计算机科学1993Vol.20№.3

并行计算模型的分类和评价*)

倪德明 黄林鹏 孙永强
(上海交通大学计算机系 上海200030)

摘 要

随着各种并行计算机体系结构的出现及并行计算在各学科领域中的广泛应用，需要并行计算模型用以研究并行计算性质，分析并行算法，简化并行软件设计。本文对并行计算模型给出一个评价准则，并以此对现有的并行计算模型进行分类和比较。

1. 引论

在并行计算机上进行程序设计，突破了人为的顺序约束，能有效地提高程序执行的效率，更自然地表达和处理所处理问题中固有的并行性，但相应地也带来了很多问题：程序的可移植性差，难以准确地进行算法评估，以及用户表达求解问题的复杂化。

我们知道，基于冯·诺依曼机的计算模型的研究不仅为顺序计算的探讨打下了基础，而且为并行计算的研究也提供了一个合适的理论和实用框架。

迄今已有很多种并行计算模型，本文拟对一些现存的模型作一个理论上的分类和比较，并针对并行计算所产生的问题，对每一个讨论的并行计算模型，将考虑以下一些因素：

体系结构独立性。现存的大部分并行软件基于特定的体系结构或特定的机器，同时新的并行机仍在不断的发展中，软件的可移植性差，在软件生存期内，不能利用新的体系结构的优点。基于一般并行计算模型的软件能够不需要完全重建地在不同的体系结构上实现，因此，独立于体系结构与否是并行计算模型的一个重要的评价标准。

一致性。基于模型的效率分析应当与实际计算中的效率相一致。特别地，应当能够对并行算法的优劣作出效率上的评判。一致性要求的困难在于哪一层次的效率可以在模型的层次上体现出来。由于实现中必须处理不同的存储器访问延迟及通讯延迟，因此完全的一致是不可能的，除非是跟特定的机器结构非常接近的模型。合理的效率衡量标准应当是计算所需的总时间，即处理器数与执行时间的积，同时忽略执行中对每个程序基

Amsterdam, 1991

[5] Peter Jackson, On the semantics of counterfactuals, proceedings of IJCAI-89, Detriot, Michigan, USA, 1989

[6] Hirofumi Katsuno, Albert Mendelzon, On the difference between updating a knowledge base and revising it, in: Proceedings of the second internal conference in principles of knowledge representation and reasoning, James Allen, et al., eds., 1991

[7] Winslett, Marrianne, Reasoning about action using a possible models approach, in: Proceedings of AAAI-88, 89-93, St. Paul, minnesota, 1988

倪德明 博士生. 黄林鹏 博士. 孙永强 教授. 博士导师. *) 上海交通大学青年发展基金资助课题.

本上不变的时间开销。也就是说，一个模型是一致的，是指用p个处理器执行时间t的计算，应当用p′个处理机执行时间t′，满足

$$t' \cdot p' = O(t \cdot p)$$

在以下的讨论中，我们在紧耦合MIMD、类超立方体、常量价MIMD上对模型考虑一致性。

刻画简洁性。这是一个实用性要求以简化用户表达求解问题的复杂性，即模型必须对程序设计者隐藏三个问题：1)并行任务的分解或显式表示；2）通讯细节；3）同步细节。

以上对模型的要求往往是矛盾的，还有一些性质，如模型的表达能力、表达的相容性与完全性。限于篇幅，我们拟在另文中讨论。

2. 不限制计算的模型

2.1 图归约模型与OBJ及其派生模型

这两种模型都基于并发重写，计算的过程是寻找归约式归约归约式的过程，互不相关的归约式可以独立和并发地归约，因此动态地发现并行性。通讯与同步也不需要显式刻画，基本上独立于体系结构。

图归约：对一些复杂的难以线性化的结构操作，基于此模型的语言有高阶函数式语言的风格。图归约模型在何种程度上保持一致性很难进行探讨，由于函数式语言实现中，保持语义相等的程序转换能极大地改变程序执行中的开销，因此，连一致性在图归约模型中究竟指的是什么，也很难下定义，即使我们假定有一个优化的转换系统并且对函数而不是特定形式的计算定义执行开销时也如此。基于图归约的语言已经在紧耦合的MIMD和松耦合的MIMD上得到实现。

OBJ及其派生模型：OBJ是具有方程逻辑语义的函数式语言。OBJ语言程序中含有两种模块：理论和对象，以及一种复杂的模块继承机制。(对象的)计算机（理论的）演绎均基于并发重写。OBJ的派生模型主要是对逻辑式语言的实现提供基础，其中相应的语言有：加上面向对象能力的FOOPS，加上逻辑式程序设计的Eqlog，以及两者均有的FOOPlog。所有这些语言的语义奠基在可安全地并发执行的重写规则上，与图归约模型的并行归约相差无几，因此，OBJ及其派生模型也不具备一致性。

2.2 UNITY模型及Action系统

UNITY既作为一种计算策略，也作为一种程序设计语言。一个UNITY程序由一连串带哨（Guard）语句组成，程序的执行就是不确定但公平地选择一个语句。检哨，若成立则执行语句的过程。一个UNITY程序的执行永不终止，但只要能达到一个不动点即可。交插语义保证一些语句能并行执行而不改变程序的意义。基于UNITY的并行实现或UNITY语言的实现存在的问题有：选择语句难以公平；不动点检测困难而且复杂；不能有效率地检哨。因此UNITY模型刻画简洁，但不太可能保持一致性，如何独立于体系结构仍在探索之中。

Action系统基于与UNITY模型相近的观点，保留了UNITY的抽象能力，但通过变元的存放减少了在松耦合MIMD上现实的通讯，与UNITY相比，更具有一致性。

2.3 数据流模型和Linda模型

这两种模型都要求通讯的显式化，但同步是隐式的。

数据流模型与图归约模型一样是处理函数式语言的。数据流模型通过共享变元，使通讯显式化，在一阶上处理函数式语言，并行和同步在模型上仍然不可见。

数据流模型已在常规的松耦合机上实现。程序执行时，其它流上的指令能够插在两个相邻的指令间执行，掩盖了存储器访问延迟，这是数据流模型比图归约模型优越的地方；图归约模型的自然存储器组织是紧密耦合的，在松耦合体系上的实现比较困难；数据流模型的自然存储器组织是松耦合的，可以直接地在简单耦合体系上实现。

综上所述，数据流模型相对地独立于体

系结构；刻画比较简洁；使用点火规则，自动实现同步，通讯通过使用变量名实现，虽然是可见的，但不难处理和使用；数据流模型的另一优点是能够利用计算中所有的并行性；由于指令可以插入执行，而且这种插入是动态的，因此不能保持一致性。

Linda定义了一种能被任何进程同样访问的、按内容编址的、抽象的共享存储器，称为组空间(tuple space)，存储的对象称为组 (tuple)，组类似于记录的一系列值的集合，可以通过操作原语in，read,out,eval增加，读出，删除组中的任一值，组无地址，仅通过内容访问。

Linda的实现只要求在不同体系结构上实现组空间，因而具备体系结构独立性，但访问该空间的开销不可预测，不能保持一致性。Linda要求显式表示并行性，通过访问组空间显式通讯，但同步仍然是隐式的。

2.4 Occam

基于紧耦合多处理机系统的模型，可以简单地表示并行性，用共享存储器通讯，通过提供一些操作原语（如NYU Ultracomputer中的fetch&op)，用临界区保持技术控制对共享存储器的同步访问，这些原语类似于PRAM扩充模型中的Scan和Multiprefix。请参见本文第四节中的讨论。

Occam是严格按照C.A.R.Hoare的CSP理论发展起的并行程序设计语言，并在Transputer上得到实现。Occam基于并行执行的概念设计，提供并发进程之间的通信和自动同步机制。并行结构（PAR）将可并行执行的部分组成一个结构，运行时这几个部分可并发执行。并通过同步通信相互制约，在Transputer上实现时，大量的并行执行能够掩盖通讯延迟。

Occam模型具有一致性，但显然不独立于体系结构，并行、通讯、同步也必须显式定义。

3. 并行数据模型

并行数据模型只使用简单的控制流，如顺序或函数复合。每个操作对大量的数据施用共同的子操作，含有大量的SIMD型的并行性。与控制流模型相比，根本的差别在于控制流模型能直接处理动态行为而并行数据模型必须将它们表示成静态的形式。

已有一些模型成功地处理了表或向量类型，如Scanvector模型，Paralation模型及在Connection Machine上实现的相应语言，以及独立于体系结构的中间语言Vcode；对表或向量的处理比数组容易。

对数组类型的处理主要考虑其在SIMD机上实现，通过调整相互作用的数组减少数据在处理器间的移动。如Fortran 90的编译中有大量的工作分析程序以确定最好的调整，Prins指出了怎样把数据索引空间转换到机器索引空间，从而把数组的结构映射到处理器拓扑上。

为使数组在不同目标机上实现，理论方面所做的工作主要是发展一套关于数组的等式理论，以便于程序转换，同时也为更好地利用处理器拓扑提供指导。

Bird-Meertens形式化方法抓住了以上各种数据结构的构造性特点，为许多互相关联的数据类型提供了一个统一的理论基础。一个Bird-Meertens理论含有一些基本类型及一些类型构造子，Malcolm及Spivey提供了一种构造方法可以保证构造出的类型有许多有用的性质。该构造本身广泛地使用了范畴理论，构造的类型称为范畴数据类型。优点在于：

- 新类型是类型构造子的象，因此基本类型上的函数可以提升为新类型上的多态（polymorphie）函数。
- 自动产生大量的构造子之间、构造子与新类型及新类型上的新操作之间的关系等式，利于程序的开发和转换。例如可以如下的方式进行程序演绎：先写出满足规范的程序，再利用这些等式将它转换为高效率的程序，这种开法方式不仅能自动保证程序的正确性，而且不需要大量的开发并行程序的经

数，具有很大的实用价值；假如编译能够确定何种形式的计算在目标机上执行效率高，则可以用这些等式将程序转换为这种特定形式的计算，程序优化不再是一门艺术，成为一种理论的形式。重要的是，这些等式构成的集合是完全的，类型的任何性质能够表示出来。因此任何等价的计算形式能够互相推导出来，程序开发是可逆的，开发者不会走入死胡同。同时类型上的操作集也是完全的，不会丢掉任何有价值的操作，没有凭经验的开发方法的缺点。

·新类型上有一个一般化的射操作，把同一个操作施用到类型的所有元素，允许这种基本形式的并行直接地表示出来。

·对自由类型，可以定义一个一般化的归约，该归约首先计算操作对基本元素作用的结果，再递归地计算更大的增量。它抓住了并行数据模型上递归计算的特点，并把它表示成有大量并行性的形式。

·类型上的同态是保持类型结构的射。对自由范畴数据类型来说，同态可以表示成一般化射和一般化归约的复合。这样每个同态的实现可以分成高度并行化的两步：先并行地把函数作用到自变数（argument）的每个元素上，再施行一般化归约操作。

所有以上的并行数据方法，均含有一些内在操作，可以在多样的体系结构上实现，假如操作原语选择得当，并行数据模型可以保持一致性，因为所有的并行形式都压缩到数据类型中，模型只有单线控制，用户不需要显式处理并行、通讯与同步，所以刻画也是洁简的。

4. PRAM模型及其扩充

PRAM模型是最常用的并行计算模型之一。大多数并行算法都是基于PRAM模型设计的。本节我们首先讨论一般的PRAM模型，分析其在紧耦合MIMD、超立方体体系结构上模拟计算的一致性。接着研究PRAM模型的扩充及思想，最后介绍基于控制结构的两个并行计算模型：构架模型和P^3L模型。

4.1 PRAM模型

PRAM是并行随机存取存储器的简写。一个PRAM是一个包含若干可带有局部存储器的处理机及一个大容量共享存储器的抽象机。在一个单位时间里，每台处理器能执行一个简单的计算，从共享存储器读入一个值或把一个值写入共享存储器。处理器之间的通讯是通过共享存储器方式实现的。PRAM模型给程序设计者提供了一个紧耦合多处理器的合适抽象。PRAM模型的不足之处是它不能反映现实机器对存储器存取的时间开销。考虑PRAM模型在一个紧耦合MIMD并行计算机上的实现，由于共享存储器经由网络开关和处理器相连，而网络开关的时间开销一般为处理器数目的对数，因此在模型上是单位时间的存储器存取在实际实现时是物理处理器数目的对数阶时间，但通过多路转换技术，我们可以证明PRAM在紧耦合MIMD及超立方体体系结构上的最优计算模拟仍是一致的。下面是使用多路转换技术证明的主要思想。

设对于一个大小为n的输入，PRAM模型上的一个并行算法使用了p(n)个处理器且花费t(n)个单位时间，则计算开销为p(n)·t(n)。现我们在一个具有p(n)/log p(n)个处理器的紧耦合MIMD机器上模拟上述PRAM计算。首先把PRAM处理器（虚拟处理器）分成p(n)/log p(n)，每组log p(n)个。使用虚拟处理器和物理处理器之间的多路转换技术，易见每个虚拟处理器上相邻指令的执行时间间隔是log p(n)，而利用这个时间段恰好可实现对共享存储器的存取。由于总的执行时间为t(n)·log p(n)，总的计算开销为p(n)/log p(n)·(t(n)·log p(n))=p(n)·t(n)，因此PRAM上的计算开销和其在紧耦合MIMD上的实现的计算开销是一致的。同样，我们可以说明由于带宽的限制在一个带常量价互连网络的松耦合MIMD上模拟PRAM计算，总的计算开销将增加一个对数因子。

事实上，一般的PRAM模型是独立于系统结构且在紧耦合MIMD上的计算开销是一致的。但PRAM模型不提供对并行计算、通讯和同步的充分抽象。

4.2 PRAM模型的扩充

由于具体实现时对数阶时间的存储器存取在PRAM模型中可被考虑为单位时间，这就激发了研究者去寻找那些在实际机器上实现为对数阶时间而在一个扩充的PRAM模型中能被处理为单位时间的计算。如果这样的计算在并行算法中能被广泛使用，通过上面介绍的方法，计算开销的一致性将保证程序执行效率的提高。

所有PRAM模型的扩充都只能在紧耦合MIMD上实现。我们知道在紧耦合MIMD上，处理器和共享存储器之间的互连开关网络是树状的，如果我们给树结点加上处理数据的功能，则存储器存取过程中的数据信息在通过开关时将被处理。一种最简单的扩充是在网络上合并对同一存储器单元的读要求为单一要求并在逆向通路上复制数据，这就是EWCR PRAM模型。通过要求开关结点对目标为同一位置的值作某些处理，并发读/写可同样被实现，这就是CRCW PRAM模型。由于并发写的语义不同将对应不同的开关结点操作，因此存在不同m对数据的处理方式：最简单的一种是放弃一个数据而传送另一个数据，另一种是使用某个结合算子把所有写入同一位置的值结合起来再写入。

PRAM模型的另外两种扩充是在计算中加入scan或multiprefix算子。给定一个结合算子⊕和一个元素为$[a_0,a_1,\cdots,a_n]$的表，scan$[a_0,a_1,\cdots,a_n]=[a_0,a_0\oplus a_1,\cdots,a_0\oplus a_1\oplus\cdots\oplus a_n]$。如果开关结点能处理结合运算⊕且i小于处理器数目的对数，则scan能被并发实现。multiprefix是scan的一般形式。假定有k个处理器$p_1,\cdots,p_k$引用一初值为a的变量A，处理器p_i计算MP(A, v_i, ⊕)产生值$a\oplus v_1\oplus\cdots\oplus v_i$。最后变量A以$a\oplus v_1\oplus\cdots\oplus v_k$结束。Ranade证明了这样的计算将在log p(n)内终止(p(n)为处理器数目)。程序设计者将通过使用这些新增算子提高程序的效率。由于这些算子在现存的体系结构上易于实现，因此扩充的模型仍是独立于系统结构。除了能更好地利用通讯网络的特性外，扩充模型并不改变计算在松耦合MIMD上实现的计算开销不一致性。

PRAM的其它典型扩充有XPRAM(或称块同步PRAM)，Kruskal等提出的YPRAM，Heywood等提出的HPRAM(或称层次PRAM)及Danelutto的VLIW-in-the-large模型。它们都是由研究PRAM模型在不同体系结构上模拟实现的理论工作中发展起来的，它们都要求PRAM上的计算具有特定的结构，这些结构使PRAM上的计算能在这些扩充的PRAM模型上高效实现。

4.3 控制结构模型

从PRAM模型的拓展，我们得到这样的结论：一个良好的并行计算模型应着眼于重复出现的计算并对其实现方式进行优化。基于这个观点，一些并行计算模型提供了全局指令或控制策略以方便并行计算或通讯、同步模式的管理。下面介绍的构架(skeleton)模型及Pisa并行程序设计语言正是这类模型的代表。

构架是Cole提出的一种可插进顺序指令代码的高层控制结构。构架是并行计算的模板，它反映了一类计算的全局结构特性。程序设计者使用构架进行工作，而实现者利用构架提供的通讯和计算模式给出程序(算法)的高效实现。

Pisa并行程序设计语言(P^3L)通过包含一个反映通讯和计算一般模式的进程构造子集合提供了类似构架的一种高层并行性的控制和管理。这些构造子有：

- 管道构造子：允许顺序进程以管道方式相连；
- 数据并行构造子：它对数据分解并分配给处理器集合，同时给处理器提供相同处理代码；

(转第55页)

含了对管理人员十分有价值的有关组织结构、商业目标和过程的信息，但是该信息仓仍然只能被系统开发的IS人员使用。将来的CASE环境应同时支持系统开发和信息传送。用户和管理人员可以根据存贮在信息仓中的元数据，取得关于所访问的应用程序数据一些前后关系。

4. 设计和选择一个集成化CASE环境的关键之处在于在集成化和灵活性间取得平衡。通常，集成化程度越高，开放性就越差。CASE外壳(也称为元系统、CASE工具生成器和元CASE) 是一类裸CASE环境，允许CASE开发者和用户根据其新的或专用的方法制定CASE工具和环境。目前已有的CASE外壳有CADWare的Foundry，Intersolv的XL/Customizer，以及Systematica的Virtual Software Factory。

5. 重用技术是提高生产率的最有效途径之一。使用可重用构件，不仅能减少开发所需的费用，而且能提高开发速度和产品质量。重用的概念包括代码重用，以至更高层的规范和过程的重用。面向对象的技术为创建可重用构件提供了有效的机制、继承和封装。人工智能技术，如类比推理和情景推理也有助于确认和选择重用构件。

6. CASE标准在开发开放的CASE系统中起着重要的作用。集成化CASE环境应以这些标准为基础。但是，目前许多CASE标准交叉重叠，甚至互相冲突。如果我们要实现切实有用的CASE标准，就必须协调各形式化标准组织，取得用户和厂家的支持，并根据其它相关的标准开发CASE标准。

7. 技术转换和组织行为的学习是与集成化CASE有关的另一个必要的研究领域。

8. 最后，AI技术使得集成化CASE环境能包含具体域的知识，以帮助终端用户可以用高级语言或图形工具来开发和维护自己的系统。最终，使得终端用户得以检索或购买高层可重用模型，并加以修改，然后将其插入一个集成化CASE环境，生成他们自己的应用程序。

(接第17页)

• syntolic构造子：它把任意的一个超平面进程结构规定为脉动阵列；

• 树构造子：形成静态或动态进程树；

• 一般并行构造子：允许不相干的进程能相互独立地并行执行。

控制结构模型独立于系统结构并且是描述简单的，但计算开销一致性则取决于构造子或控制原语的选择。

5. 结束语

上面我们指出对一个并行计算模型的基本要求有：系统结构独立性，使模型上开发的软件易于移植，能充分利用物理机器的性质；计算开销一致性，能在模型层次上分析并行算法的复杂性；描述简洁性，便于程序设计者开发并行软件。我们认为，为了使并行计算能得到广泛应用，必须把软件工程、语言设计、智能编译和系统结构设计作为一个整体加以考虑。

参考文献

[1] D.B.Skillcorn, Architecture-Independent Parallel Computation, IEEE Computer 23(12):38-51, 1990

[2] D.B.Skillcorn, Models for Practical Parallel Computation, Inter, Jour. of Parallel Programming, Vol20, No.2, 1991

[3] W.F.Mcloll, Parallel Algorithms and Architectures, LNCS, 384, 1-22,

[4] L.G.Valiant, A Bridging Model for Parallel Computation, CACM 33 (8): 103-111, 1990

[5] S.Peyton-Jones, Implementation of Functional Programming Languages, Prentice-Hall, 1987

[6] K. M. Chandy & J. Misra, Parallel Program Design:A Foundation, Addison-Wesley, 1988