第8卷第11期
1997年11月

软 件 学 报
JOURNAL OF SOFTWARE

Vol. 8, No. 11
Nov. 1997

用于含过程调用DO循环的循环嵌入方法*

原庆能　丁永华　臧斌宇　朱传琪

（复旦大学并行处理研究所　上海　200433）

摘要　循环是程序中蕴含并行性最为丰富的一种结构，因此成为并行化编译最主要的对象. 但循环内的过程调用严重妨碍了循环的数据相关性分析，使得循环语句潜在的大量并行性得不到开发. 本文提出的循环嵌入方法使部分含过程调用循环语句的并行化成为可能，对部分用其它过程间分析技术也能开发其并行性的这一类循环语句采用循环嵌入方法，并行化开销低，并且分析更精确. 采用循环嵌入方法还可降低程序由于多次过程调用带来的调度开销. 这一方法在作者开发的自动并行化编译系统AFT(automatic Fortran transformer)中得到了实现，对Spec92测试程序包的试验结果表明了本文提出的方法是行之有效的.

关键词　过程嵌入，过程繁衍，循环嵌入，数据相关性分析，加速比.

中图法分类号　TP311

高级程序设计语言中过程的广泛采用是并行处理领域无法回避的问题. 程序中过程调用的出现使数据相关性分析精度受到严重影响，大量并行性因此得不到开发. 为解决这一问题所进行的探索促进了过程间分析技术的发展，这些技术包括过程间的数据流分析、向前替代、相关性分析、过程嵌入[1]、过程繁衍[2,3]和过程间信息传播[4]等. 但是当循环语句中含有过程调用时，由于现有过程间分析技术的局限性，循环中的并行性因缺乏被调用过程的信息而难以开发出来. 过程嵌入是解决此类问题的一个简单而又有效的方法. 过程嵌入就是用过程的体变形后替换对该过程的调用语句，其效果可达到使用过程间数据流分析技术开发得到的并行性的上限. 过程嵌入看起来简单有效，实际上有不少缺点. 第1，过程嵌入并不是总能实现的，它不能处理递归调用，有些情况，如数组大小和维数不匹配，FORTRAN77被调用过程中含有save语句等，过程嵌入在实现上比较困难且繁琐. 第2，把所有过程嵌入主程序可能导致程序长度按调用深度指数级地膨胀，使得程序因占用空间过大而难以处理. 第3，过程嵌入使过程可分别进行分析、编译的优点消失，导致并行化分析、编译等操作的开销急剧增加. 第4，过程嵌入破坏了程序的模块性，导致变换后的程序不易阅读和维护. 第5，过程嵌入使子程序中的局部变量作用域扩大，可能引起新的相关. 当由于过程嵌入而增加的开销超过了过程嵌入所能带来的好处时，过程嵌入就变得没有意义了. 因此，我们只在某些

---

* 本文研究得到国家自然科学基金、国家863高科技项目基金、国家攀登计划基金和上海市重点学科基金资助. 作者原庆能，女，1966年生，讲师，主要研究领域为并行编译. 丁永华，1971年生，助教，主要研究领域为并行编译. 臧斌宇，1965年生，讲师，主要研究领域为并行编译. 朱传琪，1943年生，教授，博士导师，主要研究领域为并行处理.

本文通讯联系人：丁永华，上海200433，复旦大学并行处理研究所

本文1996-12-10收到修改稿

情况下，如要进行跨过程变形等时，才进行过程嵌入.

而过程间信息传播在保留过程及其调用点信息的情况下，通过收集、传播过程的引用信息、常数值等来解决过程带来的困难，对过程仅作一次处理. 过程间信息传播虽然能克服过程嵌入的上述缺点，但也解决不了含过程调用循环语句的并行化问题，因为在含过程调用的循环语句内，过程被多次调用，每调用一次，实参都有可能不一样，不同的实参可能带来不同的信息，因此使精确度下降.

过程繁衍同样无法得到含过程调用循环语句的精确数据流信息，因此很难并行化此类循环语句. 这是因为每次过程调用的信息都有可能不一样，而过程繁衍在有不同信息的调用点，通过产生被调用过程的副本来保留过程以及调用点的实参信息，所以过程繁衍用于含过程调用的循环语句，为了获得精确的实参信息，有可能产生等于循环次数的过程副本数，实际上这是不可行的.

其它的过程间分析技术用于并行化含过程调用的循环语句，效果也不甚理想. 为进一步开发含过程调用循环语句的并行性，降低并行化含过程调用循环语句的开销以及减少过程被多次调用的调度时间，本文提出了一个既直观又有效的方法：循环嵌入.

循环嵌入就是将循环语句和循环体内除调用语句外的其它部分嵌入到被调用的过程中，原循环语句用一个调用该过程的调用语句取代. 若同一过程有多个调用点都要作循环嵌入的变形，还可将信息相似的循环嵌入后的过程副本进行合并，这样可以减少程序的膨胀，也可降低对循环嵌入后的过程副本进行分析、编译的开销. 该方法已在我们开发的自动并行化编译系统 AFT 中实现[5]，通过对测试程序包 Spec92 的试验结果可以看到本文方法的实际效果.

1 循环嵌入方法

1.1 一个例子

循环嵌入是一种针对含过程调用循环语句的过程间变形方法. 如下面的例 1：

```
......
DO 10 I=1,500-1                    SUBROUTINE SUB(K,JBEG,JEND)
  call SUB(I,I+1,500)                ......
10 CONTINUE                        END
......
```

例 1 作循环嵌入的过程间变形后结果如下：

```
                                   SUBROUTINE SUB(K,JBEG,JEND)
......                                DO 10 K=1,500-1
call SUB(I,I+1,500)                     JBEG=K+1
......                                  ......
                                   10 CONTINUE
                                   END
```

一方面，由于循环语句嵌入过程后，该循环语句成为被调用过程 SUB 内包含所有其它语句的最外层循环，循环内的过程间数据相关性分析转化为过程内数据相关性分析，降低了循环的并行化开销. 另一方面，原循环语句由一个调用语句取代，使过程调用次数由原来的 499 次下降到 1 次，减少了过程被重复多次调用的调度开销.

1.2 循环嵌入的条件

作循环嵌入过程间变形的主要目的是为了尽量并行化含过程调用的循环,考虑到算法的复杂度和有效性以及变形后要保持程序语义的正确性,并不是对所有含过程调用的循环语句都进行循环嵌入的变形,而是作一些限制,符合以下条件的才进行循环嵌入的过程间变形:

a. 循环中无外跳 GOTO 语句.

有外跳 GOTO 语句的循环是非结构化的,其并行性本来就难以开发出来,而作循环嵌入后循环的并行性因外跳 GOTO 语句的存在仍难以开发. 据此作了条件 a 的限制.

b. 循环语句和循环体内除调用语句外的其它语句中以及在调用点有循环变量出现的实参中,除循环变量外,所有全局变量所在的公用区在被调用过程的公用区说明中出现,所有局部标量在调用点作为实参变量,所有局部数组元素的数组名或与其同名的数组元素在调用点作为实参变量且局部数组元素除最后一维外的各维上界或为常量,或为其所在的公用区在被调用过程的公用区说明中出现的全局变量,或为调用点的实参.

FORTRAN 程序中不同程序单位之间的数据交换主要通过两条途径:公用区方式和虚实结合. 请看下面的例 2:

```
......                              SUBROUTINE SUB(K,B)
DO 10 I=1,M                           ......
  call SUB(I,A)                       END
10 CONTINUE
......
```

其中 M 是局部变量.

由于 M 是局部变量,又不是调用点的实参,作循环嵌入,除非增加过程 SUB 的参数,否则 M 的值无法传递进过程 SUB 中,考虑到算法复杂度和有效性的平衡,我们对这种情况不作处理,但理论上是可以完成的.

循环嵌入被调用过程后,一方面,循环变量的值在被调用过程中被直接引用,另一方面,循环中无外跳 GOTO 语句,循环变量在循环执行完后的值可由循环的初值、终值和增值计算出来,因此,不对循环变量进行限制.

对循环语句和循环体内所有同名局部数组元素,若循环变量不在其下标表达式中出现,且在调用点与它同名的实参数组元素对应的虚参为一标量,则在本算法中将此局部数组元素作为局部标量看待.

要使每个局部数组元素在被调用过程中都有一确定的虚数组元素对应,有时需对有关虚参进行扩展. 在本算法中,若虚参是标量,则扩展为数组,若虚参是数组,则将虚数组向后扩展.

请看下面的例 3:

```
......                              SUBROUTINE SUB(B,...)
DO 10 I=1,999                         ......
  A(I)=...                            END
  call SUB(A(I+1),...)
10 CONTINUE
......
```

在例 3 中,虚参 B 扩展后,$A(1)$仍没有虚数组元素对应.

因此,循环嵌入时,还需将在调用点与局部数组元素同名的实参数组元素与虚参的结合改为实参数组元素的数组名与虚参的结合.这种改动使虚实数组元素之间的对应关系有可能发生变化,需要对虚数组元素的下标重新计算.当相结合的虚实数组不是同维、同大小的数组时,局部数组元素的下标也要重新计算.虚参标量的扩展以及对局部数组元素下标和虚数组元素下标的替换计算需要知道虚实数组除最后一维上界外的各维的上下界.在并行化编译系统AFT(automatic Fortran transformer)中,所有数组各维下界均转化为1,因此,在本算法中假定数组各维下界为1.

c. 若被调用过程中某一虚参出现在数组的维说明表达式中,在调用点此虚参对应的实参含有循环变量,则该实参只能是循环变量的线性表达式,维说明也只能是该虚参的线性表达式.

FORTRAN程序规定对同一变量不允许重复说明.为尽可能节约空间和避免数组下标越界,循环嵌入时,含有循环变量的维说明用所有循环变量值代入后维说明值的最大值取代.为降低算法的复杂度和实现算法的开销,作了条件c的限制.条件c的限制保证了维说明的最小上界在循环变量的初值或终值处取得.

1.3 循环嵌入的实现方法

对满足上述条件的循环进行循环嵌入按以下3个步骤进行:① 虚参扩展;② 变量替换;③ 嵌入变形.

1.3.1 虚参扩展

将在调用点与局部数组元素同名的实参数组元素或数组名对应的虚参进行扩展.若对应的虚参为一标量,则将此虚参说明为与实参数组同维、同大小的数组,数组最后一维的上界用"*"号表示.若对应的虚参为一数组名,将虚数组定义中最后一个维说明改为"*"号.

1.3.2 变量替换

(1)对循环语句和循环体内除调用语句外的各语句中的全局变量和局部变量进行替换.

首先,将全局变量替换为被调用过程的公用区说明中相应的变量.然后,根据虚实结合将局部变量替换为相应的虚参.替换规则是将局部标量直接替换为相应的虚参.若循环变量不是符合条件b的变量,在被调用过程中申请一新的变量来替换循环变量.对于局部数组元素,若在调用点有一实参是它的数组名或与它同名的数组元素,其对应的虚参是与它同维、同大小的数组,则将局部数组元素的数组名替换为相应的虚数组名,并对局部数组元素下标中出现的变量进行替换,替换方法如本节所述.若实参数组元素对应的虚参不是同维、同大小的数组,除了替换数组名外,数组元素的下标也需要进行替换.设该局部数组元素为N维,除最后一维外的各维上界分别为$M_1,M_2,\ldots,M_{N-1}$,各维下标分别为$a_1,a_2,\ldots,a_N$,设虚数组在被调用过程中定义为N'维,除最后一维外的各维上界分别为$M'_1,M'_2,\ldots,M'_{N-1}$.

首先,对$N,M_1,M_2,\ldots,M_{N-1},a_1,a_2,\ldots,a_N$中的变量进行替换,替换方法如本节所述.然后,计算该局部数组元素和它的第1个元素之间的距离:

$$DIS=\sum_{i=1}^{N}((a_i-1)*\prod_{j=1}^{i-1}M_j)$$

根据FORTRAN程序规定以列为主序存放的原则,用来替换的各维下标可以用如下的

程序段计算出来:

FOR $i=0,N'-1$

{

$$b_{N'-i}=INT(DIS/\prod_{j=1}^{N'-i-1}M'_j)+1$$

$$DIS=MOD(DIS/\prod_{j=1}^{N'-i-1}M'_j)$$

}

用计算出来的下标替换原来的各维下标.

(2)在调用点某些实参数组元素与虚参的结合要改为实参数组元素的数组名与虚参的结合,这些实参数组元素对应的虚参需要进行替换.

若数组元素对应的虚参为一标量,由 1.3.1 节知,已将此标量名说明为数组,将被调用过程中所有该虚参标量替换为虚数组元素,虚数组元素的下标取实参数组元素的下标,并对下标中的变量进行替换,替换方法同(1).

若数组元素对应的虚参为一数组名,设实参数组元素为 N 维,除最后一维外的各维上界分别为 $M_1,M_2,\ldots,M_{N-1}$,各维下标分别为 $a_1,a_2,\ldots,a_N$,设被调用过程中虚数组定义为 N' 维,除最后一维外的各维上界分别为 $M'_1,M'_2,\ldots,M'_{N'-1}$,在被调用过程中出现的虚数组元素的下标为 $b_1,b_2,\ldots,b_{N'}$,则虚数组元素下标的替换方法如下:

首先,对 $N,M_1,M_2,\ldots,M_{N-1},a_1,a_2,\ldots,a_N$ 中的变量进行替换,替换方法同(1)然后,计算出虚数组元素和实数组第 1 个元素之间的距离:

$$DIS=\sum_{i=1}^{N}((a_i-1)*\prod_{j=1}^{i-1}M_j)+\sum_{i=1}^{N'}((b_i-1)*\prod_{j=1}^{i-1}M'_j)$$

则用来替换的各维下标可用(1)中的程序段计算出来,用计算出来的下标替换虚数组元素下标.

(3)若调用点的实参是关于循环变量的表达式,将该实参对应的虚参被重新定义前的引用进行替换.

对此实参中出现的变量进行替换,替换方法同(1). 将已替换好的表达式取代该实参对应的虚参被重新定义前的引用. 如在例 1 中,调用点的实参表达式 $I+1$ 对应的虚参 $JBEG$ 在过程 SUB 中被重新定义前的引用用表达式 $K+1$ 替换.

(4)若含有循环变量的实参对应的虚参出现在数组的维说明中,对此维说明进行替换.

将此维说明用循环变量初值和终值代入后的最大值取代.

1.3.3　嵌入变形

将循环语句和循环体内除调用语句外的其它部分嵌入到被调用过程中,原 DO 循环被调用该过程的调用语句取代,若调用点的某一实参是数组元素,则根据需要在调用语句中将该数组元素用它的数组名取代.

循环嵌入时,对语句中的标号要作一定的变形,以避免循环嵌入后标号的冲突;被调用过程中的 RETURN 语句要改为相应的 GOTO 语句;若原 DO 循环中调用过程的语句出现在逻辑 IF 语句里,逻辑 IF 语句也要根据需要变形为块 IF.

当被调用过程有多个调用点时,针对某个调用点的变形需要对该过程作一个备份,这个备份称为过程副本,将循环嵌入到这个过程副本中,相应的调用点调用这个过程副本,而其

它调用点调用原来的过程.

若循环变量不是调用点的实参,如下面的例4:

```
      ......
      DO 10 I=1,1 000                    SUBROUTINE SUB(B)
        call SUB(A(I))                     ......
   10 CONTINUE                           END
      ......
```

若循环变量也不是全局变量,则在被调用过程中申请一新的变量来替换循环变量,原DO循环用调用被嵌入过程的调用语句和计算循环执行完后循环变量值的赋值语句取代.

循环变量在循环执行完后的值可用如下的计算公式计算:

$$v=INT((e2-e1)/e3)*e3+e1+e3$$

其中v是循环变量,$e1$,$e2$,$e3$分别为循环的初值、终值和增值.

设在被调用过程中为循环变量申请的新变量为I',则例4经循环嵌入变形后的结果如下所示:

```
      ......                             SUBROUTINE SUB(B)
      call SUB(A)                          DO 10 I'=1,1 000
      I=INT((1 000-1)/1)*1+1+1               ......
      ......                             10 CONTINUE
                                         END
```

2 循环嵌入后的合并问题

如果某一过程存在多个调用点都需要作循环嵌入的变形,可将调用点所在的循环、循环语句和循环体信息相似的情况进行合并.对同一过程的不同调用点,一方面,若循环语句和循环体内除调用语句、语句标号、局部变量名不同外,其它地方都匹配一致,而同一位置的局部变量对应着被调用过程的同一虚参;另一方面,若调用语句中含有循环变量的实参,除局部变量名不同外,在同一位置匹配一致,而同一位置的局部变量对应着被调用过程的同一虚参,则认为循环语句和循环体信息相似,可以合并.

如下面的例5,作循环嵌入后的两个过程副本是可以合并的.

```
      DO 10 I=1,M                        DO 20 J=1,N
        call SUB(I,I+1,M)                  call SUB(J,J+1,N)
   10 CONTINUE                        20 CONTINUE
```

而下面的例6,作循环嵌入后的两个过程副本在本算法中不能合并.

```
      DO 10 I=1,1 000                    DO 20 I=1,500
        call SUB(I,I+1)                    call SUB(I,I+2)
   10 CONTINUE                        20 CONTINUE
```

3 试验结果分析

循环嵌入方法在我们开发的FORTRAN程序自动并行化编译系统AFT中得到了实现.经过对spec92测试程序包测试,我们发现作循环嵌入后有两个程序取得明显效果.这两个程序是mdljdp2.f和mdljsp2.f,它们需要作循环嵌入过程间变形后才会有明显的加速比.以mdljdp2.f中最主要的过程FRCUSE内一个含有过程调用的循环为例,该循环不能并行化的原因是含有过程调用语句,而相应的被调用过程需要作跨过程的归约才能并行化,

将循环嵌入被调用过程后,只需作过程内的归约就可以并行化被嵌入的最外层循环.

表 1 显示了作循环嵌入的过程间变形的有效性. 其中 PFA 是 KAP 在 SGI 上的版本,所有数据都在 SGI 公司的 Challenge 4L 系统(共享主存的 4 机 SMP 系统)上测得. AFT1 是指 AFT 系统不作循环嵌入过程间变形的执行时间和加速比,AFT2 是指 AFT 系统作这种过程间变形后的执行时间和加速比,加速比定义为 speedup=serial times/parallel times.

表 1

	serial (time:s)	PFA (time:s)	AFT1 (time:s)	AFT2 (time:s)	PFA (speedup)	AFT1 (speedup)	AFT2 (speedup)
mdljdp2	51.10	52.09	51.74	29.40	0.98	0.99	1.74
mdljsp2	49.29	49.38	49.64	29.81	1.00	0.99	1.65

4 结束语

本文给出了循环嵌入过程间变形的具体实现方法,从试验结果可以看出这一方法能够有效地提高并行化编译工具的并行化能力.

通过在被调用过程中增加虚参可以放宽 1.2 节中的条件限制. 如例 2,在被调用过程中增加一个虚参,使局部变量 M 的值通过虚实结合传递进过程,该 DO 循环就能作循环嵌入的变形. 通过增加虚参还能使更多的同一过程作循环嵌入过程间变形的过程副本合并. 如例 6,如果在过程 SUB 中增加两个虚参,一个用来传递循环变量的终值,一个用来传递 call 语句第 2 个实参中与循环变量相加的常数,则过程 SUB 在这两个调用点作循环嵌入过程间变形后的过程副本可以合并.

增加虚参将使循环嵌入算法变得复杂. 如何找到一个有效的算法,通过适当地增加虚参,使能作循环嵌入变形的 DO 循环尽可能地多,同一过程在不同调用点作循环嵌入变形后的过程副本也尽可能地合并,这是今后进一步要做的工作.

参考文献

1 Li Zhiyuan, Yew Pen-Chung. Interprocedural analysis and program restructuring for parallel programs. CSRD Rpt. No. 720.

2 Cooper K D, Hall M W, Kennedy K. A methodology for procedure cloning. Comput. Lang., 1993, **19(2)**:105~117.

3 丁永华,陈彤,臧斌宇等. 过程繁衍及其实现方法. 软件学报,1996,**7(11)**:662~668.

4 Callahan D, Cooper K D, Kennedy K *et al*. Interprocedural constant propagation. SIGPLAN Notices, Proceedings of the ACM SIGPLAN'86 Symposium on Compiler Construction, July 1986, **21(7)**:152~161.

5 朱传琪,臧斌宇,陈彤. 程序自动并行化系统. 软件学报,1996,**7(3)**:180~186.

LOOP EMBEDDING APPLIED TO DO LOOPS WITH PROCEDURE CALLS

YUAN Qingneng　DING Yonghua　ZANG Binyu　ZHU Chuanqi

(Institute of Parallel Processing　Fudan University　Shanghai　200433)

Abstract Loops is a kind of structure in which the parallelism is the most abundant, so it is the most important source of parallelism. However procedure calls in loops obstruct greatly the data dependence analysis in loops and make the potential plentiful parallelism in loops can not be developed. In this paper, a method called loop embedding is presented. Loop embedding can make parallelizing the partial of DO loops with procedure calls become possible. For the partial of DO loops with procedure calls, in other interprocedural data dependence analysis technologies, it's parallelism can be developed, but in loop embedding, the cost is less. In loop embedding, the overhead, which calling the same procedure many times leads to, can be decreased. The scheme introduced in this paper was implemented in AFT (automatic Fortran transformer), a parallelizing compiler the authors developed. The test result on Spec92 illustrates the effectiveness of their method.

Key words Procedure embedding, procedure cloning, loop embedding, data dependence analysis, speedup.

Class number TP311