并行程序设计语言发展现状

韩 卫 郝红宇 代 丽
(燕山大学信息科学与工程学院 秦皇岛066004)

Current Development of Parallel Programming Language

HAN Wei HAO Hong-Yu DAI Li
(Department of Information Science and Engineering, Yanshan University, Qinhuandao 066004)

Abstract In this paper we introduce the history of the parallel programming language and list some of currently parallel programming languages. Then according to the classified principle. We analyze some of the representative parallel programming languages in detail. Finally, we show a further feature to the parallel programming language.

Keyword Parallel programming language

1 前言

随着计算机并行技术在科学、能源、医学、军事、人工智能和基础研究等许多领域的应用,常常要求计算机系统能够进行高效的并行处理。因此,就对传统的程序设计语言提出了新的要求,要求使用一种适应并行处理技术的并行程序设计语言,来方便程序员编写出执行速度更快、效率更高的并行程序。因此,并行程序设计语言是需要具有灵活性和有效性的高级程序设计语言。灵活性是指这种语言能使程序员在应用程序中描述各种形式的并行性;有效性即效率,是指语言可在各种并行/向量计算机系统中高效地实现。

到目前为止,伴随并行机的发展,并行程序设计语言的研究和发展已有十几年的历史。解决并行程序设计语言问题的技术和方法,概括起来有以下三种。

1.1 新语言方案

这种方案要求设计全新的并行程序设计语言来支持并行处理。近几年来推出的并行语言有:Occam、Ada、Linda 等。

新语言方案是与传统的串行程序设计语言完全不同的设计思想,且没有较成熟的设计方法,目前主要是针对特定的并行机而设计的并行程序设计语言。因此,新语言方案还很不成熟,由于它包含某些面向应用的并行结构成分,因此实现也较难。另外,这些语言中的每一种通常只支持一种形式的并行性,所以这些语言中尚无一种在商品化的并行机中受到普遍的欢迎。

新语言的另一个缺点是,程序员只能在多机上使用新语言去编写并行程序,而且由于并行性隐含着不确定性,使得并行程序的调试和正确性证明变得十分困难。

1.2 扩充语言方案

这种方案是在普通的顺序语言(或串行语言)中增加明确表示并行进程的部分,在语法上增加新的数据类型及相关操作,扩大描述问题的范围;在语义上扩展原操作符、操作对象范围和表达式语句的含义。例如用 FORK 和 JOIN 语句表示并行进程的派生和汇合。需要引入显示同步通信机制,如信号量、条件临界区、标记、管理路径表达式等。它刻画语句操作步骤间的并行性,其同步通信设施隐含于语句的结构中。也就是说通过面向系统结构的语言成分来扩充顺序语言,以便于并行性的充分发挥。

扩充语言的另一个优点是适合人们的传统的编程习惯。几十年来人们编程的思维方法总是串行的,要改变是相当困难的。但是这样的方法有一缺点:通常一种扩充语言只能支持一种并行性。由于并行性是面向并行机的,因此效率很高。但是对机器系统结构依赖性很大,从而可移植性很差。当目标机有所改变时,用户不得不用另一种程序设计语言来重新编写并行算法的程序。

1.3 顺序程序向并行程序的自动转换

大多数现有的应用软件都用像 FORTRAN 这一类顺序语言编写,为了能在顺序程序中检测出并行性,并将其转换成并行程序段或并行机器代码,这就需要设计出实现顺序程序向并行程序转换的自动转换程序。这种程序称之为智能编译程序(Intelligent Compiler)。采用这种方法不仅允许程序员仍用传统的编程方法去编程,而且长期积累下来的用顺序程序设计语言编写的程序,只需略加修改就可以用于并行计算机上。

使用这一方案,虽有上述优点但是由于使用顺序语言,迫使程序员不得不以顺序代码形式编写并行算法的程序。在一个复杂的程序中要检测出并行性是很困难的,而且编译程序也只能在有限的程度上改善性能。

1.4 三种方案的比较

首先扩充语言和新语言称为显示并行技术,顺序程序向并行程序的自动转换是隐式并行技术。一方面并行算法和并行计算机表现了多种计算模式;另一方面,上面提到的每一种语言都只能支持一种或两种计算模式,当用传统的顺序语言在并行机上编写并行算法程序时,上述矛盾就是产生不灵活性和低效的原因。

2 当前并行程序设计语言

从并行计算的分类上看并行计算可以分为数据并行计算和任务并行计算两种。在数据并行的程序设计中,程序可以理解为把由同样的操作同时施加到一个大数据结构的所有或大多数元素上的一个操作序列组成,即同一操作可以并行地施加到许多数据项上同时进行。而在任务并行程序中,程序由一组并行任务组成,这些并行任务通过显示通讯与同步相互交互。

面向对象技术已经渗入到了包括科学计算在内的各个领

韩 卫 博士生,副教授,研究领域为专家系统开发与应用、数据库系统开发与应用。代 丽、郝红宇 硕士研究生。

域。其对数据与计算的封装性、消息传递的透明性，使得程序员能够把更多的精力放在算法的设计上；另一方面，工作站网络为人们提供了一个廉价的超级计算硬件平台。将二者结合起来，既能简化用户的编程又能充分发挥工作站网络的计算能力。

针对并行计算的分类和目前比较成熟的面向对象技术，其中对最具有代表性的面向对象程序设计语言 C++的并行化的研究已经成为国内外同行们研究的一个热点。下面介绍几种有代表性的并行程序设计语言。

2.1 Compositional C++：支持对象并行的程序设计语言

许多编写并行程序的人员都抱怨，并行程序结构的依赖性、令人混淆的表示方法、正确性确认的困难以及沉重的负载开销使得并行程序设计显得乏味枯燥，并阻碍了并行程序设计的实用性，使得人们忽略了并行程序设计的许多潜在好处。

Compositional C++[1]（简称 CC++）引入了任务并行机制，即采用 MPMD 的执行方式实现不同任务在不同结点机上的并发执行。CC++通过在顺序 C++语言上进行一些简单的扩充，减轻了并行程序设计带来的困难。CC++语言是 C++语言的严格超集，所以任何有效的 C 或 C++程序也就是一个有效的 CC++程序。该语言与 C 及 C++语言的兼容性方便了程序员向并行程序设计的转变。在 CC++中保存了 C++的所有面向对象的特征。这些特征，特别是类属性与继承机制，促进了代码的重用、结构化及对参量的确认。

CC++语言是由 Caltech 的组合系统研究组开发的，它对 C++语言扩充了8种结构：par、parfor、spawn、atomic、global、processor object 和 data transfer function。这8种结构又可分成两大部分，其中 par、parfor、spawn、atomic 及 sync 为单地址空间结构，而 global、processor objects 和 data transfer function 为多地址空间结构。在 C++中围绕着对象实现了多级别的并行执行，它简单易学，指令丰富，适合在异构分布环境下求解各种确定问题和不确定问题。尽管这些扩充看起来较简单，数目也少，但是它们与 C++结合起来，就形成了一个十分丰富且有力的并行程序设计方法。

2.1.1 CC++的指令扩展

(1)par 并行语句块。一个并行块的结构如下所示：

par{statement1;statement2;…;statement n}

其中每个 statement 允许是一个语句或语句块。语句块 statement1，statement2，…，statement n 分别以线索为单位并行执行，最后 n 个线索于一点处汇齐，然后再执行 par 的下一条指令。

(2)parfor 并行循环。一个并行循环的结构如下所示：

parfor(int i=0;i<N;i++){statement1;statement2;…;statement n}

在并行循环结构中，其迭代并行地执行，而每个迭代体均与平常的 C 或 C++一样地顺序执行。在每个 parfor 的结尾处均有一个隐式的阻挡层，仅当所有迭代均完成后，parfor 语句方可结束。在一个 parfor 语句中使用的循环控制变量必须经过特别处理，此变量必须在 parfor 内部进行声明，每一次迭代都处理该变量的一个常量拷贝，循环控制变量在 parfor 体内不可以修改。

(3)spawn 派生函数。spawn 语句用来创建以并发方式执行的控制完全独立的线程。与结构化并行语句不同，在派生线程之间并没有父子关系，两者之间无阻挡层或任何其它形式的隐式同步。仅有函数可以派生，且派生函数不能返回结果。Spawn 保证派生的函数到参量在派生线程执行下一指令之前被拷贝。

(4)处理机对象。是一种特殊的对象类，用于实现可在多机并行执行的对象。处理机对象的声明是在 C++类定义前用 global 标识；处理机对象到处理机的映射是由一个专门的类实现的，首先将处理机对象所在的可执行文件指派到物理处理机上，然后由底层的通讯机制驱动对象内的成员函数的访问。

(5)sync 同步变量。又称单赋值变量，保证对公共变量并发访问的有序性。单赋值变量只允许赋值一次，赋值后相当于常量；赋值前所有试图读它的线索都被阻塞，处于等待状态，直到变量赋值才唤醒。sync 变量实现较简单，由三部分组成：变量的值，赋值标志和一个 Nexus 条件变量（用于唤醒等待 sync 变量的线索）。

(6)atomic 函数。又称原子函数，提供了一个控制线索调度的方法，保证对函数并发访问的有序性。原子函数在同一时间内，允许一个线索调用，其它线索处于等待队列。atomic 通过下层 runtime 库提供的互斥变量实现。

2.1.2 CC++的基本特点

在运行时系统 Nexus 的基础上，CC++源程序能有效地运行于各种异构分布环境，包括松散耦合的网络环境，如 LAN、工作站群机网络；共享存储 MIMD 多处理机，如 SGI IRIX；分布存储 MIMD 多处理机，如 Intel Paragon、IBM SP2等。这是因为，一方面 CC++采用了不依赖于具体的消息传递或共享存储器的抽象进程同步机制，从而无需修改 CC++源程序，就能直接在不同的系统平台上进行移植。另一方面，CC++提供了语言上对异构计算机的支持，即处理机对象。通过 CC++用户可以把不同的处理机对象显示地映射到适合的物理处理机上，保证了指令在异构网络上的高效实现。

CC++为用户提供了丰富的针对语句块、循环体和函数的并行程序设计范式，适用于诸多领域、不同粒度级的计算，从频繁同步的线索级的细粒度并行计算，到稀疏同步的分布的进程级的粗粒度并行计算。

并行的引入使 CC++具备了较为全面的处理能力，不但支持传统语言所能解决的确定性问题，而且支持结果不确定的问题的求解，可以应用到更为广泛的领域中。

2.2 pC++：分布式的对象并行语言

pC++[2]引入了数据并行机制，是一种 HPF 的并行思想在 C++中的并行实现。

pC++是一种可以运行于工作站网络（NOW）上面向对象并行编程语言，它提供了面向对象的并行编程接口——Collection。使得用户能够方便地在工作站网络上开发面向大规模计算问题的对象并行程序。Collection 是 pC++对 C++语言的新增关键字，它是一个形式上类似 C++中类的结构，利用此结构用户能够描述一组并发执行的对象集合。

(1)Collection 结构。Collection 语法结构定义如下：

```
Collection NameOfCollectionType :ParentCollection {
    private:
        〈私有数据成员和方法函数〉
    protected:
        〈保护数据成员和方法函数〉
    public:
        〈公有数据成员和方法函数〉
    MethodOfElement:
        〈添加到每个元素对象的数据和方法〉
}
```

不难看出，Collection 具有与类相似的私有、保护和公共域，而且还具有类的继承性、多态性等特点。它与类的主要不同之处在于类表示的是单独的对象个体，而 Collection 表示的是对象集合。Collection 的显著特色是它具有一个元素方法

域(MethodOfElement),这是针对对象集合中各个对象而设计的对类的扩充结构。凡是位于元素方法域中的数据成员和方法成员都将移到对象集合的每一个对象中去。通过元素方法域,程序编制人员可以灵活地对各对象进行控制。

(2)pC++的对象并行机制。在pC++中,并发对象集合向各处理机的映射是通过对准对象、分布对象和处理机对象来描述的。具体而言,处理机对象定义了处理机的拓扑结构,分布对象定义了一个抽象数据模板及模板与处理机之间的映射关系,对准对象定义了并发对象集合的数据结构与模板之间的映射关系。通过此二级映射机制,程序员可以将并发对象集合根据需要以适合的方式映射到模板上,并通过该模板分布到各处理机上。

为了描述各对象在处理机的分布位置,pC++提供了一个全局名空间表示方法,它主要是通过信息头(struct INFOHEAD)来实现。对象集合初始化时,系统在每个结点机上填写该结构,以反映各元素对象所处的位置。这样,尽管每个处理机结点只拥有数据的一部分,但它却能"看见"全部的数据。对于程序员而言,数据彷佛存放于一个连续的通过对象名访问的空间上,即所谓全局名空间。这样可以象在共享存储机上编程一样,通过对象名来访问对象,而不必考虑处理机间的通讯、同步等复杂的操作。

Collection 中的并发对象都是以 SPMD 方式执行的,各处理机包含部分对象集合,并且执行类似的操作。对象的并发执行主要是通过调用元素方法域的函数启动的,采用这种执行模式,用户可以根据实际问题的需要,在元素对象域中加入适当的方法,来并发执行各个对象,并可以对对象执行行为进行控制,这是其它对象并行语言所不具备的特点。

(3)pC++对象并行机制的库函数实现。pC++的对象并行机制主要是通过 Runtime 库的支持来实现的,pC++编译器只作了少量的工作,它主要是将 Collection 的元素方法域中成员移到各对象中,从而将 Collection 转换为类的形式。然后,通过调用 C++编译器来进行编译,并在运行时利用 Runtime 库的支持,就可以实现对象的并行执行。这种方法实现起来较简单,而且易于改进和扩充。Runtime 库的功能包括在各处理机上动态地为每个对象申请内存空间、远地对象的访问以及处理机间的同步等。

总之,pC++是一个面向对象的数据并行系统,它提供给用户一个简单的对象并行编程接口——Collection,其元素方法域使用户能够对并发对象进行较强的控制。同时它也存在着不足,如 Collection 维数较低、数据划分算法存在着缺陷等,针对这一不足需进行扩充和修正。

2.3 HPC++:一种支持多线索和远程对象调用的并行语言

高性能 C++(HPC++[3])协会是由来自大学、工业界和政府实验室的人士组成的,该协会一直为基于 C++语言并行编程而设计标准库,其目标是建立一个并行应用问题的公共系统。HPC++是该协会最近推出的一个新的独具特色的高性能并行语言,其思想来源于 K.Mani Changdy 和 Carl Kesselman 的 CC++语言;Yutaka Ishikawa 设计的 MPC++多线程模板库;IBM 的 ABC++库;对象管理研究组的 COBRA 说明和 Java 并发模型。

HPC++的主要特色是:(1)一个 Java 风格的线索库,提供了编写共享存储体系结构上的并行应用程序的办法;(2)一个模板库,它支持同步、集合等并行操作,以及远程访问存储器;(3)一个 COBRA 风格的 IDL 到代理的产生器,它支持成员函数调用远程地址空间上的对象。

HPC++设计包括两级:第一级包含不对 C++进行任何扩展情况下的一系列库类和工具的说明。第一级库包括一系列简单循环的编译指导,用它们控制做一个单上下文内的并行。第二级提供基本语言扩展和运行库,用来充分实现第一级说明。主要是由并行标准模板库(PSTL)组成,如果上下文运行于一个以上的结点时就可以使用 PSTL 或派生新的控制线索获得并行。

HPC++采用了一个基于线索类的模型,它与 Java 的线索系统相类似。Thread 和 Runnable 两个类能够在一个上下文内,初始化线索并令线索运行。基本 Thread 对象封装了一个线索,并提供了私有地址空间;Runnable 类的对象提供了使多个线索执行一个共享的成员函数的方法。

在 HPC++中,为了实现多上下文编程,就需要实现一个上下文能够调用另一个上下文的对象成员函数。因此,HPC++给用户依次提供了全局函数、全局指针和用 CORBA 产生的 IDL 代理来实现这个功能。

这种语言能够更好地挖掘程序的并行性,提高解决应用问题的效率,并且在这种语言中引入了当今流行的 Java 和 COBRA 等新技术,给并行语言的研究提供了新的思路。

2.4 SPPL:一种简单的并行程序设计语言

目前很难找出一种有效的方法将串行程序自动地并行化,因此,为标准的串行语言提供并行的提示是很自然的想法。SPPL 就是基于提示的并行语言,并尽可能保留标准串行语言的清晰度。但是,这些提示应当与机器无关,以保证并行程序的可移植性。自数据并行性中,并行计算的概念有 SIMD (Single Instruction Multi-Data)与 SPMD (Single Program Multi Data);而在任务并行性中,并行计算的概念有 MPSD (Multi Program Single Data)与 MPMD (Multi Program Single Data)。这些概念虽然也用于机器分类,但是与机器的具体组织无关,它们主要反映的是数据并行还是任务并行的并行计算方式,应当包含在并行语言的提示中。这些概念的确能提高并行程序的可理解性与可编译性,在 SPPL 语言中就是按这种思想选择提示的。

SPPL 语言只有五种提示(并行系统的提示、并行计算方式的提示、资源分配的提示、数据分解的提示和数据管理的提示),每种提示都是以 C$ 开头。编译时忽略提示,则 SPPL 语言的并行程序就自动成为串行语言的源程序,可在串行计算机上验证执行。

SPPL 基本上保持了标准串行程序的设计特色,因而使并行程序具有了与标准串行程序相近的可理解性。由于 SPPL 语言是采用提示方法描述并行计算的,因此,并行编译首先是把这些提示编译成标准串行语言的程序,它与非并行程序段(即并行程序中的串行程序)一起构成整个并行程序的标准串行语言的源程序。再经过串行编译程序,便得到机器(汇编)语言的目标程序。

2.5 其它并行程序设计语言

目前比较流行的并行程序设计语言还有:

(1) 基于 NOW 的对象式分布式程序设计语言 NC++。

(2) 基于消息传递到 MPC++并行语言系统:利用并行类库自动完成计算结点的分配、并行进程的加载、进程间的同步和通信等。

(3) 基于共享对象的 SOC++并行语言系统:采用了共享

对象模型简化了并行程序设计的难度，屏蔽了不同并行系统间的差异，还提供了队列、集合等工具，用以对并行程序设计提供高级支持。SOC++系统建立在PVM并行编程环境之上，完全用类库实现。

(4) 基于对象级并行的CCPP语言系统：CCPP是一个严格意义上的并行面向对象语言，在CCPP模型中，所有对象都是并行对象，对象间的消息传递都采用异步消息发送方式。系统提供对象间和对象内两级并行，并采用"惰性计算"模型进一步扩大并行计算的可能性，并简化并行进程间的同步。CCPP模型实现了串行/并行代码的相互重用，是一种简单、高效、易移植的并行面向对象程序设计语言模型，同时CCPP属细粒度并行语言，具有更大的理论意义。

结束语 本文根据设计并行程序语言的几个原则，介绍了几种有代表性的程序设计语言。它们为最终形成适合于描述并行系统、高效实用的并行程序设计语言做出了探索和尝试。为此，我们希望出现这样一种并行程序设计语言：这种语言易学，并在主要的应用中能灵活应用；这种语言应与系统结构无关，能适应各种计算模式SIMD、MIMD、流水线、数据流等并行处理系统。这样一种理想的语言大概必须把上述的各种语言方案结合起来开发，可能是从一个灵巧编译程序方法开始，对现有的通用语言加以某些扩充，然后作为一个长期目标逐步走向一种新语言的开发。

参考文献

1 Foster I. Compositional parallel programming languages. ACM. Transaction on Programming Language and Systems, 1996, 18(4)

2 Gannon D, Yang S X, Beckman P. User Guide for a Portable Parallel C ++ Programming Systems: pC ++. [R] Technical Report, Indiana University, Sep. 1994

3 Johnson E, Gannon D. Hpc ++ : Experiments with the parallel standard template library. [R]: [Technical Report TR-96-51]. Indiana University. Department of Computer Science, Dec. 1996

(上接第104页)

5 系统模块划分

从系统的角度分析，VoIP网关是PSTN与IP分组交换网间的接口，它必须要有相应的接口处理模块：PSTN接口模块与网络传输控制模块；另外，VoIP网关还要实现呼叫过程的呼叫请求、连接建立、呼叫维持、呼叫拆除等，我们用呼叫控制模块来完成这些功能；在VoIP网关中使用专门的语音处理模块可以降低每通话信道的数据流量，也是提高通话质量的关键；最后，网关系统管理模块将完成系统管理、用户接入认证、通信信道分配、呼出限制检查等功能，各模块间的相互关系如图4所示。

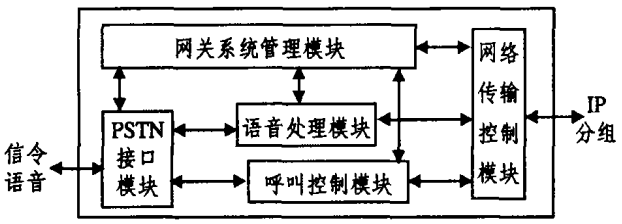

图4 VoIP网关主要功能模块及其相互关系

6 系统功能实现

◇ *接口功能* 系统提供三类接口：IP网络接口，用于连接符合IEEE802.3或IEEE802.3u的以太网；数字中继接口，通过PCM数字中继线(E1)与PSTN相连，接口信令采用中国一号信令MFC-R2(可以升级为支持中国七号信令)，可同时提供30-120路通话能力，满足大中型企业组织的需求；模拟电话接口，直接通过8-32路模拟电话线连接PSTN，可以满足中小组织的需要。

◇ *用户接入认证功能* 系统提供两种认证方法对主叫用户进行身份鉴别：主叫号码认证，根据主叫用户的电话号码来进行身份认证，该方式的主要优点是拨号简便；账号 + 密码认证，这种认证方式可以使合法用户在市内任何一部程控电话机上拨打IP电话，拓展了使用空间。

◇ *主叫用户分级机制* 将合法用户分为超级用户组和常规用户组，根据用户接入认证的具体方式，在合法主叫号码或者合法用户账号的属性中添加用户分级属性。对于不同级别的用户在信道占用、呼出限制等方面都有不同的权限。

◇ *语音处理功能* 语音信号的编解码(支持G.711、G.729a、G.723.1)；语音分组单元；静音抑制；回声消除；舒适噪音生成；抖动处理；语音信号的采集与播放等。为了提高网关的实时处理能力、减小时延，一般采用专用的DSP芯片来完成上述功能。

◇ *网络传输控制功能* 这部分主要是利用缓冲技术来实现语音数据包的正确发送和接收，并对收到的语音数据包进行乱序处理、丢包补偿处理以及网络时延处理，最大限度地保证本地语音数据的完整性和实时性，提高通话质量。

◇ *网关系统管理功能* 这部分主要包括如下子功能：系统初始化；系统参数设置，设置系统工作参数(如：指定用户接入权限认证的方式、设置缓冲深度等)；系统状态维护，状态监控、状态统计、故障报警等；计费服务，按照设定的费率计算每一个通话的费用并记录到相应用户账上。

结束语 随着业务的扩展，众多公司、企业在全国各地都设置了分支机构或办事处，有的还在国外设有分支机构，各机构间的相互联系导致长途电话费用的急剧增加；而另一方面，这些组织又通过租用电信线路(如DDN或ISDN)建有完善的企业网，并且网络带宽根本没有得到充分利用。VoIP网关就是针对这类组织而设计的，它利用组织内部的Intranet实现电话的长途转接，在不增加网络费用的情况下为组织节约大量的长途电话费，具有良好的实用价值。

另外，基于VoIP技术可以很容易对网关进行升级，使其可以同时传送语音和图像，实现可视电话、视频会议，即数据、语音、图像三网合一，其应用前景非常广阔。

参考文献

1 ITU-T Rec. H.323. Packet-based multimedia communications systems. Nov. 2000

2 Arango M, et al. Media Gateway Control Protocol (MGCP). IETF draft, Feb. 1999

3 糜正琨. IP网络电话技术. 北京：人民邮电出版社，2000

4 舒华英，赖平漳，等. IP电话技术及其应用. 北京：人民邮电出版社，1999