过程繁衍及其实现方法*

丁永华 陈 彤 臧斌宇 朱传琪

(复旦大学并行处理研究所 上海 200433)

摘要 过程的处理在并行化编译工具中是十分关键的问题,过程嵌入和跨过程信息传播是常用的解决方法.近年来,兼有前二者优点的新技术:过程繁衍(Cloning),逐渐受到人们的重视.而以往的研究中,过程繁衍仅局限于常数值的传播.本文提出了在过程繁衍中进行符号等式约束信息传播的方法,该方法可以增强系统中全局的符号分析(Symbolic Analysis)能力,并可与一些新技术(如Omega测试)互相配合,从而提高并行化系统的能力.该方法在作者开发的并行化编译工具AFT中得到了实现.对于Perfect Benchmark的测试结果表明了此方法是有效的.

关键词 过程繁衍,数据相关性分析,过程间分析,信息传播.

并行计算机是新一代超高速计算机系统的主流,为充分发挥并行机的潜在性能,要求所运行的程序具有充分的并行性.并行编译工具帮助用户开发程序中的并行性,从而有效利用处理机资源.高级语言强调程序的模块性,因此过程被广泛采用,这成为并行编译工具不可回避的问题,不仅要并行化含有过程调用的循环,还需要跨越过程获取更精确的信息,用于判断过程内的循环能否并行化.解决这个问题一般有2种方法:过程嵌入(Inline)和过程间信息传播.[1,2]

过程嵌入是一个比较直观的方法,它用过程的体变形后替换对该过程的调用语句.当调用无圈(如FORTRAN所规定的那样)时程序中所有的CALL语句都被嵌入,这样就只有一个主程序,没有子过程,因此就没有过程间分析的问题了.过程嵌入看起来简单有效,但实际上有不少缺点:首先,把所有过程都嵌入主程序可能导致程序长度按调用深度指数级的膨胀,占用空间过大而难以处理,同时使得并行化分析、编译等操作的开销急剧增加.其次,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、不易维护,难以成为交互性的工具.第3,有些情况,如数组不匹配等,过程嵌入在实现上比较困难且烦琐.

而过程间信息传播的方法可以克服上述困难,它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.[3]它对过程仅作一次处理,这样效率就提高了.但如果该过程有多个被调用点,不同的实参可能带来不同的信息,这

* 本文研究得到国家自然科学基金和国家863高科技项目基金资助.作者丁永华,1971年生,硕士,主要研究领域为并行编译.陈彤,1968年生,讲师,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1995-10-30收到修改稿

样过程内部的相关性在每次被调用时也可能是不一样的.传统的过程间分析只能抽取共有的性质,因此要损失精确度.

鉴于这 2 种技术的特点,文献[4]提出了一种更好的方法:过程繁衍(Cloning).当出现不同调用点有不同的信息时,该调用点产生一个被调用过程的副本,这样不仅保留了过程,还保留了调用点的实参信息,在这个基础上进行过程间信息传播可得到更好的结果.相对嵌入而言,过程副本仅产生在有特殊调用信息处,是需求驱动.同时,不同调用点如果调用信息相同,还可合并,因此实际中程序膨胀不如嵌入那么厉害,虽然从理论上讲最坏情况都一样.

文献[4]不仅详细介绍了过程繁衍的概念,还给出了进行过程繁衍的 3 个步骤:(1)传播对以后相关性分析有价值的跨过程信息.(2)合并具有相同实参信息的过程副本.(3)根据调用点不同的实参信息产生相应过程的多个版本,也就是进行不同的程序变换.其中的跨过程信息主要指调用点的常数信息.

过去能处理的调用点信息只是常数信息,如常数作实参,或通过常数传播后,实参是常数,在不同的调用点常数值不一样,就需作过程繁衍.为了进一步提高并行编译工具的并行化能力,必须加强符号分析能力,为此我们提出了基于符号等式关系的过程繁衍方法.该方法可在过程间传播符号等式关系,与向前替代,Omega 测试等技术相互配合,增强了符号分析能力.[5]

由于文献[4]对过程繁衍技术有比较全面的介绍,本文着重讨论了如何求得调用点符号等式关系的算法,并结合过程繁衍的其它方面进行了讨论,该方法在我们开发的自动并行化系统 AFT 中得到了实现(AFT 系统在功能上已大大超越了传统的自动并行化系统,达到了国际一流水平,详细情况可参见文献[6]),通过对 Perfect Benchmark 的测试结果可以看到本文方法的实际效果.

1　背景和我们的工作

从下面的例子来解释什么是过程繁衍.

```
CALL SUB(10,20,30)        S1
CALL SUB(10,20,5)         S2
CALL SUB(L,N,L+N)         S3

SUBROUTINE SUB(K1,K2,K3)
DO 10 I=1,K2
  A(I+K1)=A(I+K3)
10 CONTINUE
END
```

上述例子中的过程 *SUB*,里面有一个 DO 循环,能否并行化呢?我们先看 *S*1 与 *S*2 这 2 个调用语句,*S*1 和 *S*2 各给 *K*1,*K*2,*K*3 一组常数值,我们可将这些常数值传播给 *SUB*,将 *S*1 调用点的常数信息传播给 *SUB* 后,可判定数组 *A* 的读写区间不重叠,因此 *SUB* 中的循环里面不存在相关性,而在 *S*2 调用点,传播了常数信息后,可判定过程 *SUB* 的循环中数组 *A* 存在跨循环的流相关,从而我们衍生出并行化和不并行化的不同 *SUB* 的版本.

对 *S*1 和 *S*2,传播的信息都是常数,而对于 *S*3 这类情况,按传统方法仅传播常数就不能处理,因此我们提出了一个更广泛的过程繁衍方法.如果实参之间有一定的关系,如 *K*3

$=K1+K2$,也可进行过程繁衍,我们将该实参信息传播给过程 SUB 后,数组 A 的读区间为 $[1+K1+K2,K1+2K2]$,而写区间为 $[1+K1,K1+K2]$,由符号分析可判定读写区间不重叠,没有相关性,因此 SUB 在调用点 $S3$ 时仍可并行化,所以我们的工作主要是求出实参之间的关系,并使之适用于过程繁衍.

下面是该程序作过程繁衍后的变换结果.

```
CALL SUB1(10,20,30)
CALL SUB2(10,20,5 )
CALL SUB3(L,N,L+N)

SUBROUTINE SUB1(K1,K2,K3)
DOALL 10 I=1,20
      A(I+10)=A(I+30)
10  CONTINUE
    END

SUBROUTINE SUB2(K1,K2,K3)
DO 10 I=1,20
   A(I+10)=A(I+5)
10  CONTINUE
    END

SUBROUTINE SUB3(K1,K2,K3)
DOALL 10 I=1,K2
      A(I+K1)=A(I+K1+K2)
10  CONTINUE
    END
```

因为当前数据相关性分析方法主要依据数组的下标或者循环的界限来判定,而且对非线性表达式的处理还缺乏有效的方法,所以我们仅考虑实参是线性表达式的情况,试图求出并传播实参间线性的等式约束,即把一个实参表示为另一些实参的线性表达式.

2 寻找实参之间关系的算法

我们算法的目的是根据给定的实参表达式序列,尽量发掘这些实参之间的关系,即从所给的实参表达式序列中找出一组基表达式,然后将其它实参表达式由基表达式线性表示.这样我们就找到了实参之间所有的线性关系式,传播这些线性约束,使得作过程繁衍的过程副本有更多的信息进行数据相关性分析.

为了尽量使比较简单的实参表达式作为基,将我们感兴趣的实参(线性表达式并且其对应的虚参可能出现在过程的数组下标或者循环界限中)按简单到复杂进行排序,变量数越少越简单,越多越复杂,变量数相同的实参表达式排列先后看变量系数和的大小,系数和小的排前面.我们将排序之后的这些实参存放在数组 P 中.

在寻找基表达式集时,用变量系数的列向量来表示实参表达式,根据实参表达式排列的次序以及变量出现在表达式中的前后来安排变量系数在向量中的位置,如实参序列为 X,Y,Z,$X+Y$,则其对应的列向量分别为 $(1,0,0)^T$,$(0,1,0)^T$,$(0,0,1)^T$,$(1,1,0)^T$.在这里不考虑常数项,诸如 X,$X+1$ 我们认为作为基它们是相同的,在后面计算其它实参表达式由基表达式线性表示的组合系数时,我们再考虑常数项(组合系数的向量中增加一常数项).

设实参表达式个数为 n,全部变量个数为 m,则表示第 i 个实参表达式的列向量为$(a_{i1}, a_{i2}...a_{im})^T$,其中 a_{ij}是第 j 个变量的系数,$1\leqslant j\leqslant m$,$1\leqslant i\leqslant n$.

n 个列向量组成矩阵 A 为

$$\begin{pmatrix} a_{11} & a_{21} & \cdots & a_{n1} \\ a_{12} & a_{22} & \cdots & a_{n2} \\ \cdots & & & \\ \cdots & & & \\ \cdots & & & \\ a_{1m} & a_{2m} & \cdots & a_{nm} \end{pmatrix}$$

应用线性代数中的消去法可以求出基向量,即在矩阵 A 中从第 1 行开始,寻找该行具有非零元素的一列,并将其它列上该行的元素消为零,直到最后一行,但在寻找第 $i(i>1)$行的非零元素时,应该从第 i 列开始,前 $i-1$ 列不找. 最后那些元素不全为零的列向量就是基向量.

在算法中,将实参表和实参的个数作为参数传入子程序. 下面给出算法:

```
FIND_RELATION(N,P)      /* 实参存放在数组 P 中,N 是实参的个数 */
BEGIN
    SORT(N,P);     /* 将实参排序 */
    BN <- FIND_BASE(N,P,BS);     /* 寻找基表达式,将其存放在 BS 中,并返回基表达式个数 */
    IF (BN<N)
      将其它实参表达式用基集 BS 表示,并记录表示信息.
    ENDIF
END

SORT(N,P)
BEGIN
    FOR I=0,N-1
        X[I] <- P[I]的变量个数;
        Y[I] <- P[I]变量系数的和;
    ENDFOR
    按数组 X,Y 中元素值的大小顺序排列 P;     /* X 元素相同看 Y 元素的大小 */
END

FIND_BASE(n,P,BS)
BEGIN
    m <- 全部变量的个数;
    A <-n 个实参表达式的列向量;     /* A 就是前面提到的矩阵 */
    FOR I=1,m
        FOR J=I,n
                IF (A[I,J] <> 0)
                  将其它列上该行的元素消为零;
                ENDIF
        ENDFOR
    ENDFOR
    FOR I=1,n
        FOR J=1,m
            IF (A[J,I] <> 0)
              将该列表示的实参表达式加入基集 BS 中;
              break;
            ENDIF
        ENDFOR
```

```
        ENDFOR
    END
```

将基表达式集 BS 求出之后，其它实参表达式就可以由基集 BS 中的表达式线性表示. 设基表达式集 $BS=\{x_1,x_2,\cdots,x_q\}$，设

$$x_i=a_{i1}y_1+a_{i2}y_2+\cdots+a_{im}y_m+a_{i(m+1)} \quad (i=1,2,\cdots,q) \tag{1}$$

其中 $a_{i1},a_{i2},\ldots,a_{im}$ 是变量 $y_1,y_2,\ldots,y_m$ 的系数，$a_{i(m+1)}$ 是常数项. 设某一非基表达式

$$x=r_1y_1+r_2y_2+\cdots+r_my_m+r_{m+1} \tag{2}$$

那么该实参表达式 x 由基表达式集 BS 中表达式 $x_1,x_2,\cdots,x_q$ 线性组合的问题，也就是求解系数 $k_1,k_2,\ldots k_{q+1}$ 使得 $k_1x_1+k_2x_2+\cdots+k_qx_q+k_{q+1}=x$ (3)

由(1)—(3)式我们可以得出

$$\begin{pmatrix} a_{11} & a_{21} & \ldots & a_{q1} & 0 \\ a_{12} & a_{22} & \ldots & a_{q2} & 0 \\ \cdots\cdots & & & & \\ \cdots\cdots & & & & \\ a_{1m} & a_{2m} & \ldots & a_{qm} & 0 \\ a_{1(m+1)} & a_{2(m+1)} & \ldots & a_{q(m+1)} & 1 \end{pmatrix} \begin{pmatrix} k_1 \\ k_2 \\ \cdots \\ k_q \\ k_{q+1} \end{pmatrix} = \begin{pmatrix} r_1 \\ r_2 \\ \cdots \\ r_m \\ r_{m+1} \end{pmatrix}$$

记 $\begin{pmatrix} a_{11} & a_{21} & \ldots & a_{q1} & 0 \\ a_{12} & a_{22} & \ldots & a_{q2} & 0 \\ \cdots\cdots & & & & \\ \cdots\cdots & & & & \\ \cdots\cdots & & & & \\ a_{1m} & a_{2m} & \ldots & a_{qm} & 0 \\ a_{1(m+1)} & a_{2(m+1)} & \ldots & a_{q(m+1)} & 1 \end{pmatrix}$ 为 A

$(k_1,k_2,\ldots,k_{q+1})$ 为 K^T，$(r_1,r_2,\ldots,r_{m+1})$ 为 R^T，则 $AK=R$，根据前面讨论 A 是列满秩阵，A^TA 的逆阵存在，所以由 $A^TAK=A^TR$ 可得

$$K=(A^TA)^{-1}A^TR \tag{4}$$

在计算所有非基表达式由基表达式线性表示的组合系数时，矩阵 A 都是相同的，所以(4)式中的 $(ATA)^{-1}A^T$ 只要计算 1 遍.

如前面的例子，实参表达式为 $L,N,L+N$，则其对应的列向量为 $(1,0)^T,(0,1)^T,(1,1)^T$，在寻找基向量时，只有第 3 个向量最后被消为零向量，所以基集由第 1、第 2 个实参表达式组成. 最后求解第 3 个实参 $(I+J)$ 由第 1、2 个实参线性表示的组合系数，相应的矩阵 A 为 $\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$，即单位矩阵 I_3，R 为 $(1,1,0)^T$. 这样由(4)式求得 $K=(1,1,0)$，也就是说第 3 个实参是第 1、2 个实参的和(常数系数为 0).

将所有非基实参表达式由基集中表达式线性表示的组合系数求出后，我们就找到了实参之间的所有线性约束信息，把这些信息保存在作 Cloning 的过程副本中，为使以后数据相关性分析更加精确提供了一定的信息. 在对过程作 Cloning 时，具有相同实参信息的过程副本可以合并成一个过程副本. 文献[4]中对常数传播具有相同信息的过程副本进行合并作了详细的介绍，这里常数传播作为特例，主要解决相同等式信息的合并，其方法也是类似的，即

如果几个调用点有完全相同的等式约束信息,那么就可以把它们合并起来.

3　试验结果分析

本文提出的方法在我们开发的 FORTRAN 程序自动并行化编译器(AFT)中得到了实现,经过对 Perfect Benchmark 13 个程序的测试,我们发现作 Cloning 后,有 3 个程序取得明显效果,其中 TISI. F 的过程 OLDA 里面标号为 100 的 1 个三重循环串行执行时间占 60%,作 Cloning 后最外层循环也可以并行化,而不作 Cloning,只能并行化里面两层(该子程序只有 1 个调用点). LWSI. F 的过程 INTERF 里面标号为 1 000 的 1 个三重循环串行执行时间占 92%,作 Cloning 后在所有的 2 个调用点上都可以并行化,而不作 Cloning 均不能并行化,其另一个过程 BNDRY 里面标号为 100 的一重循环作 Cloning 后可并行化,不作 Cloning 也不能并行化. 还有一个程序是 OCSI. F,过程 FTRVMT 内标号为 109 的 1 个三重循环串行执行时间占 45%,作 Cloning 后可以并行化(全部的 4 个调用点),不作 Cloning 都不能并行化.

表 1 列出了这 3 个程序在 SGI Challenge 4L 共享主存四处理机服务器上 AFT 封闭 Cloning 功能后和作 Cloning 以及 PFA(power fortran accelerator,是国际公认的代表当前最高水平之一的自动并行编译系统)的加速比(串行执行时间/并行执行时间).

表 1

程序名	PFA 加速比	AFT 封闭 Cloning 加速比	AFT 作 Cloning 加速比
LWSI. F	1.01	1.02	3.87
TISI. F	0.35	0.96	1.59
OCSI. F	1.00	1.19	1.18

AFT 对 OCSI. F 作 Cloning 后的实际执行效果并不理想,这是因为程序并行化后存在 Cache 抖动(Thrashing)问题,如果消除这个影响,我们相信该程序作 Cloning 也是有效果的. 而 PFA 的过程间分析(包括 Cloning)能力比较弱,所以对这 3 个程序都没有效果.

表 2 列出了 Perfect Benchmark 13 个程序作 Inline 和 Cloning 后程序膨胀的比较. 其中长度是指程序的行数,不包括一些注释行和未被调用的子程序. 从表中我们可以发现作过程繁衍的程序膨胀远小于过程嵌入,这也是过程繁衍的一大优点.

表 2

程序名	源程序长度	Inline 长度	Cloning 长度
APSI. F	6 805	64 891	7 863
CSSI. F	18 465	6 613 239	35 093
LGSI. F	2 816	17 574	2 936
MTSI. F	3 910	6 483	4 129
NASI. F	4 518	8 092	4 932
OCSI. F	3 163	39 454	3 437
SDSI. F	8 264	17 511	11 006
SMSI. F	3 470	77 681	9 578
SRSI. F	4 582	6 254	5 950
TFSI. F	2 339	4 695	2 693
TISI. F	398	489	441
WSSI. F	4 590	58 058	6 591
LWSI. F	1 445	2 224	1 713

4 结束语

本文基于过程间分析的Cloning技术中过程副本对实参信息的依赖，给出了一个寻找实参之间关系的方法，把问题归结为从一列实参表达式中找出一组基，然后将其它实参表达式用基表达式线性组合，用求解线性方程组的方法来解出这个组合系数.这样得到的实参之间的联系是一些相等关系表达式，为提高数据相关性分析的精确性提供了一定的信息.从对我们系统的测试结果来看，这种方法是比较有效的.但根据程序上下文得到的实参之间关系更多的可能是一些不等关系，而且数据相关性分析在很多情况下只是用到这些不等关系，因此根据程序上下文提取实参之间这些不等关系也是过程间分析的一个重要方面，这将是我们以后工作要解决的问题.

参考文献

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

2 Dale Allan Schouten. An overview of interprocedural analysis techniques for high performance parallelizing compilers. CSRD Rpt. NO. 1005.

3 Callahan D, Cooper K, Kennedy K *et al*. Interprocedural constant propagation. Proceedings of the ACM SIGPLAN'86 Symp. on Compiler Construction, SIGPLAN Notices, 1986,**21(6)**:152～161.

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

5 Burke M, Cytron R. Interprocedural dependence analysis and parallelization. Proc. of the ACM SIGPLAN'86 Symposium on Compiler Construction. ACM SIGPLAN Not., 1986,**21(7)**:162～175.

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

CLONING AND ITS IMPLEMENTATION

Ding Yonghua Chen Tong Zang Binyu Zhu Chuanqi

(*Institute of Parallel Processing Fudan University Shanghai* 200433)

Abstract Interprocedural analysis is one of the critical issues for parallelizing compilers. Traditionally, inline and interprocedural information propagation are the two methods to solve the problem. Recently, a new technique called cloning shows its advantage. Cloning creates specialized copies of procedure body to allow distinct call site inherits an environment context that allows for better code optimization. Previous research focused on constant propagation. This paper presents a method that can propagate the information of symbolic linear equations. The capability of cloning is enhanced by the new method. The authors implemented their scheme in AFT, a parallelizing compiler that they developed. The test result illustrates the effectiveness of this method.

Key words Cloning, data dependence analysis, interprocedural analysis, information propagation.