并行程序设计模型和语言*

安 虹, 陈国良

(中国科学技术大学 计算机科学技术系,安徽 合肥 230027);
(国家高性能计算中心(合肥),安徽 合肥 230027)
E-mail: han@ustc.edu.cn
http://www.nhpcc.ustc.edu.cn

摘要: 并行计算技术的发展已有20多年的历史了.时至今日,高性能并行计算仍然缺乏有效的并行程序设计方法和工具,使得编写并行程序、理解并行程序的行为、调试和优化并行程序的性能都很困难.从分析并行程序设计困难的原因入手,指出了当前各种高性能并行机系统支持的并行程序设计方法存在的诸多问题,综述了并行程序设计模型和语言的研究现状,给出了并行程序设计模型的评价标准,并提出了这一研究领域所面临的挑战性问题,指出了一些未来可能的发展方向.

关 键 词: 并行程序设计;模型;语言;可移植性;可扩放性;局部性
中图法分类号: TP312 文献标识码: A

并行计算技术的发展已有 20 多年的历史了,高性能并行计算机系统正在进入越来越多的应用领域.与硬件的发展相比,并行软件的发展则显得更加滞后,影响了硬件效率的发挥,限制了并行机系统的广泛应用.并行程序设计理论和技术的研究已成为发展并行处理技术当务之急的课题.并行程序设计方法经历了多年的发展和实践检验,进一步的发展面临着诸多挑战性问题,现在应该是很好地进行总结的时候了.

1 并行程序设计困难的原因

时至今日,高性能并行计算机上至今仍然缺乏有效的并行程序设计方法和工具,使得编写并行程序、理解并行程序的行为、调试和优化并行程序的性能都很困难.并行程序设计困难的深层原因在于,高性能并行计算技术在发展过程中存在着以下问题:

(1) 迄今为止,并行计算所需的理论仍然是不成熟的,没有理论或很少有理论为技术的发展指明方向.我们还不知道更多的表示并行计算的方法,也不知道逻辑上如何推理它们,甚至不知道在实际的体系结构上,什么样的并行算法才是有效的.

(2) 人们花了很长时间才理解并行计算机的不同部分的性能需要平衡以及这个平衡是如何影响性能的.要获得好的性能,则需要仔细控制处理器速度和互连通信性能的关系,同时也需要与存储器层次的性能取得平衡.许多并行系统无法很好地支持实现这种平衡,甚至把这个艰难的工作交给用户去做.

(3) 人们已经习惯于以串行的方式来理解、设计和调试程序,因而感到并行程序设计有些困难.从另一方面来说,与串行程序设计相比,并行程序设计更具挑战性,这有 3 个方面的原因:第一,并行程序必须包含在处理器间交换数据和处理互斥区的机制,这就增加了程序的语义复杂性和语法复杂性;第二,对一个有效的并行程序来说,必须在处理器之间均匀地分配任务,这种对并行程序设计在算法方面的挑战是串行程序设计中所没有的;第

* 收稿日期: 2000-04-20; 修改日期: 2000-07-12
基金项目: 国家 863 高科技发展计划资助项目(863-306-ZD01-02-3);中国科学技术大学青年科学基金(KA1109)
作者简介: 安虹(1963－),女,山东胶州人,博士,副教授,主要研究领域为并行计算机体系结构,并行程序设计环境与工具,高性能计算;陈国良(1938－),男,安徽颍上人,教授,博士生导师,主要研究领域为并行分布计算,并行计算机体系结构,并行算法.

三,数据也必须在处理器间进行划分,以保持数据的局部性,因为移动数据是要开销的.

(4) 与串行程序设计相比,并行程序设计中存在的可移植性问题更加严重.由于至今还没有哪个并行体系结构家族是独立于技术变化的,当需要将一个软件系统从一个并行平台移植到另一个上时,则可能意味着要完全重建这个软件.许多用户至今还未能从高性能并行计算中得到实际的好处,就是因为重写代码的难度太大.并行软件用户期望在他们所使用的并行计算平台上能获得平滑的过渡,这样可以平滑地重新设计和重建软件.

(5) 长期以来,并行系统的设计目标只面向市场较小的科学和工程计算,而忽视了大的工业和商用市场,使得并行机十分昂贵,增加了制造商和用户的投资风险,减小了用户使用并行机的热情,从而势必造成其上的软件技术发展缓慢.并行程序设计环境和工具从来都缺少较长的生长期,无法通过不断的版本更新,形成与现今的PC 机或工作站一样好用的环境和工具.

2 研究现状

并行程序设计模型是硬件和软件之间的桥梁,是并行计算的低层实现与高层抽象的界面.为了帮助程序员解决并行程序设计面临的挑战性问题,并行程序设计模型和语言的研究者们已经研究了许多种不同的模型和语言,但迄今为止,还没有哪一种模型是通用的和完美的.

2.1 模型研究的硬件结构抽象观点

尽管模型的建立应独立于具体的硬件结构,但现有的大多数并行程序设计模型和语言的产生与发展还是深受并行硬件系统结构发展的影响,尤其在并行系统发展的早期.

在 20 世纪 80 年代后期有两个相互竞争的硬件方法:SIMD(single instruction multiple data)和 MIMD (multiple instruction multiple data),最后是 MIMD 占了上风.硬件也可区分为共享存储和分布存储机器.近年来,共享存储和分布存储的概念本质上已趋于合一.现今大规模并行一般使用多个计算节点和分布存储来实现,每个计算节点内部采用共享存储.一些商家试图在分布存储的机器上提供一个单一的存储映像,使并行机看上去更像一台大的工作站.目前,分布共享存储程序设计只是在处理器数较少的情况下获得了成功,随着处理器数量的增加很少能获得成功.其原因主要不在硬件方面,而在于编译器不能识别和利用数据的局部性.例如,Cray T3E 的通信模式已经做得开销很低了,但显式的消息传递依然是使用多个处理器时最有效的方式.因此,我们认为分布共享存储程序设计可能并非并行计算的圣杯,尽管它是当前许多研究者所追寻的目标.究竟采用哪种方法才是正确的,可能取决于可得到的硬件以及想要得到什么量级的并行.

模型和语言代表了用户与并行体系结构之间的中介,应该保障有更多的应用领域能简单而有效地使用并行计算.从体系结构中抽象出来的模型和语言的可用性对并行软件的开发过程有着很重要的影响,也对并行计算系统能否获得广泛应用产生重要影响.第 2.2~2.4 节主要从这一角度来讨论模型和语言的发展现状.

2.2 共享存储的模型和语言

共享存储的程序设计模型大多是由 PVP(parallel vector processor)和 SMP(symmetric multiprocessor)平台提供,共享存储的程序大多是在特定的多处理器平台上用平台专用的语言写成.在这种模型上,数据处在单一地址空间,分为共享和私有两种,数据通信通过共享存储来完成.平台独立的共享存储并行程序设计的标准模型有 X3H5[1]、Pthread[2]和 OpenMP[3].X3H5 标准于 1993 年建立,它没有像 MPI 标准那样被广泛接受,现在已发展成为 OpenMP 标准.POSIX Threads(简称 Pthreads)标准,由 IEEE 标准化委员会建立,在功能和接口方面都类似于 Solaris 的线程.当前最重要的共享存储标准是 OpenMP.它通过一组编译说明、库例程和环境变量为 UNIX 和 Wondows NT 平台提供共享存储的应用程序接口.支持共享存储并行的标准 Fortran 语言有 PCF(parallel computing forum)[4].它是在 Fortran 77 里加入并行结构,使程序员能对并行性进行控制,但需要程序员关注进程同步和数据共享问题.商用共享存储语言的例子有 SGI Power C[5].它在串行 C 语言的基础上扩展了编译制导(compiler directives)和库函数,用于支持共享变量的并行程序设计.SGI 对 Fortran 也提供了类似的扩展结构.

2.3 消息传递的模型和语言

分布存储结构的出现带来了两大焦点问题:局部性要求和并行程序设计的复杂性.一种解决局部性的方法是使用显式描述并行的方法,对应于采用消息传递库的消息传递模型和采用说明语句的数据并行模型.

在消息传递模型中,一个并行应用由一组进程组成,每个进程的代码是本地的,只能访问私有数据,进程之间通过传递消息实现数据共享和进程同步.消息传递的优点是用户可以对并行性的开发、数据分布和通信实现完全控制,主要缺点是要求程序员显式地处理通信问题.对大多数科学计算程序来说,消息传递模型的真正困难还在于显式的域分解,即将对相应数据的操作限定在指定的处理器上进行.另一个问题是,消息传递程序无法以渐进的方式,通过逐步将串行代码转换成并行代码而开发出来.

CMMD[6]是一个用于 Thinking Machines CM-5 系统的消息传递库,其特点是基于主动消息(active message)机制在用户空间实现通信,以减少通信延迟.Parasoft 公司的 Express[7]是一个支持点到点和群集通信以及并行 I/O 的程序设计环境.Nx[8]是为 Intel MPP(例如 Hypercubes 和 Paragon)开发的微核系统.现在已由用于 Intel/Sandia ASCI TFLOPS 系统中的新的微核系统 PUMA 所代替.Fortran-M[9]是对 Fortran77 的扩展.它在设计上既支持共享存储也支持消息传递,但当前只实现了对消息传递的支持.该语言提供了许多机制,用于支持开发行为确定的、模块化的并行程序.其他的消息传递软件系统还有 P4,Vertex,PARMACS,Zipcode,UNIFY 和 PICL 等.

在当前所有的消息传递软件中,最重要、最流行的是消息传递接口标准 MPI(message passing interface)[10]和免费软件 PVM(parallel virtual machine)[11].它们能运行在所有的并行平台上,包括 PVP、SMP、MPP(massively parallel processor)、工作站和 PC 组成的机群系统,并已经在 Windows NT 和 Windows 95 这样的非 Unix 平台上实现,提供了对 C 语言、Fortran 语言和 Java 语言的绑定.在国产的三大并行机系列——神威、银河和曙光上都实现了对 MPI 和 PVM 的支持[12].

2.4 数据并行的模型和语言

数据并行模型的目的是要在分布存储的机器上实现在全局名空间进行并行程序设计,以屏蔽显式的通信问题.它是一种细粒度的并行.数据并行语言用单线程控制配合用户定义的数据和计算在处理器上的分布注释,使用户能够有效地将一些低层细节留给编译器和运行时系统去实现,用户所要做的全部工作就是给出有关并行的指示(parallel directives),说明哪段程序要并行执行.与消息传递模型相比,数据并行程序设计能在一定程度上减轻程序员的负担,但是完全依赖于程序员能否确定一个好的数据分布.

最早允许用户控制数据布局(the layout of data)的语言是为 SIMD 机器 ILLIAC IV 开发的语言 IVTRAN[13].在 MIMD 环境下第一个引入分配声明(distribution declarations)的语言是 Kali[14].Kali 编译器[15]是第一个集成静态和运行时通信策略的编译器.Thinking Machines 联合 COMPASS 公司在 Connection Machine 上的 Fortran-8x 的子集中引入了静态布局说明(static layout directives)以及数组对准(alignment of arrays)说明[16].Fortran D[17]采用了稍有不同的指示数据分布的方法,即所谓“分解(decompositions)”的方法.它通过先将数据数组对准虚拟数组的方法来指示数据分布,然后用类似于 CM-Fortran 中的方法,对不同维使用相对权重分配到一个隐式的处理器集合中.该语言允许根据简单规则的分布和不规则的分布来扩展一组对准.Vienna Fortran[18]是上述 Fortran 语言中第一个提供了完整的定义映射结构的语言,它主要基于 Kali 模型.允许数组与数组的对准,因此能够将数组显式地分布到一个处理器阵列上.Thinking Machines 公司为它的 Connection Machine CM-2 和 CM-5 开发了几个有影响的数据并行语言,像 C*,CM Fortran 和*Lisp.一些其他的项目也对促进数据并行及其编译技术的开发有所贡献,其中包括很多商业上的努力[19].

最重要的数据并行语言是 Fortran 90[20]和 HPF(high performance Fortran)[19,21].HPF 是 Fortran 90 的扩充.它们都与现有的 Fortran 语言标准兼容.随着体系结构的发展和科学计算程序设计越来越复杂,该语言的局限性变得越来越明显.对 HPF 的批评主要有 3 个方面:① 认为 HPF 是一个太高层的语言,不如 MPI 风格的语言适用;② 认为 HPF 是一个太低层的语言,通过改进语言编译技术和体系结构,完全可以避开 HPF 风格的语言所要求的编译制导;③ 认为 HPF 尽管抽象层次适当,但还要作一些扩充才能满足在某些未来的体系结构上处理某些

应用的需要.

2.5 模型研究的其他观点

以模型的抽象性作为分类的依据,并行程序设计模型还可以分为显式(抽象层次低)和隐式(抽象层次高)两种类型.D.B.Skillicorn 和 D.Talia 从模型的抽象性角度对各种引入了重要思想的并行程序设计模型和语言进行了评价,阐述了它们的发展历史和相互关系,着力描述了各种模型的关键特性[22].我们还可区分基于并行计算理论的模型(如 PRAM[23],BSP[24]和 LogP[25]模型)、基于问题描述的模型(如 GAMMA[26]和 UNITY[27]模型)、基于程序构造的模型(如 CSP[28]和 Linda[29]模型),也可以根据模型的可表达性和可扩放性来对模型进行分类.限于文章的篇幅,在此不一一讨论.

总的说来,尽管有关并行程序模型和语言的研究异彩纷呈,但根据厂家支持和用户人数的规则,当前在实际应用中两种最成功的并行程序设计模型是显式的消息传递标准 MPI 和共享存储的标准 OpenMP.在这两者当中,只有 MPI 的成功是名副其实的.OpenMP 比较新,还需要时间的考验.

3 并行程序设计模型的评价标准

通过对大量现有的各种并行程序设计模型和语言的分析,可以总结出一个好的并行程序设计模型所应具备的性质.我们可以将这些性质作为模型的评价标准,用于比较各种模型的主要性能和指导新模型的设计.这些性质包括:

(1) 尽可能抽象和简单,易于学习与理解.大规模并行程序设计通常是非常复杂的,常常要产生大量并行执行的进程(或线程),并要求能可靠地控制它们的执行,提供进程之间交互的手段.为了减少算法或程序设计人员的负担,模型应能通过一定的抽象,将尽可能多的指示程序并行执行的程序结构由转换机制(如编译器和运行时系统)来插入,而不应由程序员手工去做.这就意味着模型应该尽可能隐藏以下编程细节:① 程序到并行线程的分解;② 线程到处理器的映射;③ 线程间的通信;④ 线程间的同步,从而为用户提供一种简洁的描述程序并行的手段,使得模型易于学习和理解,否则,软件开发者就不会愿意使用它.

(2) 体系结构独立.由于处理器和互连网络技术发展迅速,计算机系统结构只有较短的生命周期.并行计算的用户必须准备好可能每隔 5 年甚至更短的时间就要更换他们所用的并行机,并且新的并行计算机不可能与被更换掉的完全一样.如果想普及并行计算,就必须将并行软件从并行计算机底层的更新变化中隔离出来.模型应能够描述在不同体系结构的并行机上实现的并行算法或程序.模型越抽象,其体系结构的独立性越强,算法和软件的可移植性也就越强.

(3) 能提供一套完整的软件开发方法.迄今为止已有的大量并行软件都是数值的或用于科学计算的,对它们的开发都没有优先考虑开发方法.这有两方面的原因:① 许多数值计算基本上都是线性代数类的,程序的结构大多是规则的,并与导出它们的数学方法相关,因此这种软件比起大量的非科学计算类应用来说,相对要简单;② 由于科学计算类的应用大多属于研究性质,而不是生产性的,因此很少强调这类软件发展的长期性,因为许多程序只打算短期使用.随着生产性并行应用的增加,比起串行程序设计,完整的并行软件开发方法似乎是更为基本的问题.我们需要经过长期的工作来建立正确的并行软件构造方法.模型的研究必须充分地考虑这一点.

(4) 能够保障性能.模型应该能在各种并行系统结构上保障性能.现在我们已经完全明白,通信性能是对系统结构的基本限制.系统性能降低的原因通常是由于通信拥塞.只有支持限制通信频度的模型,或者是作了足够的限制来实现程序和数据局部性的模型,才能在所设计的并行计算机上保证好的性能.

(5) 可测量程序的成本.程序的成本主要包括程序的执行时间,其次是处理器的利用率和软件开发的成本.在并行程序开发中对并行程序作小的改变和对目标计算机的选择都会影响程序的成本.一个程序在模型中的性能与其在实现中的性能之间应该存在等价关系.成本测量还要与模块性配合好.现代软件几乎都是在不同的时间里以分块的方式开发出来的,这意味着要能给每个开发小组限定一个资源预算.如果每个小组都能满足分配给他们的个别成本,就可以实现总的成本目标.这就意味着成本测量必须是可合成的,总的成本易于由其部分成本计算出来.

在上述并行程序设计模型的评价准则之间不可避免地存在着一定的相互对立和矛盾.如,高层抽象提高了模型的通用性和简洁性,但同时失去了模型应具有的精确性;低层抽象虽然保证了模型的精确性,但却限制了模型的通用性,并增加了描述并行的复杂性.为了缓和这种矛盾,目前对模型的设计和使用大多是在这些性质之间进行了某种权衡.

4 未来的挑战和发展趋势

在过去 10 年里,并行计算的应用领域发生了两方面的变化:一方面是由市场较小的科学和工程计算转向市场巨大的商业和工业应用;另一方面是并行计算应用的规模和复杂性大大增加,呈现出高性能、多样性和多功能的发展趋势.对于许多复杂的应用(其中包含许多不同类型的子问题),不仅要求高性能,而且还要求快速的代码开发、代码可移植、性能可移植和稳定性,同时要求集成各种现有的标准、软件包、库和工具,并且在多平台、多学科、异构的环境下协同工作.

硬件方面的变化主要体现在商品化的并行系统开始流行由共享存储的计算节点构成的机群系统(SMP-nodes MPP).它是一种共享存储与分布存储集成的混合结构,其主要特性之一是使用多级存储,常常带有多级高速缓存.这种新的体系结构对并行程序设计模型和语言提出了一系列新的挑战.在这种结构上不仅要处理分布存储机器上的局部性问题,而且要处理缓存局部性问题.在 SMP 节点上适合做基于线程的程序设计,而在节点之间更适合采用消息传递的程序设计.这种混合编程模式可能会使已经十分复杂的算法设计任务变得更加困难[30].

面对这些变化,我们认为应着力于改变并行程序设计模型和语言在以下技术层次上的不足:

(1) 层次的并行程序设计模型.并行程序设计模型应该代表处于各个层次上的各种并行程序设计方法.在多学科的应用中,必须借助高层的模型和自动化工具来重建和优化可能有很大规模的异质软件模块的集合,这些异质模块可能是用不同标准的语言和工具在多种平台上分别开发出来的.基于这种考虑,我们提出的并行程序设计模型包括并行软件体系结构(高层)、并行程序框架(中间层)和代码(低层)三个设计层次,在每个层次上分别研究新的或集成现有的成功的设计方法.

(2) 可视化的并行软件构造方法.在高层进行并行程序设计时,应该特别强调可视化的设计方法[31].用可视编程语言开发并行软件,设计者可以把头脑中的概念模型先用一张图“画”出来,这张图就是可视的并行程序.可视并行程序由系统提供的变换程序自动或半自动地翻译成在具体的并行机上可并行的执行代码,用户主要关心系统的概念模型,不必一开始就陷入复杂的并行语言的编程细节之中.我们正在研究以可视化的标准建模语言 UML(unified modeling language)[32]为框架,建立面向对象的可视化并行软件开发方法.

(3) 面向对象的并行程序设计语言.传统的并行程序设计方法主要基于功能分解,存在着功能抽象较为困难、功能容易变化以及功能分解的随意性等问题,因而不适于并行程序的设计、调试和复用,并发的面向对象程序设计语言正在成为新的研究热点[33].近年来,计算科学界也开始倾向于采用更先进的面向对象方法,而不再是抱着 Fortran 语言不放[34].从短期来看,让计算科学界学习新的面向对象语言和方法似乎很费时间,但从长远来看,这将有利于从根本上产生更好的并行软件,提高并行软件的生产效率.

(4) 并行程序动态行为分析理论.由于并发、通信、同步和不确定的特性,使得并行程序的正确性验证和性能效率的判断非常困难.并行程序的正确性调试、性能的分析调整不应该全部放到设计的最后阶段——运行时来做,而是要借助并行程序动态行为分析理论,支持开发过程中的正确性验证和性能调整.我们认为,日渐成熟的高级 Petri 网论已经可以有效地描述和分析复杂的并发、并行和分布系统[35],是一个值得重视的研究方向.

(5) 异质环境下的并行计算问题.我们注意到,中间件技术 CORBA[36]和跨平台的面向对象语言 Java[37]已经解决了传统的硬件、网络和语言方面的低层互操作问题,为实现分布计算提供了很好的解决方案.我们可以在此基础上开展对网格计算(或称元计算)环境下的并行程序设计研究.

并行程序设计方法既受到来自底层的驱动,即技术上的可行性,又受到来自顶层的驱动,即理论上的完美.但是,我们认为,并行程序设计方法最大的进步,或者说最有希望看到的未来发展趋势,是从中间层驱动,即着力于在模型层解决并行程序设计所面临的一系列问题.目前,在中国科学技术大学计算机科学技术系和国家高性

能计算中心(合肥)正在进行的并行程序设计模型、语言、环境和工具的研究[38,39]正是沿着这个方向进行的.

References:

[1] American National Standards Institute. ANSI Technical Committee X3H5. Parallel Processing Model for High-Level Programming Languages, 1993.

[2] IEEE. POSIX P1003.4a: Threads Extension for Portable Operating Systems. Piscataway, NJ: IEEE Press, 1994.

[3] OpenMP Standards Board. OpenMP: a Proposed Industry Standard API for Shared Memory Programming. 1997. http://www.openmp.org/openmp/mp-documents/paper/paper.Html.

[4] Parallel Computing Forum. PCF: parallel Fortran extensions. Fortran Forum, 1991,10(3):1.

[5] Silicon Graphics, IRIS Power C User's Guide, Silicon Graphics Computer Systems, Mountain View, CA, 1989.

[6] Tucker, L.W., Mainwaring, a. CMMD: active messages on the CM-5. Parallel Computing, 1994,20(4):481~496.

[7] Kolawa, A. Parasoft: a comprehensive approach to parallel and distributed computing. In: IEEE Computer Society, ed. Proceedings of the Workshop on Cluster Computing. Los Alamitos, CA: IEEE Press, 1992.

[8] Pierce, P., Regnier, G. The paragon implementation of the NX message passing interface. In: IEEE Computer Society, ed. Proceedings of the Scalable High-Performance Computing Conference. Los Alamitos, CA: IEEE Press, 1994, 184~190.

[9] Foster, I., Chandy, K.M. Fortran M: a language for modular parallel programming. Journal of Parallel and Distributed Computing, 1995,26(1):24~35.

[10] CORPORATE the MPI Forum. MPI: a message passing interface. In: ACM, ed. Proceedings of the conference on Supercomputing'93. New York: ACM, 1993. 878~883.

[11] PVM Home Page at UTK. http://netlib2.cs.utk.edu/pvm.

[12] Chen, Guo-liang. Parallel Computing: Architecture·Algorithm·Programming. Beijing: Higher Education Press, 1999 (in Chinese).

[13] Millstein, P.E. Control structures in ILLIAC IV Fortran. Communications of the ACM, 1973, 16(10):621~627.

[14] Mehrotra, P., Rosendale, J.V. Programming distributed memory architectures using Kali. In: Nicolar, A., Gelernter, D., Gross, T., *et al.*, eds. Advances in Languages and Compilers for Parallel Processing. Pitman: MIT Press, 1991. 364~384.

[15] Koelbel, C., Mehrotra, P. Compiling global name-space parallel loops for distributed execution. IEEE Transactions on Parallel Distributed System, 1991,2(4):440~451.

[16] Albert, E., Knobe, K., Lukas, J.D., *et al.* Compiling Fortran-8×array features for the connection machine computer system. In: Proceedings of the Symposium on Parallel Programing: Experience with Applications, Languages, and Systems (PPEALS). New Haven, CT, 1988. 42~56.

[17] Fox, G., Hiranandani, S., Kennedy, K., Koelbel, C., *et al.* Fortran D language specification. Technical Report, TR90079, Houston: Department of Computer Science, Rice University, 1991.

[18] Chapman, B., Mehrotra, P., Zima, H. Programming in Vienna Fortran. Scientific Programming, 1992,1(1):1~35.

[19] Mehrotra, P., Rosendale, J.V., Zima, H. High performance Fortran: history, status and future. Parallel Computing, 1998,24(3-4): 325~354.

[20] Adams, J., Brainerd, W., Martin, J., *et al.* The Fortran 95 Handbook. Menlo Park, CA: MIT Press, 1997.

[21] High performance FORTRAN language specification. 1993. ftp://titan.rice.cs.edu.

[22] Skillicorn, D.B., Talia, D. Models and languages for parallel computation. ACM Computing Surveys, 1998,30(2):123~169.

[23] Fortune, S., Wyllie, J. Parallelism in random access machines. In: ACM, ed. Proceedings of the 10th ACM Symposium on Theory of Computing. New York: ACM Press, 1978. 114~118.

[24] Valiant, L. G. A bridging model for parallel computation. Communications of the ACM, 1990,33(8):103~111.

[25] Culler, D., Karp, R., Patterson, D., *et al.* LogP: towards a realistic model of parallel computation. In: Proceedings of the 4th ACM SIGPLAN Symposium on Principles and Practices of Parallel Programming (PPoPP'93). San Diego, CA: ACM Press, 1993. 1~12.

[26] Banatre, J.P., Metayer, D.L. Introduction to gamma. In: Banatre, J.P., LeMetayer, D., eds. Research Directions in High-Level Parallel Programming Languages. LNCS574, Springer-Verlag, 1991. 197~202.

[27] Bickford, M. Composable specifications for synchronous systems using UNITY. In: Proceedings of the International Symposium on Advanced Research in Asynchronous Circuits and Systems. 1994. 216~227.

[28] Hoare, C.A.R. Communicating Sequential Processes, International Series in Computer Science. Englewood Cliffs, NJ: Prentice-Hall, Inc., 1985.

[29] Carriero, N., Gelernter, D. Application experience with Linda. In: ACM, ed. Proceedings of the ACM/SIGPLAN Symposium on Parallel Programming, Vol 23. New York: ACM Press, 1988. 173~187.

[30] Womble, D.E., Dosanjh, S.S., Hendrickson, B., *et al*. Massively parallel computing: a Sandia perspective. Parallel Computing, 1999,25(13-14):1853~1876.

[31] Beguelin, A., Nutt, G. Visual parallel programming and determinancy: a language specification, an analysis technique, and a programming tool. Journal of Parallel and Distributed Computing, 1994,22(2):235~250.

[32] Booch, G. UML in action. Communications of the ACM, 1999,42(10):26~28.

[33] Yang, Da-jun, Zhang, Ming, Lü, Jian. The study of concurrent object oriented programming language. Computer Research and Development, 1998,35(9):769~775 (in Chinese).

[34] Gosling, J. Extensions to Java for numerical computation. In: ACM, ed. Proceedings of the ACM 1998 Workshop on Java for High-Performance Network Computing. 1998. http://www.cs.ucsb.edu/conferences/java98.

[35] Lin, Chuang. Stochastic Petri Net and System Performance Evaluation. Beijing: Tsinghua University Press, 2000 (in Chinese).

[36] Zhan, Yong-zhao, Xie, li, Sun, Zhong-xiu. The strategies of implementation of distributed application systems by using CORBA. Computer Science, 1999,26(4):13~16 (in Chinese).

[37] Arnold, K., Gosling, J. The Java Programming Language. Reading, MA: Addison-Wesley, 1996.

[38] An, Hong, Li, Hong, Wu, Ming, *et al.* A Java/CORBA based universal framework for super server user-end integrated environments. In: Chen, Jian, Lü, Jian, Meyer, B., eds. Proceedings of the 31st International Conference on Technology of Object Oriented Languages and Systems (TOOLS' 31). Los Alamitos, CA: IEEE Computer Society Press, 1999. 336~341.

[39] Wang, Feng, Zheng, Qi-long, An, Hong, *et al.* A parallel and distributed debugger implemented with Java. In: Chen, Jian, Lü, Jian, Meyer, B., eds. Proceedings of the 31th International Conference on Technology of Object Oriented Languages and Systems (TOOLS'31). Los Alamitos, CA: IEEE Computer Society Press, 1999. 342~346.

附中文参考文献:

[12] 陈国良.并行计算——结构·算法·编程.北京:高等教育出版社,1999.

[33] 杨大军,张鸣,吕建.并发面向对象程序设计语言研究与进展.计算机研究与发展,1998,35(9):769~775.

[35] 林闯.随机 Petri 网和系统性能评价.北京:清华大学出版社,2000.

[36] 詹永照,谢立,孙钟秀.使用 CORBA 实现各种分布式应用系统的策略.计算机科学,1999,26(4):13~16.

Parallel Programming Models and Languages*

AN Hong, CHEN Guo-liang

(*Department of Computer Science and Technology, University of Science and Technology of China, Hefei* 230027, *China*);

(*National High Performance Computing Center at Hefei, Hefei* 230027, *China*)

E-mail: han@ustc.edu.cn

http://www.nhpcc.ustc.edu.cn

Abstract: Parallel computing is about 20 years old. Till now, there is a lack of effective parallel programming methods and tools in high performance computing as a result that it is very difficulty to parallel programming, understanding behaviors of parallel program, debugging parallel codes and optimizing performance. In this paper, the reasons why so difficulty parallel programming is are analyzed, while the issues about parallel programming methods existing in recent high performance parallel machines are addressed, the current status of parallel programming models and languages are surveyed, the view of the criteria is offered to evaluate parallel programming models, the challenge problems in this area are brought forward, and some future research directions are pointed out.

Key words: parallel programming; models; languages; portability; scalability; locality

* Received April 20, 2000; accepted July 12, 2000

Supported by the National High Technology Development 863 Program of China under Grant No.863-306-ZD01-02-3; the Youth Foundation of the University of Science and Technology of China under Grant No.KA1109