计算机科学2000Vol. 27№. 8

信息传送和对象生成的程序设计方法*)

The Programming Method of Message Transferring and Object Generating

吕英华 刘学军
(东北师范大学计算机科学系 长春130024)

Abstract NETL is the developing language of parallel object oriented. It enhanced the message synchronization and parallel transferring. It can also generate objects dynamically. A programming method is given to describe the parallel transferring of the message and the generating of the dynamic object.

Keywords Dynamic object, Parallel message transferring, Method description

1 引言

面向对象中的最基本概念就是对象(object)和信息(message)。程序是若干个对象这一独立模块的集合。各对象具有独自的内部状态,对象可以引用或改变自己的内部状态,但不能被其它对象直接引用或更改。描述对象动作的是方法 method。在面向并行对象语言的程序中,通过对象之间的信息交换,启动被委托的 method 的执行[1]。

如果某对象向另一对象传送信息后,在接收到被委托的对象的回答之前不再进行后面的处理的话,这同平常的"子程序调用/返回"没有什么大的区别,对面向并行对象来说,某对象把信息传送出去后仍可继续后面的处理,此时,送信一方与受信一方就在并行地执行各自的程序。

在面向并行对象语言中,设置了有关同步和并行的描述,使得在程序中可以对问题中的并行性进行自然的描述。NETL 就是以在并行处理机上的执行作为前提,以 ACTOR 模型、ABCL/1 的信息传送、Smalltalk-80的 method 等作为参考,在 NEWS 工作站上利用 C 语言开发的面向并行对象语言[2,3]。下面就 NETL 语言中信息传送和对象生成的描述和程序设计方法作一概述。

2 对象及对象装入的描述

2.1 对象的描述

对象的描述如下:

```
object objectName
control controlStatements
state state Variables and their initialization
methods{
    messagePattern{|temporaries|statements}
    messagePattern{|temporaries|statements}
    ...
    ;
```

可见,对象是 method 的集合。如果对象接收的信息和某一信息模式匹配的话,就执行其后的处理,即 method。在一个对象内,某信息模式是唯一的。

2.2 对象装入的描述

把一个处理分成若干个任务在多个处理器上执行的时候,处理器之间的通信时间所占的比例对程序的执行时间有很大的影响。当然,在同一处理器上执行速度反而快的应用问题也存在。NETL 中设置了装入对象的描述,使得相互通信量小并且可以并行执行的对象装入在相互邻近的、尽可能不同的处理器上,而通信量大并且不能并行执行的对象装入在同一个处理器上。

下面的描述中,指定"group"后面的几个对象尽可能装入到相互邻近的几个不同的处理器上。

group objectname1,objectname2,objectname3.

下面的描述中,指定"collect"后面的几个对象尽可能装入到同一个处理器上。

collect objectname1,objectname2,objectname3.

未经上面指定的对象在装入时尽可能装入到不同的处理器上。

3 信息传送的描述

3.1 单一信息传送

为了提高并行性,因为同步所等待的时间越少越好,为此在 NETL 中设置 past、now、future 三种信息类型,仅在必要的时候才采用同步。

(1)past 型 对象 sender 向对象 receiver 传送信

*)本文工作获日本学术振兴会资助。

息 message 的描述如下：

receiver message

sender 把信息 message 送出后，继续下面的处理，因此是非同步的信息传送。主要用于对可以与其并行执行的、不需要返回处理结果的对象的委托。

(2)now 型　对象 sender 向对象 rcv 传送信息 message 的描述如下：

?receiver message

sender 把信息 message 送出后，一直等待来自 receiver 的回答信息。不管 sender 是否利用该回答信息，都要中断下面的处理，因此 sender 与 receiver 之间不能并行处理。receiver 在处理的过程中，如遇到"!"，则向 sender 返回回答信息。

(3)future 型　对象 sender 向对象 receiver 传送信息 mess 的描述如下：

result＝receiver message

sender 把信息 message 送出后，同(1)一样继续下面的处理。当引用存贮 receiver 的返回值(亦称未来值)的变量 result 时，如未来值尚未返回，则在引用处中断下面的处理，等待未来值的返回；如在引用时，未来值早已返回，则不发生中断继续下面的处理。receiver 在处理的过程中，如遇到"!ans"，则把变量 ans 的值作为未来值返回给 sender。

3.2　多个信息并行传送

NETL 重视多个对象间的通信，定义了把同一信息同时向若干个对象传送的信息扩散传送方式。

NETL 中，如把表示对象所在的地址，即对象指针存于某表中的话，当在表前加上@时，即向表中所对应的各个对象进行信息扩散传送。如对象 sender 以信息扩散传送方式向三个对象 obj1，obj2，obj3传送过去型信息 mess 时，可描述如下：

```
objList with:(obj1,obj2,obj3).   "destination list"
@objList mess.   "multicast"
```

对未来型信息进行扩散传送时，其它各对象的回答信息按到达的顺序以表的形式存于指定的变量中，该表在所有的回答信息全部返回前不许读出。

3.3　多个信息的接收

对于启动 method 执行的信息，在描述时可指定多个，只有这些信息都被接收后才启动该 method。

mess1，mess2、mess3为不同类型的信息，这三种信息全部被接收后某 method 才被启动，此时可描述如下：

```
mes1 & mess2 & … &messN "receptions of mes1 through mesN"
```

如某种类型信息 mess 有三个被接收后才启动某 method，则可描述如下：

```
3 * mes "  the N-time receptions of mes"
```

下面就是一个将自身的 ID 作为过去式信息发送给 N 个对象，由 M 个回答信息来启动 method 的 NETL 程序。

```
#define N 10
object element[N] "indexed object"
state myOpinion "state variable"
methods{
      ask:sender|"method ask "
         "decide myOpinion"
         case myOpinion{
          'agree' [sender agree]
          'disagree' [sender disagreee]
              default ["abstention"]
         }
        }
}
#define N 10
#define N 6
extern element[N]. "external object"
object collector "object name"
state poll. with:nil.  "initialize a state variable"
methods{                "method start"
start{
 |temp|
 temp=1.
 while(temp<=N){
       poll addLast:element[temp]. "make a list of the element"
       temp=temp+1.
 }
 @poll ask:self. "multicast a message"
}
M * agree{"decide agreement"} "multicast receive method"
M * disagree("decide disagreement"}
}
```

4　对象生成的描述

4.1　对象的动态生成

在模拟或解决协调型分散问题等领域，往往需要若干个具有同一处理过程的对象[4]。在 NETL 中，把这样的对象的雏形作为 class 来定义。在某对象执行的过程中，如向 class 传送信息，就会生成新的对象。由于程序本身可以随时增殖，就可以大大减少编程时对对象的描述。一般把程序执行时由 class 生成的对象叫动态对象，程序在执行前就存在的对象叫静态对象[5]。

下面程序中，有一名为 quick 的 class。

```
#define MAX 127
#define N 4
class quick
methods{
      sort:(num,data){
          |noOfdata rIndex lIndex rSon lSon lData rData|
          noOfData=data size.
          rIndex=2 * num+2.
          if((noOfData<N)|(rIndex>MAX))then{
          "……sequential sort…"
          }
           else {
         rSon=quick(rIndex)new
         lIndex=rIndex-1.
         LSon=quick(lIndex)new
         "…separate the data into two,lData and rData…"
         lData=lSon sort:(lIndex,lData).
         rData=rSon sort:(rIndex,rData).
         "…merge lData and rData into data…"
```

(下转第58页)

问题4的规范描述如下.

9)DM(D,A,B,B−A+1)⇐find some L where L from 1 to B−A+1 such that (DC(D,A,B,L) and L≥all k where k from 1 to B−A+1 such that DC(D,A,B,k))

根据 RS_2,方程9)可推导为:

⇐φ　　if A>B
⇐B−A+1　if DC(D,A,B,B−A+1)
⇐Find some L where L from 1 to B−A such that (DC(D,A,B,L)and L≥all (k) where k from 1 to B−A such that DC(D,A,B,k)) otherwise

上式的划线部分是方程9)的一个实例,用方程9) folding 可得:

⇐φ　　if A>B
⇐B−A+1　　if DC(D,A,B,B−A+1)
⇐DM(D,A,B,B−A)　otherwise

根据方程7),8),9),我们可得到最后的算法 A_4:

DM(D,A,B,B−A+1)
⇐φ　　if A>B
⇐B−A+1　　if DC(D,A,B,B−A+1)
⇐DM(D,A,B,B−A)　otherwise
DC(D,A,B,M)⇐False　if A+M−1>B
⇐True　if Equ(D,B−M,B)
⇐DC(D,A,B−1,M)　otherwise
Equ(D,k,j)⇐True　if k−1≥j
⇐Equ(D,k,j−1)　if D[j]=D[j−1]
⇐False　otherwise

结束语　形式化开发算法程序是提高软件生产率和可靠性,最终实现软件自动生成的重要途径,从上述算法程序的演绎综合实践,我们看到将算法的共性上移,个性延迟决策是实现算法程序推导步重用的重要手段,这有助于利用尽可能少的推导步形式化地开发出尽可能多的一类算法。容易看到,使用这种方法我们还能推导出类似子集和问题、背包问题等其它一类数组问题的算法程序。

参 考 文 献

1 Darlington J. A synthesis of several sorting algorithms. Acta Informatic,1978,11:1～30
2 许卓群,张乃孝,杨冬青,唐世渭. 数据结构. 高等教育出版社,1987
3 黄明和,刘润杰. 图回路算法的演绎综合. 计算机科学,1999 11

(上接第51页)

```
  }
  !data. "return the sorted data"
 }
} "end of the methods."
```

如果其它对象执行下面这一未来型信息传送式:

newobject=quick new.

于是,就生成了新的动态对象,该动态对象指针由于 class 内回答信息的返回而被代入变量 newobject 中。这样,发出生成动态对象委托的对象可用变量 newobject 来引用新生成的动态对象,亦可用 newobject 作为参数向其它对象传送信息,即向其它对象通知新的动态对象的诞生。于是,其它对象也可以引用新生成的动态对象了,在 class 内部引用临时变量 cell 来引用新生成的对象。

4.2 动态对象的消除

与对象的动态生成相对应,NETL 设置了动态消除对象的功能,这由执行下面的语句来实现:

objectName delete.

即向要消除的对象发出特殊的信息 delete 即可。被消除的对象将从系统的管理表中彻底删除掉,因此,用户适当使用生成动态对象和消除对象的功能,可使有限的硬件资源得以充分的利用。

4.3 method 执行的中止

某对象启动另一个对象,当已接收到来自被启动的对象的回答信息后,如果此时被启动的对象还在继续执行,就可以中止被启动的对象的处理。NETL 利用特殊信息"kill"来中止被委托的 method 的执行。假如被委托的对象为 anobject,其 method 的信息模式为 job 的话,中止该 method 的描述为:

anobject kill:job.

有权发出处理中止的只能是启动该 method 的对象。

参 考 文 献

1 吕英华. A-NET 模拟程序中进程间通信接口的实现. 计算机科学,1999,26(11),8～10
2 吕英华. UNIXシステムコールによるA-NETルータ间通信模拟の实现. In:研究纪要(日本数学教育学会春季年会发表论文),1998. 19～21
3 吕英华. 并列计算机シミュレータにおけるHostとRouter间のインタフエース. In:研究纪要(日本数学教育学会秋季例会发表论文),1998. 61～63
4 吕英华. UNIXのペイプによるセマフオの作成. In:日本第42回数学の文化史研究会发表论文,1998
5 吕英华. A-NETL 中对象的动态生成和在 A-NET 机中的实现. 计算机科学,1998,25(4):111～112