并行算法研究方法学

陈国良 孙广中 徐 云 吕 敏
（中国科学技术大学计算机科学与技术系 合肥 230027）
（国家高性能计算中心(合肥) 合肥 230027）
（安徽省高性能计算重点实验室 合肥 230027）

摘 要 并行算法是计算机科学中重要的研究内容，已有几十年的发展历程. 回顾一下其研究历程，既有高潮也有低谷，究其原因是，它没有形成自身的一套研究方法学. 为此文中提出并行算法研究要建立起一套完整的“理论-设计-实现-应用”的学科体系，也就是所谓的并行算法研究的生态环境. 只有这样才能够保持并行算法研究稳定、可持续发展，并使得并行算法的研究成果更加实用，从而更富有生命力.

关键词 方法学；并行算法理论；并行算法设计与分析；并行算法实现；并行应用
中图法分类号 TP301

Methodology of Research on Parallel Algorithms

CHEN Guo-Liang SUN Guang-Zhong XU Yun LU Min
(*Department of Computer Science, University of Science and Technology of China, Hefei* 230027)
(*National High Performance Computing Center at Hefei, Hefei* 230027)
(*Anhui Provincial Key Laboratory of High Performance Computing, Hefei* 230027)

Abstract Parallel Algorithm is an important research area in computer science, with a development history of several decades. Based on the research history with downturn and climax, the authors figure out the reason for that is absent of a research methodology on parallel algorithms. This paper concludes research on parallel algorithms should establish a complete system of "Theory-Design-Implementation-Applications", which is ecological environment of parallel algorithms research. Only in this way can we maintain the steady and sustainable development of parallel algorithms research. Moreover, the results of research on parallel algorithms will be more practical and more full of vitality.

Keywords methodology; parallel algorithm theory; design and analysis of parallel algorithms; implementation of parallel algorithms; parallel applications

1 引 言

简单地讲，并行算法就是用多台处理机联合求解问题的方法和步骤，其大致的执行过程是将给定的计算问题首先分解成若干个尽量相互独立的子问题，然后使用多台计算机同时求解它. 在现实世界中存在着大量的并行性，在日常生活中，人们可能自觉或不自觉地都在运用着并行，只不过人们往往习惯了串行方式，而对并行性似有陌生之感而已. 同时就计算科学而言，并行计算理论仍处于发展阶段，特别是早期并行机均很昂贵，编写并行软件又很难，所以

收稿日期：2008-06-03. 本课题得到国家自然科学基金重点项目(60533020)资助. **陈国良**，男，1938 年生，教授，博士生导师，中国科学院院士，研究领域为并行计算模型、并行算法. E-mail：glchen@ustc. edu. cn. **孙广中**，男，1978 年生，博士，讲师，研究方向为并行计算、组合优化. **徐 云**，男，1960 年生，博士，副教授，研究方向为并行算法、生物信息学. **吕 敏**，女，1977 年生，博士，讲师，研究方向为并行算法、组合图论.

并行性的优点尚未被普遍地认同.

人们对并行性的需求是很明显的.首先,对于那些要求快速计算的应用问题,单处理机由于器件受物理速度的限制而无法满足要求,所以使用多台处理机联合求解就势在必行了.其次,对于那些大型复杂的科学工程计算问题,为了提高精度,往往需要加密计算网格,而细网格的计算也意味着大计算量,也通常需要在并行机上实现.最后,对于那些实时性要求很高的应用问题,照例都要在并行机上用并行算法实现之.

随着半导体器件工艺水平的提高,计算技术和通信网络的迅速发展,人们自行用多台 PC 机搭建的集群系统也越来越多,并行机的用户要求学习和使用并行算法也甚为迫切,这给研究并行算法带来了新的机遇.特别是近几年来,多核(Multi-core)和众核(Many-core)的出现,桌面并行机系统和成千上万个 CPU 的高端并行机相继研制成功,这一方面为并行算法的研究再次带来了新的机遇,但与此同时,面临如此巨量的 CPU,人们如何充分有效地利用由此带来的巨大并行性,却又成了一个极富有挑战的问题.

并行算法和并行计算,两者仅一字之差,所以现在常被混淆为同一回事.其实两者是不尽相同的,并行算法是并行计算的子集,并行计算的研究面更广一些,它包括了并行计算的硬件平台(即并行计算机)、并行计算的软件支撑(即并行程序设计)和并行计算的理论基础(即并行算法).并行算法的研究比较偏重于学科理论,而并行计算的研究更注重实际应用,例如大规模科学与工程计算等.本文重点介绍并行算法的研究方法学,旨在如何保持并行算法这一学科的健康可持续发展.

并行算法的研究大体上可以分为数值并行算法(Numeric Parallel Algorithm)和非数值并行算法(Non-numeric Parallel Algorithm),前者研究基于代数关系运算的一类数值计算问题的并行算法,主要包括矩阵运算、方程组的求解和数字信号处理等;后者研究基于比较关系运算的一类符号处理问题的并行算法,主要包括图论问题、数据库操作和组合优化等.当然,这只是为了研究的方便,其实两者是相通的,我们可以用数值并行算法解决非数值计算问题,反之亦然.

从历史上看,并行算法研究,特别是非数值并行算法的研究,高峰期似乎在 20 世纪 70 年代和 80 年代.这一阶段,在各种不同互连结构的 SIMD 模型上和共享存储的 SIMD 模型上设计出了很多优秀的非数值并行算法.相应的在这一时期,也出版了几部非常优秀的并行算法方面的专著和教材[1-9].20 世纪 90 年代中期以后,并行算法的研究渐渐面向实际而内容有所拓宽,在研究并行算法设计与分析的同时也适当地涉及到了并行机体系结构和并行程序设计等有关内容,集中表现在上述几本书的作者,有的在书名中更强调了并行计算而出版了第 2 版,有的将内容扩充,另出版了新书[10-13].但总的看来,20 世纪 90 年代中期以后,纯并行算法的研究似乎低落了下来.

回顾一下并行算法的研究历程,既有高潮也有低谷,究其原因是,它没有形成自身的一套研究方法学.为此我们认为:并行算法研究要建立一套完整的如图 1 所示的"理论-设计-实现-应用"的学科体系,也就是所谓的并行算法研究的生态环境.其中,算法理论包括并行计算模型和并行计算复杂性等;算法设计包括并行算法的设计和并行算法的分析等;算法实现包括软件支撑和硬件平台等;并行应用包括科学工程计算应用和社会计算应用等.我们认为,只有这样,并行算法的研究才能稳定、可持续发展且变得更加实用,从而更富有生命力.

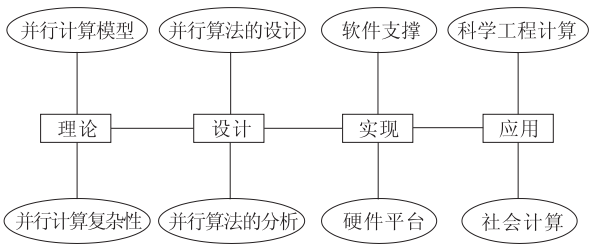

图 1 并行算法研究学科体系

本文第 2 节为并行算法理论;第 3 节为并行算法的设计与分析;第 4 节为并行算法的实现;第 5 节为并行算法的应用;最后是结论.

2 并行算法理论

2.1 PRAM 模型及其等效性

(1) PRAM 模型

我们在设计某一特定问题的有效算法时都致力于将所考虑的问题高度并行化,采用功能尽可能强的并行计算模型,以期达到成本最佳或接近最佳.

并行随机存取(PRAM)模型[14]是目前使用很

广泛的一种抽象的并行计算模型. 在这种模型中，假定存在着一个容量无限大的共享存储器；有有限或无限个功能相同的处理器，且均具有简单的算术运算和逻辑判断功能；在任何时刻各处理器均可通过共享存储单元相互交换数据. 根据处理器对共享存储单元同时读、同时写的限制，PRAM 模型可分为：① 不允许同时读和同时写的 PRAM 模型，简记为 PRAM-EREW；② 允许同时读但不允许同时写的 PRAM 模型，简记为 PRAM-CREW；③ 允许同时读和同时写的 PRAM 模型，简记为 PRAM-CRCW. 显然，允许同时写是不现实的，所以又对 PRAM-CRCW 模型作进一步约定：允许任意处理器自由写的 PRAM-CRCW 模型，称为任意的 PRAM-CRCW 模型，简记为 APRAM-CRCW；只允许所有的处理器同时写相同的数，称为公共的 PRAM-CRCW 模型，简记为 CPRAM-CRCW；只允许最优先的处理器先写，称为优先的 PRAM-CRCW 模型，简记为 PPRAM-CRCW. 上述模型中，PRAM-EREW 是最弱的计算模型，而 PRAM-CRCW 是最强的计算模型. 令 T_M表示某一并行算法在并行计算模型 M 上的运行时间，则有

$$T_{\mathrm{EREW}} \geq T_{\mathrm{CREW}} \geq T_{\mathrm{CRCW}}.$$

(2) PRAM 模型的等效性

1979 年，Eckstein 采用二叉树的方法来解决读写冲突问题[15]. 这样，一个具有时间复杂度 T 和空间复杂度 S 的 PRAM-CREW 或 PRAM-CRCW 算法可在 PRAM-EREW 上模拟实现，其时间和空间复杂度分别为

$$T_{\mathrm{EREW}} = O(T_{\mathrm{CREW}} \times \log p) = O(T_{\mathrm{CRCW}} \times \log p),$$

$$S_{\mathrm{EREW}} = O(S_{\mathrm{CREW}} \times p) = O(S_{\mathrm{CRCW}} \times p),$$

其中，p 为处理器数目. 可见用二叉树解决读写冲突是费空间的.

1983 年，Vishkin 提出另一种解决读写冲突的方法[16]，其结果是

$$T_{\mathrm{EREW}} = O(T_{\mathrm{CREW}} \times \log^2 p) = O(T_{\mathrm{CRCW}} \times \log^2 p),$$

$$S_{\mathrm{EREW}} = O(S_{\mathrm{CREW}} + p) = O(S_{\mathrm{CRCW}} + p).$$

虽然省了空间，却增加了时间.

不同的 PRAM-CRCW 模型之间也可相互模拟：① 如果对处理器数或存储空间不作任何限制，则 PPRAM-CRCW = APRAM-CRCW = CPRAM-CRCW. ② 在 p 个处理器 PPRAM-CRCW 上时间为 T 的算法可以在 $p\log p$ 个处理器 CPRAM- CRCW 上用时间 $O(T)$ 模拟之. ③ p 个处理器 PPRAM-CRCW 中一条并发写指令，可以在 p 个处理器 CPRAM- CRCW 上用时间 $O(\log p/\log\log p)$ 模拟之. ④ p 个处理器 PPRAM-CRCW 中一条并发写指令，可以在 p 个处理器 APRAM-CRCW 上用 $O(\log\log p)$ 时间模拟之.

2.2　分代和分层并行计算模型

(1) 三代并行计算模型

PRAM 模型的好处在于它十分抽象，对于从事并行算法理论研究的研究者非常适用，缺点就是距离实际并行机结构相差较远. 在过去的几十年的研究中，从经典的 PRAM 模型开始，前后发展出了如图 2 所示的三代并行计算模型[17].

图 2 中，第一代计算模型为共享存储模型，包括共享存储 SIMD 机器的同步 PRAM 模型和分布存储 MIMD 机器的异步 APRAM 模型等；第二代计算模型为分布存储模型，包括分布存储 MIMD 机器的大同步 BSP 模型和异步 LogP 模型等；第三代计算模型为分布共享存储模型，包括均匀存储层次 UMH 模型和分布存储层次 DRAM(h)模型以及层次并行存储 HPM 模型等. 这些新的并行计算模型的研究往往是为了解决并行计算模型与实际并行机结构之间的差异[18].

(2) 分层并行计算模型

随着并行机体系结构的快速发展，按照图 2 所示，并行计算模型沿着丰富、强化单一模型的路线在发展和演变，致使单一的并行计算模型变得越来越复杂，最终导致模型不实用和不可操作. 为此文献[19]提出了对并行计算模型分层研究的思想，

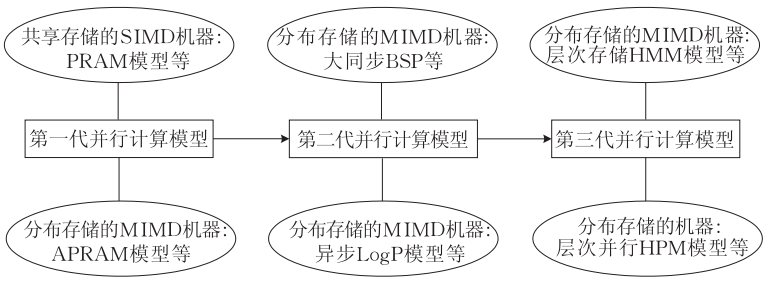

图 2　三代并行计算模型

如图 3 所示，并行计算模型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等三个层次. 通过分层，各层的模型分工明确、目标单一，从而便于使用.

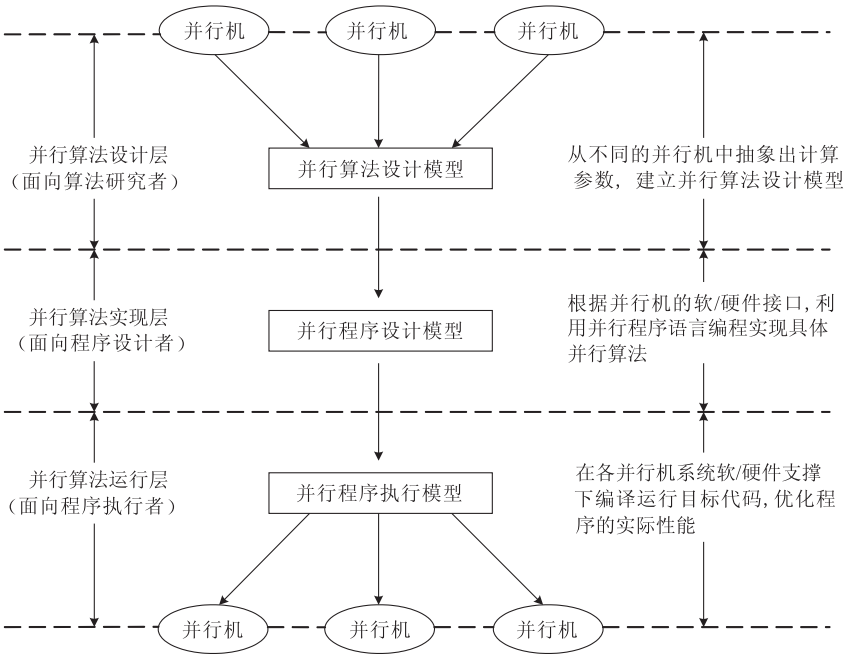

图 3 分层并行计算模型

2.3 可并行化和 NC 类问题

(1) 问题的可并行化

有些问题即使使用了相对多的处理器也似乎很难并行处理，所以我们将问题按照可并行性进行分类.

一个问题，如果处理器在某一取值范围内（例如 $1\leq p\leq \alpha(n)$，$\alpha(n)$ 为问题规模的增函数）随着处理器数目的增加可较快地求解，则就定义此问题为可并行化的. 如果在给定的 p 的范围内，并行算法能 $\Theta(p)$ 倍快于已知的最好的串行算法，则称其达到最佳性.

一个问题，如果用多项式数目的处理器，在对数多项式（Polylogarithmic）时间内可求解，则也可定义此问题为可并行化的.

(2) 并行复杂性理论

并行复杂性理论的研究类似于串行复杂性理论的研究. 我们可以比照串行复杂性理论中的 P 类、NP 类和 NP-完全（NPC）类问题来相应地研究并行复杂性理论中的 P 类、NC 类和 P 完全（PC）类问题.

在确定的图灵机上，多项式时间内可以求解的一类问题，称为 P 类问题.

P 类问题在不同的计算模型下保持不变，特别地，P 类问题能定义为所有可在 PRAM 上用多项式时间求解的问题.

在 PRAM 模型上，使用多项式数目的处理器，在对数多项式时间内可求解的一类问题称为 NC 类问题.

由上述两定义可知，NC⊆P 是显然的，但是 P⊆NC 是否成立却是复杂性理论中的一个基本问题. 目前在学术界，公认的假设是 P≠NC.

给定两个问题 L' 和 L，如果存在一个 NC-算法，可以将求解的问题 L' 转化为对问题 L 的求解，则称为 L' 可 NC-归约到 L.

如果问题 $L\in P$，且 P 类中的每一个问题 L' 均可 NC-归约到 L，则称问题 L 是 P 完全问题，简记 P 完全问题的集合为 PC 类问题.

所有 P 完全的问题在 NC-归约下都是等效的. P 类、NC 类和 PC 类这三类问题之间的关系，可用图 4 表示.

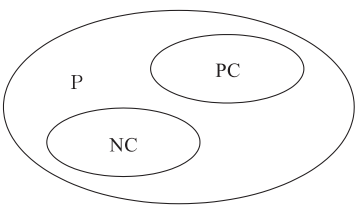

图 4 P、NC 和 PC 之间的关系

为了建立并行复杂性的理论基础，我们将 PRAM-CRCW 模型和无界扇入电路联系起来[20]，

利用并行时间、顺序空间和电路深度三者之间的关系，借助无界扇入电路的下界来推出 PRAM-CRCW 模型的下界.

1975 年，Ladner 证明了电路值问题(CVP)是 P 完全问题[21]. 该问题及其一些变体为建立很多其它问题的 P 完全性提供了一个强有力的工具. 对问题 $L \in P$，我们只需将一个已知的 P-完全问题 NC-归约到 L，即可证明 L 也是 P 完全的.

3　并行算法的设计与分析

并行算法是一些可同时执行的诸进程的集合，这些进程互相作用和协调动作从而达到给定问题的求解[9]. 并行算法总体上可分为非数值并行算法和数值并行算法两大类，对于非数值并行算法的设计方法在文献[22]中已有论述，本文重点介绍数值并行算法的设计方法. 虽然前者的大多数设计方法可以用于后者中，但由于数值并行算法面向计算问题的独特性，仍具有一些不同之处，似有必要进行总结和分析. 在后面的论述中，我们主要介绍数值并行算法的设计策略、设计方法以及并行算法的复杂性标准与性能评估. 本文所讨论的数值并行算法是基于数值分析的数学原理，将不涉及算法中的稳定性和收敛性等问题.

3.1　数值并行算法的设计策略

类似于非数值并行算法的设计策略，数值并行算法一般有两种设计策略(如图 5 所示)：① 串行算法的直接并行化：就是充分发掘和利用现有串行算法中的并行性，直接将串行算法改造为并行算法，这是数值并行算法中最常用的设计策略. ② 从计算问题和数值计算原理开始从头设计：就是从问题本身和计算方法出发，不考虑相应的串行算法，设计一个全新的并行算法. 如我国计算数学家周毓麟先生提出的具有并行本性差分格式算法就是一个很好的例子[23]. 设计全新的并行算法是一个挑战性和创造性的工作，需要充分挖掘问题的固有特性与并行的关系.

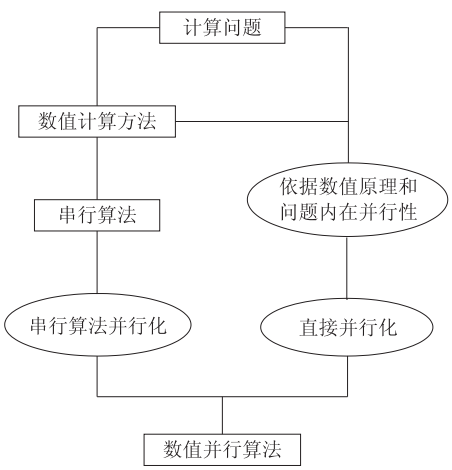

图 5　数值并行算法设计策略

第①种策略是在计算过程(某一串行算法)已确定的前提下，挖掘串行算法中的并行性，再考虑到适当的数据分布. 第②种策略要抛开已有的串行算法，从问题本身内在的并行性和数值计算原理出发且同时考虑数据分布，重新设计利于并行化的算法.

3.2　数值并行算法的设计方法

数值并行算法总体上遵循 PCAM 方法学[24]，在每个步骤中又会应用具体的设计方法. PCAM 设计过程分为 4 个步骤：划分(Partitioning)、通信(Communication)、组合(Agglomeration)和映射(Mapping). 它反映了并行算法设计的自然过程，其首先尽量开拓算法的并发性和满足算法的可扩放性，然后着重优化算法的通信成本和全局执行时间，同时通过必要的整个过程的反复回溯，以期最终达到一个满意的设计，具体过程如图 6 所示.

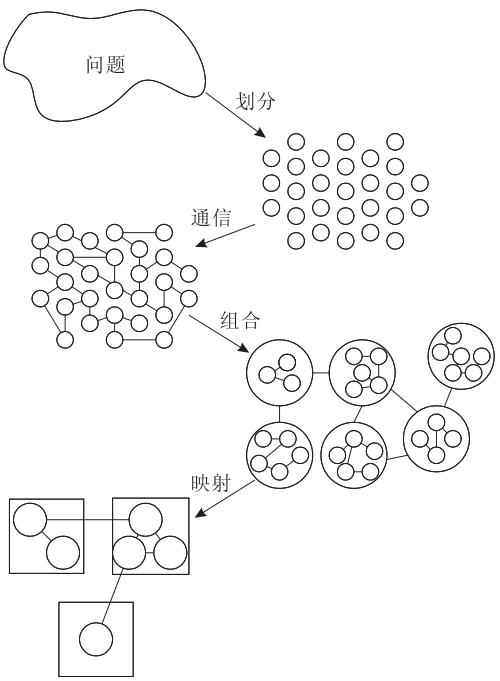

图 6　PCAM 设计过程

(1) 划分

划分阶段是将整个计算任务分解成一些小的任务，其目的是尽量开拓并行执行的机会. 主要使用域分解和功能分解方法将计算问题分解成若干个可部分或完全并行执行的子任务：① 域分解(Domain

Decomposition)又称数据划分,是将问题相关的大块数据尽可能分割成大小均匀的小数据片,并把计算关联到它所操作的数据上.如分治法、偏微分方程数值求解的区域分解法、方程组迭代求解中的红/黑着色法都属于域分解的范畴.② 功能分解(Functional Decomposition)又称计算划分,是以被执行的计算作为划分的对象,而不是计算所需的数据.虽然适用范围没有域分解广,但有时却更能揭示问题的内在结构而获得优化性能的机遇.如流水(脉动)方法是其中的典型代表.

为了后续设计的灵活性,划分得到的任务数应高于目标处理器数目一个量级.在划分中避免冗余计算和冗余存储,并保证子任务的尺寸大致相当.

(2)通信

通信阶段是确定划分后子任务执行中所需交换的数据并协调子任务的执行,由此也可检验上述划分的合理性.计算问题划分得到诸任务后常常存在彼此的数据依赖,而不能完全独立地执行,因而子任务之间需要进行数据传输,从而产生通信.由子任务间数据依赖关系的差别便产生了不同的通信模式(局部/全局通信、结构化/非结构化通信、静态/动态通信、同步/异步通信等).

在通信设计中一般要求各子任务的通信量大致相同,尽量进行近邻通信以减少全局通信,尽量使用可并行执行的通信操作.如5点差分格式进行"重新排序"代替1维的"自然排序"则会达到近邻通信的目的.

(3)组合

组合阶段是对上两个阶段所划分的子任务再进行一次整理,重新考虑在划分阶段和通信阶段所作的抉择,通过合并子任务,力图在某种计算模型上得到最有效的并行算法.其主要目的是通过合并小尺寸的任务来减少任务数,以提高效率和减少通信成本.

在组合设计中要考虑表面-容积效应(surface-to-volume effects),即每个子任务有较少的通信对象时,则增加划分粒度能减少通信次数,并同时减少总通信量.因此,在其它条件等同的情况下,高维组合一般更有效.重复计算(replication computation)也是在组合阶段可以考虑的方法,即采用不必要的多余计算来获得相关数据来减少通信要求或执行时间.

(4)映射

映射阶段是将在组合阶段合并的任务分配给各处理器.其主要目的是减少算法的总的执行时间.主要策略有两种:一是将能并发执行的任务映射到不同处理器上以增强并行度;二是将需要频繁通信的任务映射到同一处理器以提高局部性.需要考虑的负载平衡和任务调度两方面:① 负载平衡(load balancing)有很多方法,如递归对剖法、局部算法、概率方法、循环映射等.其中递归对剖法又能分化出递归坐标对剖、非平衡递归对剖、递归图剖分等方法.循环映射能衍生出反射映射、块循环映射等方法.② 任务调度(task scheduling)大致有两类,一类是经理/雇员(Manager/Worker)模式,在大规模问题中常采用层次经理/雇员模式以避免经理成为瓶颈;另一类是非集中模式.并且任务调度算法需要执行结束检测(termination detection)来检测操作是否结束.

3.3 复杂性标准和性能评估

并行算法的性能分析比串行算法考虑的因素要多,不仅仅要考虑算法的时间和空间复杂性,而且还要考虑算法的其它性能参数(加速比、效率、可扩放性等).

(1)时间/空间复杂度(time/space complexity)

并行算法和串行算法一样具有时间/空间复杂度的上界、下界和紧致界,也有最坏情形、平均情形的时间复杂度以及期望时间复杂度的概念.对于规模为 n 的问题,其时间复杂度包括:① 运行时间 $t(n)$:运行时间就是在给定的计算模型上求解问题所需的时间,即算法从开始执行到结束所经过的时间.此时间是由计算时间 t_c 和通信时间 t_r 两部分组成,一般不是这两部分的简单相加,而是要扣除它们之间重叠的时间.② 处理器数 $p(n)$:求解给定问题所需的处理器数目.③ 并行成本 $c(n)$:就是运行时间 $t(n)$ 与其所需的处理器数 $p(n)$ 的乘积,即 $c(n)=t(n)\cdot p(n)$.如果并行成本 $c(n)$ 等于最坏情形下求解此问题的最快串行算法的运行时间阶,那么称此并行算法是成本最优的,这也是并行算法设计所追求的目标.

(2)性能评估(performance evaluation)

并行算法还有其他性能参数,其主要包括:① 加速比(speed up) $S_p(n)$:$S_p(n)=t_s(n)/t_p(n)$,其中 $t_s(n)$ 为求解问题的最快串行算法在最坏情形下所需的运行时间,$t_p(n)$ 为求解同一问题的并行算法在最坏情形下的运行时间.加速比 $S_p(n)$ 反映了算法的并行性对运行时间的改进程度.若 $S_p(n)=p(n)$,则达到线性加速;若 $S_p(n)>p(n)$,

则为超线性加速(一般出现在某些特殊的应用中，如并行搜索等). ② 并行效率(parallel efficiency) $E_p(n)$：$E_p(n)=S_p(n)/p(n)$，该参数反映了并行算法中处理器的利用程度. ③ 可扩放性(scalability)评测标准，包括等效率度量标准、等速度度量标准和平均延迟度量标准等[25-27].

4　并行算法的实现

并行算法需要用某种程序设计语言，通过编译和程序加载，最后在具体的并行机上运行求解应用问题. 所以并行算法实现时，需要软件支撑(即并行程序设计)和硬件平台(即并行计算机). 由于并行系统具有较为丰富的多样性，所以同一个并行算法在不同的并行系统上实现的效率和性能可能差别很大. 因此，并行算法的实现是与并行算法研究密切相关的重要问题，直接影响着并行算法的设计分析及应用.

4.1　并行算法的实现过程

并行算法的实现是影响并行算法实际性能的重要因素，对于并行算法的研究者而言，了解在实现过程中存在的问题，有利于更好地开展并行算法的研究，增加并行算法的实用性.

(1) 算法与结构的绑定

并行算法实现的过程可以视为并行算法和具体结构绑定的过程. 在这个过程中，存在着多个进行绑定的时机：①在算法设计时，可以为特定机器结构设计并行算法；②在程序设计时，由程序员将算法映射到机器结构上；③在进行编译时，为了将代码运行在另一台机器上，由独立于结构的编译器重新编译它；④程序加载时，相同的目标代码可运行在不同结构的机器上. 显然，绑定越早，实现的性能越好；绑定越迟，实现的移植性越好.

在并行算法实现中，需要进行逻辑处理器到物理处理器的映射. 映射通常以最小化通信开销和最大化负载平衡为目标. 处理器指派问题是指将并发执行的任务指派到不同处理器上，通常将需经常通信的任务置于同一个处理器中，以最小化数据通信为目标. 这些问题在一般情况都是 NP 难解的优化问题，无法在可接受的时间内得到最优方案，通常使用各种近似算法和启发式算法来求解，得到具有一定性能保证的次优方案[28].

(2) 开发工具与环境

相对于串行算法的实现，并行算法的实现具有更大的难度. 对于并行算法的实现，需要良好的应用开发环境：① 图形化的用户编程界面，将使用并行机的工具环境以图形界面的方式集成到用户桌面上，从而方便普通用户使用并行机. ② 并行调试器为用户提供并行程序的正确性调试和性能优化调试. ③ 计算可视化工具为用户提供程序运行过程中和计算结果的动画显示.

并行算法的实现涉及许多工程性的工作，对于并行算法的研究者而言，好的并行开发工具与环境，是从事并行算法研究的重要工具，可以大大提高并行算法研究的效率和质量，为并行算法研究的可持续发展提供保障.

4.2　并行算法实现的软硬件支撑

并行算法的实现需要硬件平台和软件系统的支持. 在进行并行算法实现的过程中，实现的形式会因为硬件平台和软件系统的变化而不断改变.

(1) 并行算法实现的软件支撑

并行算法设计出来后，需要考虑如何正确、方便、快速地用某种编程语言来实现. 并行编程模型[29]，是提供给程序员使用的，为程序员提供了编程接口，隐藏了通信和任务调度的细节，在一定程度上简化了并行算法的实现. 并行编程模型需要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，提高程序的生产力.

并行编程模型是并行计算系统的低层实现与并行算法的高层抽象之间的桥梁. 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. 一个成功的并行编程模型需要具有可编程性、可移植性、可扩展性、通用泛化性.

当今流行的并行编程模型主要有大粒度的进程级的消息传递模型，如 MPI 等；中、细粒度的线程级的共享变量模型，如 OpenMP、Pthread 等；细粒度进程级的数据并行模型，如 HPF 等.

(2) 并行算法实现的硬件支撑

尽管并行编程模型应独立于具体的硬件结构，但现有的大多数并行编程模型和语言的产生与发展是受并行硬件系统结构发展的影响. 并行编程模型并不能够完全将硬件系统的特征屏蔽.

存储系统是并行硬件系统的重要组成部分，直接影响并行算法的实现. 硬件平台按照存储系统的特点一般分为共享存储系统、分布存储系统以及分布式共享存储系统等多种类型. 相应的，并行软件方

面也具有多种类型.操作系统和运行时系统,也是影响并行算法实现的重要因素.这些不同类型的软件平台的存在,为并行算法的实现提出了新的挑战.如果并行算法实现在共享存储的并行机上,编程实现时一般使用共享变量的程序设计模型;如果并行算法实现在分布存储的并行机上,算法编程实现时一般使用消息传递的程序设计模型.一般来说,并行算法在共享存储的并行机上实现时,使用 OpenMP 等标准语言实现的性能较好;算法在分布存储的并行机上实现时,使用 MPI 标准语言实现的性能较好.需要注意到的是,共享变量的程序不仅可执行在共享存储的并行机上,也可以执行在分布存储的并行机上,但相应的性能可能并不很好.同样的,消息传递的程序也可以在分布存储的并行机上执行,但也通常存在性能上的问题.

5 并行算法的应用

并行应用是推动并行算法研究的重要动力.并行算法研究应当结合科学、工程和国民经济建设中的应用需求才能不断发展,学科才具有生命力.

5.1 并行算法的主要应用领域

(1)开发并行应用待思考的问题

在开发并行应用时需要考虑的因素很多.首先需要对于应用问题有深刻的了解和认识,在此基础上考虑它是否能够并行化以及对它进行并行化的效率如何.有些问题由于其固有的特性,可能并不能够很好地并行化.同时也有些问题虽然可以较好地并行化,但并行化过程过于复杂,使得并行化的收益相对于成本而言并不高.

在进行并行算法研究的同时,也要意识到并行化并不能够解决各种问题,I/O 与网络性能、存储延迟与带宽等均可能是影响实际应用性能的主要因素.当确定通过并行来进行应用优化后,还需要慎重选择并行级别与任务粒度,对任务划分技术做新的设计,并选择合适的硬件体系结构等.

国家高性能计算中心(合肥)在多年并行算法的研究过程中,针对实际应用中的问题,进行了一系列的并行算法研究,如淮河流域的水利防洪减灾[30]、计算生物学[31]等,在解决具体应用问题的同时,也带动了学科的发展.

(2)科学工程计算应用和社会计算应用[32]

在过去 20 年,计算物理是应用的主要驱动力,这类应用具有如下共同特点:应用程序常常由定义在 $R^3\times t$ 子空间的一组偏微分方程所描述.多物理模拟的不同物理区域的计算方法各不相同.很多应用研究均主要集中在大学和研究所.相应的研究侧重点在于离散化 PDE 及其相应的线性/非线性方程的求解.此类应用中,强调并行机的速度和问题求解精度,很少关心相关的 I/O 操作.采用的并行化方法最常使用的是区域分解法,最常使用的编程模型是数据并行.

最近几年,与社会相关的应用急剧增加,与传统的计算应用分享市场空间,其共性是应用问题常常由图来定义,而不是离散的 R^3 空间.计算过程中的交互常是全局性的,而不是通过边界来交换信息的.新出现的应用,无相应的使用经验和成熟的软件.很多应用常集中在与社会相关的国计民生方面.相应的研究侧重点在于建立诚信机制:包括硬件、软件、人机界面、安全协议的使用等.此类应用中,巨量的数据,而不是科学计算,成为主要处理对象.I/O 是最为关心的因素,数据的安全、属主、管理等也带来了一系列技术、法律和人道等问题.典型的应用实例包括:社会安全、交通管理、军事应用、市场营销、风险投资、股票市场、数字图书馆、数据挖掘、Web 服务和远程医疗等.

5.2 科学工程计算中的若干支撑技术

传统科学工程计算典型应用领域包括:计算生物学、天体物理学、流体力学、计算化学、材料科学、环境科学、数值气象预报、油藏模拟、溃坝模拟、交通管理、水资源优化调度等等.这些应用主要是求解大规模偏微分方程组.为了提高计算速度和保证更高的计算精度,在大规模科学工程计算中,逐步形成了一系列具有共性的支撑技术,如网格生成、图划分、偏微分方程的求解、可重用的软件和算法等.下面简要介绍其中的三种[32].

(1)网格的产生

问题计算求解过程是将物理区域及其边界面离散化为一组点,然后将点连成网格,最后给出格点控制方程并计算求解.网格产生过程是将物理空间映射为计算空间,为所有实体表面定义精确数值描述,并沿着表面产生合适的网格.典型的网格产生方法有迪卡尔网格法、结构网格法、非结构网格法和通用网格法等.为了将计算网格映射到一组处理器,可以使用图的剖分技术进行网格的映射.典型的网格生成软件有 ADMesh 等.

(2)方程求解器

方程组求解分为稠密和稀疏线性系统两种.在

稠密线性方程组的求解中，Cholesky 分解和 LU 分解最具有代表性，典型的稠密线性代数软件包有 LINPACK、EISPACK、LAPACK、SCALAPACK等. 典型的稀疏线性代数算法有直接与迭代解法等，其中最常用的稀疏迭代解法有定常迭代法、Krylov 子空间法、预条件子法等，而典型的稀疏线性代数方程组迭代法软件包有 BlockSolve95、ISIS++、PCG 等.

(3) 可重用软件与算法

首先将已有的典型软件和并行算法进行编目分类，以方便查找. 其次将软件和算法封装成可重用形式，形成设计模板、函数与库、组件和对象等. 可重用主要方法是定义模板、开发 MPI 通信子和中性数据结构、标准库和组件等.

6 结　论

并行算法研究涉及到理论、设计、实现、应用等多个方面，要保持并行算法研究的持续性和完整性，需要建立一套完整的“理论-设计-实现-应用”的学科体系，也就是所谓的并行算法研究的生态环境. 其中，并行算法理论是并行算法研究的理论基础，包含并行计算模型和并行计算复杂性等；并行算法的设计与分析是并行算法研究的核心内容；并行算法的实现是并行算法研究的应用基础，包含并行算法实现的硬件平台和软件支撑技术等；并行应用是并行算法研究的发展动力，除了包含传统的科学工程计算应用外，还有新兴的与社会相关的社会服务型计算应用等.

在建立了“理论-设计-实现-应用”的并行算法研究的生态链后，根据其中某一部分的新的发展，就可以推动其他部分的研究，从而维持并行算法研究的可持续性发展. 比如，当新的并行计算模型被提出后，研究者便可在其上进行许多经典问题的并行算法研究. 又如当来自于实际的新应用问题时，则又会给研究者提供了研究求解新问题的新并行算法等等.

参 考 文 献

[1] Akl S G. Parallel Sorting Algorithms. San Diego, CA: Academic Press, 1985

[2] Quinn M J. Designing Efficient Algorithms for Parallel Computer. New York: McGraw-Hill Book Company, 1987

[3] Gibbons A, Rytter W. Efficient Parallel Algorithms. Cambridge: Cambridge University Press, 1988

[4] Akl S G. The Design and Analysis of Parallel Algorithms. Englewood Cliffs, NJ: Prentice-Hall, 1989

[5] Jaja J. An Introduction to Parallel Algorithms. Reading MA: Addison-Wesley, 1992

[6] Leighton F T. Introduction to Parallel Algorithms and Architectures: Arrays, Trees and Hypercubes. San Mateo, CA: Morgan Kaufmann, 1992

[7] Reif J H ed. Synthesis of Parallel Algorithms. San Mateo, CA: Morgan Kaufmann, 1993

[8] Kumar V, Gupta A et al. Introduction to Parallel Computing: Design and Analysis of Algorithms. Redwood City, CA: Benjamin/Cummings Publishing Company, Inc, 1994

[9] Chen Guo-Liang. Design and Analysis of Parallel Algorithms. Beijing: Higher Education Press, 1994(1st Edition); 2002(2nd Edition)(in Chinese)
(陈国良. 并行算法的设计与分析. 北京：高等教育出版社，1994(第 1 版)；2002(第 2 版))

[10] Quinn M J. Parallel Computing: Theory and Practice (2nd Edition). New York: MeGraw-Hill, 1994

[11] Akl S G. Parallel Computation Models and Methods. Englewood Cliffs, NJ: Prentice-Hall, 1997

[12] Grama A, Gupta A et al. Introduction to Parallel Computing (2nd Edition). Redwood City, CA: Benjaming/Cummings Publish Company, Inc.; Beijing: China Machine Press, 2003

[13] Chen Guo-Liang. Parallel Computing: Architecture, Algorithm, Programming. Beijing: Higher Education Press, 1994 (1st Edition); 2002(2nd Edition)(in Chinese)
(陈国良. 并行计算——结构·算法·编程. 北京：高等教育出版社，1999(第 1 版)；2003(第 2 版))

[14] Forture S, Wylie J. Parallelism in random access machines// Proceedings of the 10th Annual ACM Symposium on the Theory of Computing. 1987: 114-118

[15] Eckstein D M. BFS and biconnectivity. Department of Computer Science, Iowa State University of Science and Technology, Ames, Iowa: Technical Report 79-11, 1979

[16] Vishkin U. Implementation of simultaneous memory access in models that forbid it. Journal of Algorithms, 1983, 4(1): 45-50

[17] Zhang Y Q, Chen G L, Sun G Z, Miao Q K. Models of parallel computation: A survey and classification. Frontiers of Computer Science in China, 2007, 1(2): 156-165

[18] Chen Guo-Liang. More realistic parallel computational model. Mini-Micro Systems, 1995, 16(2): 1-9(in Chinese)
(陈国良. 更实际的并行计算模型. 小型微型计算机系统，1995, 16(2): 1-9)

[19] Chen Guo-Liang, Miao Qian-Kun, Sun Guang-Zhong, Xu Yun, Zheng Qi-Long. Layered models of parallel computation. Journal of University of Science and Technology of China, 2008, 38(7): 1-7(in Chinese)
(陈国良，苗乾坤，孙广中，徐云，郑启龙. 分层并行计算模型. 中国科学技术大学学报(50 周年校庆专刊)，2008, 38(7): 1-7)

[20] Stockmeyer L, Vishkin U. Simulation of parallel random access machines by circuits. SIAM Journal on Computing, 1984, 13(2): 409-422

[21] Ladner R E. The circuit value problem is log space complete for P. SIGACT News, 1975, 7(1): 18-20

[22] Chen G L, Sun G Z, Zhang Y Q, Mo Z Y. Study on parallel computing. Journal of Computer Science and Technology, 2006, 21(5): 665-673

[23] Zhou Yu-Lin. On the difference schemes with intrinsic parallelism for nonlinear parabolic systems. Journal on Numerical Methods and Computer Applications, 1995, 16(3): 162-172 (in Chinese)
(周毓麟. 非线性抛物组的具有并行本性差分格式. 数值计算与计算机应用, 1995, 16(3): 162-172)

[24] Foster I T. Designing and Building Parallel Programs: Concepts and Tools for Parallel Software Engineering. New York: Addison Wesley Publishing Company, 1995

[25] Kumar V et al. Parallel depth-first search, Part II: Analysis. International Journal of Parallel Programming, 1987, 16(6): 501-519

[26] Sun X H, Rover D T. Scalability of parallel algorithm-machine combinations. IEEE Transactions on Parallel and Distributed Systems, 1994, 5(6): 599-613

[27] Zhang X D et al. Latency metric: An experimental method for measuring and evaluating parallel program and architecture scalability. Journal of Parallel and Distributed Computing, 1994, 22: 392-410

[28] Du J, Leung J Y-T. Complexity of scheduling parallel task systems. SIAM Journal on Discrete Mathematics, 1989, 2(4): 473-487

[29] An Hong, Chen Guo-Liang. Parallel programming models and languages. Journal of Software, 2002, 13(1): 118-124 (in Chinese)
(安虹,陈国良. 并行程序设计模型和语言. 软件学报, 2002, 13(1): 118-124)

[30] Mao Rui, Huang Liu-Sheng, Xu Da-Jie, Chen Guo-Liang. Algorithm and implementation of multi-reservoir optimal scheduling on Huaihe river. Mini-Micro Systems, 2000, 21(6): 44-48(in Chinese)
(毛睿, 黄刘生, 徐大杰, 陈国良. 淮河中上游群库联合优化调度算法及并行实现. 小型微型计算机系统, 2000, 21(6): 44-48)

[31] Wang Wen-Rui, Chen Guo-Liang, Sun Guang-Zhong. Large parallel optimal algorithm for molecular dynamics software GROMOS96. Journal of University of Science and Technology of China, 2004, 34(1): 79-85(in Chinese)
(王文睿,陈国良,孙广中. GROMOS96 分子动力学模拟的并行优化算法. 中国科学技术大学学报, 2004, 34(1): 79-85)

[32] Mo Ze-Yao et al ed. Sourcebook of Parallel Computing. Beijing: Publishing House of Electronics Industry, 2005(in Chinese)
(莫则尧等译, 李晓梅校. 并行计算综论. 北京: 电子工业出版社, 2005)

CHEN Guo-Liang, born in 1938, professor, Ph. D. supervisor, member of the Chinese Academy of Sciences. His research interests include parallel computing model, parallel algorithm.

SUN Guang-Zhong, born in 1978, Ph. D., lecturer. His research interests include parallel computing, combination optimization.

XU Yun, born in 1960, Ph. D., associate professor. His research interests include parallel algorithm, bioinformatics.

LU Min, born in 1977, Ph. D., lecturer. Her research interests include parallel algorithm, combination and graph theory.