有限延时消息通信模型及在并行程序调试器设计中的应用*)

A Delay-Bounded Message Passing Model and its Applications in Distributed Debuggers on Cluster Systems

刘 建 沈美明 郑纬民

(清华大学计算机系高性能研究所 北京100084)

Abstract A cluster system is a distributed memory multi-computer system based on message passing. Message passing models are the basis for the researching of other technologies in cluster systems. First, we formally describe existing message passing models. Then, we suggest a DFIFO (Delay Bounded FIFO) model, which is more compliant with the actual cluster system than others. The applications of DFIFO model in designing of distributed debuggers are briefly described.

Keywords Distributed debugger, Cluster system, Message passing

1 概述

在分布存储多计算机系统中,消息传递是进程间唯一的通信手段,消息通信模型决定了并行计算的模型。一般情况下,通过对并行程序计算模型的描述来研究消息通信的模型。

分布存储多计算机系统中消息通信机制可以归为两类[1]:阻塞(Blocking)通信和非阻塞(Non-blocking)通信。阻塞通信又称作同步通信,其特点是发送方和接收方都就绪,通信才进行,如果有一方未准备好,另一方便进行等待;非阻塞通信又叫异步通信,特点是发送方不必等待接收方就绪便可将消息发出,然后接着进行后面的操作。异步通信需要引入缓冲机制,同步通信则可以没有缓冲。在异步通信中,接收依然是阻塞的,这是从接收操作完成的角度来说的,与一些系统上提供的非阻塞接收函数概念不一样。

根据消息通信方式的不同,基于消息通信的并行计算可以分成同步计算和异步计算两类。如果一个计算中的所有通信都是同步通信,那么这个计算是同步计算,支持同步计算的消息通信模型称作同步消息通信模型。如果一个计算中包含有异步通信,则该计算是异步计算,支持异步计算的消息通信模型称作异步消息通信模型。

并行程序的执行由若干个顺序进程构成,这些进程通过消息进行通信,每个进程有一个局部算法。从事件模型的角度来看,进程的执行是一个事件序列。这里的事件是进程的原子操作。通常,这些事件可以分成三种:发送事件、接收事件和内部事件。发送事件发出一个消息,接收事件接收一个消息,内部事件导致进程局部状态的变化。如果一个发送事件 s 发出的消息被一个接收事件 r 接收,则称 s 和 r 对应。

这样一个计算过程可以用时空图表示[2]。如图1所示时空图中,横线称作进程线,每一条进程线表示一个进程,线上的圆点表示事件,进程线之间的箭头表示消息传送,箭头的始端连接发送事件,末端连接对应接收事件。在同一条进程线上,左边的事件比右边的事件要先发生,从左向右表示时间的推移。

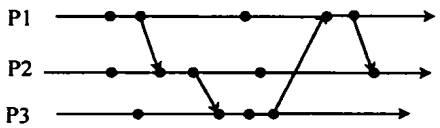

图1 并行计算的时空图

在时空图上,如果事件 e 和事件 e' 之间存在一条有向路径(沿着箭头和时间的方向)可以从 e 到达 e',说明事件 e 和 e' 之间存在一种相关性。这种相关性具有传递性,而且不可能形成环,因此它构成一个非自反偏序关系,记作"π",$e\pi e'$ 表示 e 在 e' 之前发生,或者说 e' 依赖于 e。如果事件 a 和 b 在同一个进程上发生,我们记作 a～b。

2 消息通信模型的形式化描述

下面,用形式化的方法描述几种并行程序计算模型及消息通信模型。

2.1 异步通信模型

机群系统并行程序的计算可以抽象成 n 个进程 $P_1, P_2, \cdots, P_n$,进程间只能通过消息进行通信,进程之间的相对速度以及消息传送的延迟时间都没有限制。这种计算由两部分组成:一是由每个进程的局部计算 C_i 构成的一个 n 元组 $C=(C_1, C_2, \cdots, C_n)$,二是构成计算的所有事件间的依赖关系。每个进程 P_i 相当于运行一个串行程序,其执行过程可以看作是一个有限的事件序列 C_i,这个序列叫做进程 P_i 上的局部计算,所有的局部计算 $C_1, C_2, \cdots, C_n$ 就构成整个并行计算 C。

给定计算 C,记 $\Gamma=\{(s,r)\in C_i\times C_j | s$ 与 r 对应$\}$,即 Γ 是所有对应的发送事件和接收事件对的集合。在这里,我们只讨论一对一通信,不考虑一对多及多对一的情况,并且假设通信都在不同的进程间进行。在一对一的情况下,一个消息通信事件最多只能有一个与其对应的消息接收事件,由于被接收的消息肯定已经发出,因此每个接收事件都有一个对应的发送事件。如果 C 中的每个发送事件都有对应的接收事件,则说明所有的消息都可以被接收,这样的计算称作是完全计算。

现在我们来给并行计算 $C=(C_1, C_2, \cdots, C_n)$ 上的关系依

*)本文的研究得到国家自然科学基项目 NO. 69933020的资助。刘 建 博士研究生,主要研究领域为并行程序调试环境。沈美明、郑纬民 博士导师,主要研究领域为并行/分布处理。

赖 π 下一个精确的定义。由于每个进程都是顺序执行的，因此每个局部计算 C_i 中的所有事件根据发生的时间存在一个全序关系，记作 π_i，它表示同一进程上早发生的事件对晚发生的事件有影响。显然 π_i 在 C_i 上构成一个非自反的全序关系。不同进程上的事件间的依赖关系是由于消息通信造成的。如果 $(s,r)\in\Gamma$，则接收事件 r 依赖于发送事件 s。因此，$\pi_1,\pi_2,\cdots,\pi_n$ 和 Γ 的传递作用就构成了 C 上的整个偏序关系 π。

定义1 计算 C 上的因果依赖关系(Causality Relation)π(是满足以下条件的最小集：(1)$a\pi_i b \Rightarrow a\pi b$；(2)$(s,r)\in\Gamma\Rightarrow s\pi r$；(3)$a\pi b \wedge b\pi c\Rightarrow a\pi c$。

对两个不同的事件 a 和 b，如果 $\neg(a\pi b)\wedge\neg(b\pi a)$，则我们说 a 和 b 是并发的，记作 a//b。

定义2 一个异步计算(Asynchronous Computation)是由局部计算元组 $C=(C_1,C_2,\cdots,C_n)$ 和 C 中消息通信事件对集合 Γ 以及 Γ 上的偏序关系 π 构成，支持异步计算的消息通信模型称作异步消息通信模型。

在定义2中集合 Γ 是至关重要的。我们来看一个似乎正确的定义：一个异步计算是由局部计算组成的元组 C，加上 C 中所有事件由 π 构成的偏序关系。这个定义乍一看没错，但是它没有包括 Γ 中的发送事件与接收事件的对应关系，因此有时不能区分两个不同的计算。图2就是一个例子，根据这个定义，图2(a)和图2(b)表示的是相同的计算，但是根据定义2便可以明确地将二者区分开来。

异步计算的"异步性"体现在 π 的定义上，定义1的条件(2)指出，接收事件依赖于对应的发送事件，但反过来不成立。这样，消息接收必须阻塞，因为它必须等发送完成后才能执行，消息发送不必阻塞，因为它不依赖于接收事件。

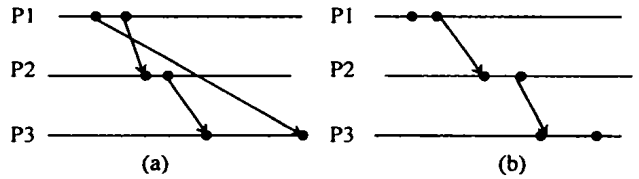

图2 两个不同的并行计算

2.2 加入顺序限制的异步通信模型

实际的并行计算不象定义2中的那么随意，而是受到一些限制。最常见的一种限制条件是：在同一个通道上(限特定的两个进程之间)进行通信时，消息被接收的顺序与消息被发出的顺序一致，即消息通道具有 FIFO(先进先出)的特性。由此我们给出 FIFO 消息通信的定义。

定义3 计算 C 中，如果对所有的 (s,r) 和 $(s',r')\in\Gamma$，满足：$s\sim s'\wedge r\sim r'\wedge s\pi s'\Rightarrow r\pi r'$，则我们称 C 是一个通道有序的计算(先进先出计算)，简称 FIFO 计算，支持 FIFO 计算的消息通信模型称作 FIFO 消息通信模型。

如果去掉定义3中的 $s\sim s'$ 前提，我们可以得到一个更强的限制条件：所有发往同一个进程的消息的接收顺序与对应发送事件的因果顺序一致，如果两个发送事件是并发的，则对应的消息接收事件的顺序则是任意的。这种性质叫做因果有序(Causally Ordered)。

定义4 计算 C 中，如果对 $\forall(s,r),(s',r')\in\Gamma$，满足：$r\sim r'\wedge s\pi s'\Rightarrow r\pi r'$，则称 C 是一个因果有序的计算(Causally Ordered Computation)，简称 CO 计算，支持 CO 计算的消息通信模型称作 CO 消息通信模型。

CO 计算有几个很好的性质，讨论如下。

定理1 计算 C 是 CO 计算当且仅当对 $\forall(s,r),(s',r')\in\Gamma$，都有：$s\pi s'\Rightarrow\neg(r'\pi r)$。

证明：⇒如果 C 是 CO 计算，假设存在 $(s,r),(s',r')\in\Gamma$，使 $s\pi s'\Rightarrow r'\pi r$，根据 π 的定义可知存在一个事件链 $a_0\pi a_1\pi a_2\pi\cdots\pi a_k$，其中 $a_0=r'$，$a_k=r$，对每个 $j<k$ 有 $a_j\pi_i a_{j+1}$ 或 $(a_j,a_{j+1})\in\Gamma$。(1)若是 $a_j\pi_i a_{j+1}$，则说明 a_0 到 a_k 都属于同一进程，即 $r'\sim r$，因为有 $s\pi s'$，由于 C 是 CO 计算，从而 $r\pi r'$，与假设矛盾。(2)若是 $(a_j,a_{j+1})\in\Gamma$ 的情况，取链中最后一对满足该情况的事件，记为 (σ,ρ)，显然 $r'\pi\sigma,\rho\sim r$ 且 $\rho\underline{\pi}r$，从 $s\pi s',(s',r')\in\Gamma$，知 $s\pi r'$；又因为 $r'\pi\sigma$，故可得 $s\pi\sigma$；由于 C 是 CO 计算，因此由 $s\pi\sigma$，$(\sigma,\rho)\in\Gamma$ 和 $(s,r)\in\Gamma$ 知 $r\pi\rho$，与 $\rho\underline{\pi}r$ 矛盾。由(1)(2)可得假设不成立。故结论正确。

⇐如果计算 C 满足 $\forall(s,r),(s',r')\in\Gamma$，都有：$s\pi s'\Rightarrow\neg(r'\pi r)$，此时 $r\sim r'\wedge s\pi s'\Rightarrow r\sim r'\wedge\neg(r'\pi r)\Rightarrow(r\pi r'\vee r'\pi r)\wedge\neg(r'\pi r)\Rightarrow r\pi r'$，因此 C 是 CO 计算。

从定义4中我们只能看出，对于 CO 计算到达同一进程的所有消息是按因果依赖关系排序的，而定理1则进一步说明对于 CO 计算，所有的消息传送都是遵循因果依赖关系的。

定理2 计算 C 是 CO 计算当且仅当对 $\forall(s,r)\in\Gamma$，集合 $A=\{x\in C\mid s\pi x\pi r\}$ 是空集。

证明：(用反证法) ⇒假设存在 $(s,r)\in\Gamma$，A 非空；设 $x\in A$，由于 $\neg(s\sim r)$，所以有 $\neg(x\sim s)$ 或 $\neg(x\sim r)$，两种情况下都存在一个链 $a_0\pi\cdots\pi x\pi\cdots\pi a_k$，其中 $a_0=s$，$a_k=r$，该链中至少存在两个相邻的事件 $(\sigma,\rho)\in\Gamma$ 且 $\sigma\neq s\wedge\rho\neq r$ 且 $s\pi\sigma\wedge\rho\pi r$，因此不满足 $s\pi\sigma\Rightarrow\neg(\rho\pi r)$ 的条件，由定理1可知 C 不是 CO 计算。

⇐如果 C 不是 CO 计算，则存在 $(s,r),(s',r')\in\Gamma$，$s\pi s'\wedge r\sim r'\wedge r'\pi r$，由于 $s'\pi r'$，因此有 $s\pi s'\pi r'\pi r$，从而 A 非空。

定理2说明，在 CO 计算中，对应的发送事件和接收事件在因果关系链中是相邻的。这样就不可能出现图2(a)中的那种情况。

性质1 计算 C 是 CO 计算当且仅当对 $\forall(s,r)\in\Gamma,\forall x\in C$ 有：(1)$x\pi r\Rightarrow\neg(s\pi x)$，(2)$s\pi x\Rightarrow\neg(x\pi r)$。

性质1说明，在 CO 计算中，先于接收事件发生的事件不会晚于对应的发送事件(可能与发送事件并发)，晚于发送事件发生的事件也不会先于对应的接收事件(可能与接收事件并发)。

CO 计算中，一对发送和接收事件，中间不会插入与它们有依赖关系的事件，但是可能有一些与它们并发的事件。如果一个计算能保证发送和接收之间不存在任何其它事件，即接收是"紧接在"发送之后的，那么这个计算在语义上与同步计算是相同的，由于通信机制并不是同步的，我们把这种计算叫准同步计算，或者叫可用同步通信实现(Realizable with Synchronous Communication，RSC)的计算[4]。

对一个计算的执行过程中，如果从绝对时间来看，所有的事件可以按时间排定一个顺序，这个顺序我们用关系 $<$ 来表示，$(C,<)$ 是一个全序的非自反集。不难看出，关系 $<$ 与 π 是一致的，即 $a\pi b\Rightarrow a<b$。因此 $(C,<)$ 是 (C,π) 的一个线性扩展。

定义5 对一个异步计算 C，$(C,<)$ 是 (C,π) 的线性扩展，如果对 $\forall(s,r)\in\Gamma$，有 $\{x\in C\mid s<x<r\}$ 为空集，则我们把 C 叫做准同步计算，或叫可用同步通信实现的计算，简称 RSC 计算，支持 RSC 计算的消息通信模型称作 RSC 消息通信模型。

由上面的讨论可知，异步通信、FIFO 通信、CO 通信和

RSC 通信存在一种层次关系，即：RSC 通信⊂CO 通信⊂FIFO 通信⊂异步通信。

2.3 同步通信模型

在同步通信模型中，消息的发送方和接收方都是阻塞的，即要双方都准备好后通信才能进行，在此之前不能进行其它的操作。从某种意义上说，发送和接收是相互依赖的。如果用＜表示同步计算中的依赖关系，则有这样两个性质：(1)$\forall$ (s,r)$\in\Gamma$,$\forall$ x$\in$C：s＜x$\Leftrightarrow$r＜x；(2)$\forall$ (s,r)$\in\Gamma$,$\forall$ x$\in$C：x＜s$\Leftrightarrow$x＜r。性质(1)的意思是，在发送和接收没有进行之前，所有依赖于发送或依赖于接收的事件都不可能；性质(2)意思是，在可以进行发送和接收之前，所有先于发送或先于接收的事件都必须完成。

异步计算没有这样的性质。在异步计算中，只要发送完成，依赖于发送的事件便可以发生，不必等接收的完成，同样，只要先于发送的事件都已完成，便可以发送，不必等待先于接收的事件完成。

下面对同步计算下一个精确的定义。

定义6 同步的因果依赖关系＜是满足下列性质的最小集：(1)aπ,b$\Rightarrow$a＜b；(2) (s,r)$\in\Gamma\Rightarrow\forall$ x$\in$C：(x＜s$\Leftrightarrow$x＜r)$\wedge$(s＜x$\Leftrightarrow$r＜x)；(3)(a＜b)$\wedge$(b＜c)$\Rightarrow$a＜c。

定义7 同步通信的计算(简称同步计算)是由局部计算元组 C＝(C_1,C_2,…,C_n)、消息通信事件对的集合 Γ 以及 Γ 上的偏序关系＜构成，支持同步计算的消息通信模型称作同步消息通信模型。

3 机群系统的有限延迟 DFIFO 模型

对于机群系统，我们通常可以认为其通信机制有如下特性：

(1)*异步通信* 通信以异步方式进行。机群系统中普遍采用的是异步通信，采用同步通信的很少，研究异步通信系统更有普遍性。

(2)FIFO *通道* 两个进程间消息通道是 FIFO 型的。大量的通信支撑软件都保证消息通道的 FIFO 特性，即两个进程间的多次消息发送顺序和接收顺序是一致，这个特性往往是程序员所希望的。支持这个特性并不困难，只要在消息控制域中加上一个序列号便可以实现。

(3)FIFO *缓冲* 发往同一个进程的消息，接收顺序与消息实际到达的顺序一致。因为存在缓冲机制，当一个进程要接收消息时，可能有几个消息已经到达该进程的消息缓冲区，这时应该取最早到达的消息。

(4) *消息延迟有限* 一般的异步计算模型总是假设消息延迟是任意的，在实际系统中，我们可以加上限制，认为消息延迟可以在一个区间[D_{min},D_{max}]内变化，D_{min}是最小延迟，D_{max}是最大延迟。这里的消息延迟指的是从消息发送到消息到达目的进程的消息缓冲区的时间。

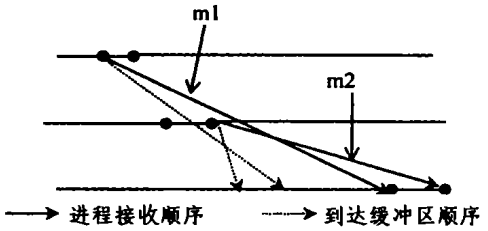

图3 一个 CO 计算

(1)(2)两个特性说明，这种计算是 FIFO 计算。(3)(4)两个特性提出了新的限制条件。不是所有的 FIFO 计算都符合这两个特性。例如图3中的计算是一个 CO 计算，自然也是一个 FIFO 计算，我们看到，消息 m2实际上比 m1晚到(虚线箭头表示消息到达进程的接收缓冲区)，而被先接收，这样做保证了消息的因果顺序，但不符合特性(3)的要求。图3的情况是很可能出现的，虽然 CO 计算可能更能反映算法的意图，但是很少有系统能保证 CO 计算，而我们描述的这些特性更接近实际的系统。

我们从一个进程接收消息的角度来形象化地看一看我们的模型。如图4所示，进程有一个接收缓冲区队列，每一个其它的进程都有一个消息通道连到该缓冲区队列，该缓冲区队列和所有的通道都是 FIFO 队列，可以说这是一种双重 FIFO 队列，再加上对消息延迟的限制，我们把这个通信模型叫做有限延迟的双重 FIFO(Delay Bounded Double FIFO)模型，简称 DFIFO 模型，基于这个消息通信模型的计算叫 DFIFO 计算。下面我们对 DFIFO 模型进行形式化描述。

与前面的模型不同，DFIFO 模型中引入了时间因素，我们假设有一个理想计时器，为每个时间打上统一的时戳，事件 e 的时戳用 T(e)表示。每个消息有一个延迟，为了表示这个延迟，我们为每个消息引入一个影子事件(Shadow Event)，表示消息到达接收方缓冲区的事件。如果消息发送事件为 s，则记其对应的影子事件为 s*，我们假设每个消息都能到达目的地，因此每个发送事件都有影子事件。

定义8 一个并行计算 C，我们说 C 是一个通信延迟在[D_{min},D_{max}]之间的 DFIFO 计算，是指满足以下条件：(1)对$\forall$ (s,r),(s',r')$\in\Gamma$,有：s$\sim$s'$\wedge$r$\sim$r'$\wedge$sπs'$\Rightarrow$rπr'；(2)对$\forall$ (s,r),(s',r')$\in\Gamma$,有：r$\sim$r'$\wedge$s*πs'*$\Rightarrow$rπr'；(3)对$\forall$ (s,r)$\in\Gamma$,有：(T(s)+D_{min})$\leqslant$T(s*)$\leqslant$(T(s)+D_{max})和 T(s*)$\leqslant$T(r)。

DFIFO 模型除了具有 FIFO 计算的所有性质外，还具有下面性质：

性质2 如果 C 是一个 DFIFO 计算，则对$\forall$ (s,r),(s',r')$\in\Gamma$,有：r$\sim$r'$\wedge$(T(s)+D_{max})＜T(s'))$\Rightarrow$ rπr'。

性质2告诉我们，目标相同的两个消息发送事件在时间上相隔足够远时，消息延迟的变化不会引起它们对应的接收事件之间的顺序的变化。这个性质在研究调试器对并行程序的打扰性(Disturbance)时很有用。我们将在以后文章中详细描述。

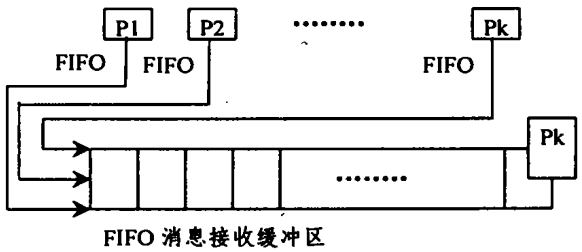

图4 双重 FIFO 机制

我们将 DFIFO 模型与其它几种模型做一比较，RSC 模型和同步模型对机群系统显然是不合适的；一般的异步模型适用范围广，但过于宽泛，不能反映实际系统的一些特性(如 FIFO 特性)，给算法分析带来困难；CO 模型具有许多有趣的性质，从算法设计的角度来讲，CO 模型很具吸引力，但是 CO 模型在实际系统中得不到保证，缺乏实用性；FIFO 模型与

DFIFO 模型最为接近，但是 DFIFO 考虑了接收缓冲区和消息延迟，比 FIFO 模型更能反映实际系统的特性，尤其是机群系统的特性。

4 DFIFO 消息通信模型的应用

DFIFO 消息通信模型，是我们正在开发的机群系统并行程序调试环境--DENNET(DEbugging eNvironmeNt for parallEl programs on clusTer systems)[5]的基础，它主要应用于以下几个方面：

1)记录/重放调试 依据 DFIFO 消息通信模型，可以对并行程序中的不确定性事件进行排序，从而解决调试并行程序时的不确定性问题。在 DENNET 中，我们已经实现了基于向量时钟的执行/重放功能，可以对并行程序进行确定性的调试[5]。

2)调试器对并行程序执行的干扰特性的判定 并行程序具有不确定性的特征，调试器的介入会影响并行程序的不确定性行为，这就是我们常说的干扰效应。有了 DFIFO 消息通信模型，可以方便地研究调试器对并行程序的干扰特性，进而搞清在什么情况下有干扰，在什么情况下没有干扰。我们设计了一个判定算法，它可以在调试程序的过程中实时地告知程序员本次调试活动是否对并行程序的行为造成了干扰[6]。

3)检查点设置/回卷恢复时全局一致性状态的确定 在 DENNET 中，拟实现检查点设置/回卷恢复功能，以支持大规模长时间运行程序的调试。回卷恢复时，关键是要确定全局一致性的状态，我们结合记录的检查点设置方法和 DFIFO 消息通信模型来确定回卷恢复时的全局一致性状态。

4)性能调试中的性能预测 性能调试是 DENNET 系统的重要功能[7]，我们拟采用解析建模的方法实现有限的性能预测功能[8]，解析建模的对象(并行程序)同样以 DFIFO 消息通信模型为基础。

本文主要是介绍我们提出的 DFIFO 消息通信模型，上述几方面的内容将会有专门的文章单独讨论，在此不再详细叙述。

结论 本文讨论了机群系统中消息通信的模型，首先用形式化的方法，总结了常用的几种消息通信模型：同步消息通信模型、异步消息通信模型、FIFO 消息通信模型、CO 消息通信模型和 RSC 消息通信模型，讨论了它们的一些性质及其相互关系。然后，本文提出了一个有限延迟的双 FIFO 消息通信模型。与现有的模型相比，DFIFO 消息通信模型更能反映机群系统的特性，如消息缓冲、消息延迟等，利用 DFIFO 消息通信模型可以对并行程序的执行进行更为细致而实际的分析与研究。从而为解决机群系统中其它的关键技术建立形式化的基础。

建立机群系统的消息通信模型，是研究机群系统中其它技术的基础。在设计调试器时，可以用上述的消息通信模型解决不确定性问题；也可以应用消息通信模型来分析研究调试器对并行程序的干扰特性；在检查点技术中，要应用消息通信模型来确定全局一致性的状态；机群系统的时钟问题涉及确定性调试、性能调试和容错技术等诸多方面的基础，建立机群系统的时钟需要应用消息通信模型；对并行程序进行性能预测，同样要建立并行程序的消息通信模型；在静态分析并行程序的不确定性因素时，建立高效而又符合实际的消息通信模型是非常重要的。本文对 DFIFO 消息通信模型在 DENNET 的设计和实现中的应用情况，作了简要描述。

参考文献

1 Shatz S M. Communication Mechanism for Programming Distributed Systems. Computer, 1984, 17: 21～28
2 Lamport L. Time, Clock and Ordering of Events in a Distributed System. Communication of ACM, 1978, 21(7)
3 Birman K, Joseph T A. Reliable Communication in the Presence of Failure. ACM Transactions on Computer Systems, 1987, 5: 47～76
4 Soneoka T, Ibaraki T. Logically Instantaneous Message Passing in Asynchronous Distributed Systems. IEEE Transactions on Computer, 1994, 43: 513～527
5 Liu Jian, Yu Hong-liang. Implementation of a Debugger for Parallel Program in Cluster System. The 7th Joint Intl. Computer Conference, 2000. 11, Shantou: Guangdong
6 Xiong Jianxin, et al. On-line Debugging of Parallel Program. Intl. Conf. on Parallel Algorithm(ICPA'95), Wuhan Oct. 1995
7 刘建，余宏亮，等. 并行程序性能调试环境模型. 中国计算机体系结构2000学术年会，哈尔滨：2000
8 赵刚，沈美明，郑纬民. 并行程序集成环境的特点及 IPCE2.0实现中的关键技术. 中国计算机体系结构2000学术年会，哈尔滨：2000

(上接第26页)

1. 同步或异步的方法调用：同步调用时，COT 必须收到调用的结果。因此，COT·的库所不能被标记，直到调用返回。同时，"target"值是"yes"。在异步调用时，即使托肯从 COT 传给了 OIP，COT·的库所仍然有标记，COT 的异步调用中，托肯传给 OIP 和 COT·，方法调用的结果保存在全局变量中，以便以后使用。

2. 嵌套调用：在调用期间，可能调用其它对象的方法或自己的方法。变量 pid 是表示方法调用的嵌套。当 COT 调用一个方法时，变量 pid 为传递托肯值，进程识别器识别由方法调用创建的类的实例，pid 的大小和子序列调用次数成正比。

3.6 对象实例化

当一个类初始化系统行为(在过程语言中相当于 main())，或一个类被其它类(或对象)调用时，该类的一个实例可创建。类的实例有识别器，它可区别变量 pid 的值。实例 pid 的值对应 main()函数，是自包含的，被调用实例的 pid 值返回指针给调用类。

总结 HOONet 在建模方法上采用细化和分层，用 HOONet 建模通过合适的接口支持信息隐藏；通过抽象库所、变迁和托肯支持抽象；通过动态绑定抽象托肯支持多态，通过类之间的共享支持继承；通过消息传递支持交互(interaction)。因此，HOONet 不仅具有 Petri 网的直观模拟系统动态行为的能力，同时也具有面向对象技术的优点。

参考文献

1 Kilov H, Ross J. Information Modeling—An Objected Oriented Approach. Prentice-Hall, 1994
2 Bastide R. Approaches in unifying Petri nets and the object-oriented approach. In: Proc. of the Intl. Workshop on Objected-oriented Programming and Models of Concurrency, 1995, 6
3 Peterson J. Petri Net Theory and The Modeling of Systems. Prentice Hall, 1981
4 Hong Jang-Eui, Bae Doo-Hwan. Software modeling and analysis using a hierachical object-oriented Petri net. Information Sciences, 2000
5 袁崇义. Petri 网原理. 北京：电子工业出版社，1998
6 张新晖，朱淼良，吴春明. 立体分层 Petri 网对象化专家系统实现. 小型微型计算机系统，2000, 3