提下,都尽量使程序员从高层入手,不要求有并行编程的经验,就可以很容易地编写出高效的可并行执行的程序. 目前这些系统支持的应用还有一定的局限性,只有能和系统提供的编程模型匹配的应用才可以很好地在以上系统上实现,主要应用在诸如信息检索、数据挖掘等领域. 随着研究的深入,它们支持的编程模型也将越来越多.

并行编程语言是并行程序设计模型的一个重要组成部分. 随着多核核技术的迅猛发展,学术界和工

业界对高生产率并行编程语言需求迫切,对新一代并行编程语言的研究正日渐升温. Stanford 大学开发的 Sequoia[32]面向 Cell 多核处理器和机群系统,提供对存储层次进行程序设计的方法,通过把存储层次暴露给程序员,允许他们显式地对数据在各级存储中进行分配,在保证高可移植性的前提下,取得了很好的性能和效率. 美国 HPCS[33]项目资助三种新型并行编程语言,IBM 的 X10[34]、Sun 的 Fortress[35]以及 Cray 的 Chapel[36]. 它们都属于研究型语言,从降低并行编程难度出发,力争提高并行软件的生产率,同时还要提供高性能、便于移植和健壮性的支持. 目前只在小范围内使用,还没有对当前并行编程和并行软件的开发产生革命性的影响. 随着时间的推移,CPU 和存储器性能差异会日益加大,存储层次也会变得更加复杂,这些新出现的编程语言与以往最大的不同是提供了对存储层次的显式控制,允许程序员在编程语言中对存储局部性进行描述,明确指出数据所在的位置,试图解决目前存储系统的性能瓶颈问题.

2.3　并行程序执行模型

并行程序的执行过程包括:将高级语言编写的程序编译成机器语言,运行时的资源分配和线程调度,数据在各级存储系统中的移动,每条指令在实际机器上的执行过程以及指令流水线的动态执行. 这些因素在使用算法设计模型进行算法设计和使用程序设计模型编写程序时,不需要考虑或者有些时候没法考虑. 由于在设计算法和编写程序时并不清楚最终程序会在什么样的机器上执行,而且为一台机器编写的代码往往需要移植到其他体系结构特性相差很大的机器上,所以很难取得最好的性能.

并行程序编写好之后,还需要花大量的时间对它进行优化才能取得满意的性能,因此需要使用并行执行模型来指导如何在实际的机器上进行性能优化. 并行程序执行模型将具体的并行计算机抽象成若干个影响程序执行性能的因素,从而可以使程序运行者针对具体的机器环境,进行程序的优化工作. 并行执行模型使用执行时间、加速和效率、可扩展性等标准来衡量程序性能. 并行程序执行模型通过对程序执行环境进行抽象,得到一个面向程序执行者的简洁的统一模型. 执行模型定义了低层次的系统结构的编程抽象,表示程序在一个并行计算机上真实的执行行为. 程序执行模型是编译器设计人员与系统实现人员之间的接口,编译器设计人员决定如何将一种高级语言程序按某种程序执行模型转换成一种机器代码;系统实现人员则决定该程序执行模型在具体目标机器上的有效实现. 程序执行模型的适用性决定并行计算机是否以最低的代价提供最高的性能. 根据程序执行模型进行调整的并行程序,可以在一台具体的并行计算机上获得最优的性能. 并行应用程序去匹配这个模型,匹配的程度越高应用程序可以取得的性能就越好.

影响并行程序执行性能的因素包括硬件和软件的诸多方面,如缓存共享、处理器连接方式、通信的次数和每次通信的数据量、同步开销,多个处理器之间的负载平衡等,使得并行程序的性能优化更加复杂,这些都是串行程序不需要考虑的.

并行程序执行模型中,需要考虑如下机器硬件因素对程序执行性能的影响:(Ⅰ)中央处理器性能参数:时钟频率,功能单元执行速度,寄存器数量,片内缓存大小,指令流水线,指令多发射,指令和数据预取等;(Ⅱ)存储系统性能参数:存储系统的层次,每级存储系统的容量,块大小,延迟,带宽等;(Ⅲ)互连网络性能参数:存储一致性协议,网络的延迟,带宽等;(Ⅳ)输入/输出系统性能参数:磁盘的容量,速度以及其他的 I/O 设备的性能等.

并行执行模型中,需要考虑如下应用程序和运行时软件环境相关的问题:(Ⅰ)并行性级别问题:例如,任务级并行(考虑任务划分、映射、调度),数据级并行(考虑数据划分,迭代和循环分解),指令级并行(考虑如何高效发挥向量流水线,多发射,执行预取的效率等);(Ⅱ)线程问题:如何来进行线程调度,如何分配任务到细粒度的线程,尽量使得负载平衡,线程间如何通讯以及使得线程间通讯代价更小;(Ⅲ)存储和通信问题:对共享地址空间的支持,存储一致性的支持,通过分块、数据重新组织调整提高程序的空间局部性和时间局部性,通信协议的选择,通信软件的额外开销等;(Ⅳ)同步问题:锁和临界区的代价;(Ⅴ)性能剖析问题:相应工具的在线剖析,垃圾收集,动态编译,自适应多版本执行,实现反馈式的性能优化,提供一些简单的自动优化.

线程问题是并行程序相对串行程序所独有的,用线程来表示并行执行的任务,定义了各线程间数据同步和数据通信的方式. 在并行程序设计模型中定义了一些线程的实现接口,在程序实际运行时并行程序执行模型通过运行时环境和硬件执行部件来具体负责创建线程、维护线程、销毁线程以支持这些接口. 近年来有研究者提出使用投机多线程[37]

表 1 分层模型的对照比较

Tab. 1 Comparisons of layered models

比较对象	并行算法设计模型	并行程序编程模型	并行程序执行模型
面向对象	算法设计者	编程者	程序运行者
作用	算法设计者和机器结构设计者之间桥梁	程序设计者与计算机软/硬之间接口	编译设计者与系统实现者之间接口
关注点	算法正确性低时、空开销	确保算法正确语义,正确编程实现	优化程序执行性能
要素	机器计算参数 计算行为 计算复杂度函数	编程模式 编程基本单元 易用性,可编程性	机器实际性能因素 运行时系统的行为 性能指标
方法学	设计方法(划分,分治,流水线,平衡树)	编程风范(SPMD,循环并行,主从法 MPMD,Fork/Join,放牧法,流水线法)	执行模式(线程/进程产生,管理与撤消;同步;通信)
复杂度	算法步数	高级语句条数	机器指令条数
支撑条件	算法理论 硬件平台 软件支撑	并行语言 工具环境 应用编程接口 API	编译器 OS 运行时系统硬件结构(CPU,Memory,I/O)
现有模型	PRAM,APRAM,BSP,LogP,DRAM(h),NHBL	MPI,OpenMP,Pthread,HPF	有待进一步的工作

(speculative multithreading)和事务内存[38](transaction memory)来处理多线程并发的行为,解决多线程程序的相关性及同步问题,把同步控制的压力由程序开发人员转换到了编译运行时的系统设计人员.使用这两种执行模型时,程序员使用的编程模型就是传统的串行编程模型,编译器和运行系统进行并行任务的划分及执行.采用这种方式的并行应用程序没法用并行算法设计模型和用并行编程模型来分析,因为直到执行前它们的设计都与串行程序无异;也没法用串行程序的分析方法来分析,因为实际执行的时候它是并行执行的.因此,我们需要在并行程序执行模型中考虑这种应用的行为,对应用的执行性能进行分析.图 4 给出了并行执行模型相关的一些组成部分以及功能特性等.

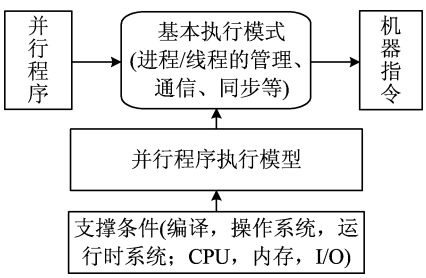

图 4 并行执行模型图示

Fig. 4 Overview of parallel execution model

3 结论

根据并行计算模型的发展和演变的趋势,对并行计算模型进行分层是势在必行的.按照并行计算模型的功能和使用对象,我们认为把并行计算模型分为三个层次(并行算法设计模型,并行程序设计模型和并行程序执行模型)是充分和必要的,每层模型的特点如表 1 所示.并行算法设计模型和并行程序设计模型,目前相对比较成熟,而并行程序执行模型尚处于酝酿阶段,有待进一步研究与开发.我们认为,并行执行模型关注点在并行程序的性能,相应的研究应该从并行程序的性能优化入手,而性能优化应首先从编译系统、操作系统和运行系统等方面入手,兼顾考虑处理器、存储器和输入/输出设备等因素.

参考文献(References)

[1] 陈国良. 并行计算——结构·算法·编程[M]. 2 版,北京:高等教育出版社,2003.

[2] Zhang Y Q, Sun J C, Yuan G X, et al. A brief introduction to China HPC top100: from 2002 to 2006 [C]//Proceedings of the 2007 Asian Technology Information Program's 3rd Workshop on High Performance Computing in China: Solution Approaches to Impediments for High Performance Computing. Reno, USA: ACM Press, 2007: 32-36.

[3] Sun J C, Yuan G X, Zhang L B, et al. 2007 China top100 list of high performance computer[EB/OL]. http://www.samss.org.cn/2007-China-HPC-TOP100-20071110-eng.htm

[4] TOP500 Supercomputing Site[EB/OL]. http://www.top500.org/.

[5] 张俊霞,张焕杰,李会民. 基于龙芯 2F 的国产万亿次高性能计算机 KD-50-I 的研制[J]. 中国科学技术大学学报,2008,38(1):105-108.

[6] Chen G L, Sun G Z, Zhang Y Q, et al. Study on

parallel computing[J]. Journal of Computer Science and Technology, Special Issue Dedicated to the 20th Anniversary of NSFC, 2006, 21(5): 665-673.
[7] 陈国良. 并行算法设计与分析[M]. 北京:高等教育出版社, 2002.
[8] Zhang Y Q, Chen G L, Sun G Z, et al. Models of parallel computation: a survey and classification[J]. Frontiers of Computer Science in China, 2007, 1(2): 156-165.
[9] Fortune S, Wyllie J. Parallelism in random access machines [C]//Proceedings of the 10th Annual Symposium on Theory of Computing. San Diego, CA: ACM Press, 1978: 114-118.
[10] Goldschlager L M. A universal interconnection pattern for parallel computers[J]. Journal of the ACM, 1982, 29(4): 1 073-1 086.
[11] Cole R, Zajicek O. APRAM: incorporating asynchrony into the PRAM model[C]// Proceedings of the First Annual ACM Symposium on Parallel Algorithms and Architectures. Santa Fe, New Mexico: ACM Press, 1989: 158-168.
[12] Valiant L G. A bridging model for parallel computation [J]. Communications of the ACM, 1990, 33(8): 103-111.
[13] Culler D, Karp R, Patterson D, et al. LogP: towards a realistic model of parallel computation [C]// Proceedings of the 4th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming. San Diego, California, USA: ACM Press, 1993: 1-12.
[14] 计永昶, 丁卫群, 陈国良,等, 一种实用的并行计算模型[J]. 计算机学报, 2001, 24(4): 437-441.
[15] Algern B, Carter L. The uniform memory hierarchy model of computation[J]. Algorithmica, 1994, 12(2/3): 72-109.
[16] 张云泉. 面向高性能数值计算的并行计算模型 DRAM(h)[J]. 计算机学报,2003,26(12): 1 660-1 670.
[17] Qiao X Z, Chen S Q, Yang L T. HPM: a hierarchical model for parallel computations [J]. International Journal of High Performance Computing and Networking, 2004, 1(1/2/3): 117-127.
[18] Mattson T G, Sanders B A, Massingill B L. Patterns for Parallel Programming[M]. USA: Addison-Wesley Professional, 2004.
[19] 陈国良, 安虹,陈峻,等. 并行算法实践[M]. 北京: 高等教育出版社, 2003.
[20] MPICH site[EB/OL]. http://www-nuix. mcs. anl. gov/mpi/mpich/.
[21] The MPI Forum. The MPI message-passing interface standard[EB/OL]. http://www. mpi-forum. org/.
[22] OpenMP standards board. OpenMP: a proposed industry standard api for shared memory programming [EB/OL]. http://www. openmp. org/openmp/mp-documents/papaer/paper. html.
[23] Pthread interface. ANSI/IEEE Standard 1003. 1 [S]. 1996.
[24] The high performance FORTRAN forum[EB/OL]. http://hpff. rice. edu/.
[25] Intel multi-core site[EB/OL]. http://www. intel. com/multi-core/index. htm.
[26] AMD multi-core technology site[EB/OL]. http://multicore. amd. com.
[27] IBM cell processor site[EB/OL]. http://www-01. ibm. com/chips/techlib/techlib. nsf/products/Cell _ Broadband_Engine.
[28] Dean J, Ghemawat S. MapReduce: Simplified data processing on large clusters[C]// Proceedings of the 6th Symposium on Operating System Design and Implementation. San Francisco, USA: USENIX Association, 2004: 137-150.
[29] Isard M, Budiu M, Yu Y, et al. Dryad: distributed data-parallel programs from sequential building blocks [J]. ACM SIGOPS Operating Systems Review, 2007, 41(3): 59-72.
[30] Hadoop project site[EB/OL]. http://hadoop. apache. org/core/.
[31] MacDonald S. From patterns to frameworks to parallel programs[D]. PhD dissertation, Edmonton, Alberta, Canada: Department of Computing Science, University of Alberta, 2002.
[32] Fatahalian K, Knight T J, Houston M, et al. Sequoia: programming the memory hierarchy[C]// Proceedings of the 2006 ACM/IEEE Conference on Supercomputing. New York, USA: ACM Press, 2006:83.
[33] High productivity computing systems [EB/OL]. http://www. highproductivity. org/.
[34] X10 programming language[EB/OL]. http://www. research. ibm. com/x10/.
[35] Fortress project[EB/OL]. http://research. sun. com/projects/plrg/.
[36] Chapel project [EB/OL]. http://chapel. cs. washington. edu/.
[37] Krishnan V, Torrellas J. A chip multiprocessor architecture with speculative multithreading[J]. IEEE Transactions on Computers, Special Issue on Multithreaded Architectures. 1999, 48(9):866-880.
[38] Herlihy M, Moss J. Transactional memory: architectural support for lock-free data structures[C]// Proceedings of the 20th International Symposium in Computer Architecture. New York, USA: ACM Press, 1993: 289-300.