一个新的强化别名分析算法*

金国华 陈福接

(长沙工学院,长沙 410073)

摘要 相关分析和程序并行化等技术已普遍受到重视,现有技术对过程内的相关分析和并行化效果是令人满意的,但是过程调用的出现为分析增添了许多困难.过程间相关性分析的目的正是为了开发含过程调用情况的程序的并行性.本文在深入研究别名对程序并行化影响的基础上,提出了一个新的强化别名分析算法,使被调用过程段的并行化成为可能.

关键词 别名,过程调用,并行化,数据相关性.

为满足应用领域对计算机性能要求的不断提高,计算机结构高度并行势在必行.然而要充分发挥硬件的潜在并行能力,关键还要有足够多的程序并行性.依靠过程内的相关性分析和现有的并行化技术能开发出一部分并行性[1-4].但仅仅局限于过程范围,过程调用的出现使程序的分析精度受到了严重的影响.由于缺乏必要的过程间信息,相关性分析往往过于保守,存在于调用过程和被调用过程中的大量并行性得不到开发.过程间相关性分析通过获取和传递过程间信息,能有效地提高程序的分析精度,开发含过程调用情况下程序的并行性,其中包括:(1)含CALL直接式程序段的并行性;(2)含CALL循环的并行性;(3)被调用过程的并行性.

1. 含call直接式程序段优化

如图1所示,在没有任何过程间信息的情况下,我们只能假设子程序plus对所有调用参数和变量作读写访问,从而程序段必须以串行方式执行.然而,如果知道了子程序仅对第一个参数写访问,而对其它参数皆为读访问,则语句S_1、S_2和循环L_1之间不存在相关关系,可以并行执行.

```
        ……
S1:     CALL PLUS(C,A,B,N)
S2:     CALL PLUS (D,A,B,N)
L1:     DO I=1,N
            SUM=SUM+A(I)
        ENDDO
        ……
        SUBROUTINE PLUS(X,Y,Z,N)
L2:     DO I=1,N
```

* 本文1991-08-24收到,1992-03-16定稿

作者金国华,30岁,博士生,主要研究领域为并行处理,超级计算.陈福接,58岁,教授,博士导师,主要研究领域为并行处理,超级计算,电子技术,计算机系统设计,逻辑设计和工程实现.

本文通讯联系人:金国华,长沙 410073,长沙工学院6系

```
      X(I)=Y(N-I+1)+Z(N-I+1)
    ENDDO
    RETURN
    END
```

图1　含CALL直接式程序段优化

2. 含CALL循环优化

受过程调用影响最大的莫过于循环优化. 循环优化对整个程序的并行化起着非常关键的作用. 当循环中出现了过程调用时,被调用过程的访问信息是判别循环能否并行化的主要依据之一,从图2,我们不难发现循环是完全可以并行化的,然而前提是我们已知道对MAX的每次调用只访问A的第I行.

```
      ……
      DO I=1,N
S1:       A(I,N+1)=MAX(A,I,N)
      ENDDO
      ……
      REAL FUNCTION MAX (X,I,N)
      MAX=0
      DO J=1,N
          IF(X(I,J).GT.MAX) MAX=X(I,J)
      ENDDO
      RETURN
      END
```

图2　含CALL循环优化

3. 被调用过程的优化

程序中,除主过程以外的所有执行过程都可以看成是某一过程的被调用过程,被调用过程的优化质量直接影响着整个程序的优化效果.

为了对被调用过程进行优化和转换,别名和常数传递信息是必须的. 在图1中,只要Y、Z和X不互为别名,循环L_2是可以并行执行的.

本文将主要讨论别名对程序并行化的影响,并提出开发被调用过程并行性的一些思想. 首先我们将给出有关别名的概念,然后介绍现有别名分析技术. 最后,我们提出已有技术所存在的问题并给出解决这些问题的一种新的强化别名分析方法.

1　概　念

别名——当两个或多个名字同指一个存储单元或一片存储区域时,我们说它们互为别名. 一般来说,别名可分三类:显式别名,参数别名和指针别名.

· 显式别名——由某种程序结构显式说明的别名. 如FORTRAN中的EQUIVALENCE语句和C中的UNION语句都具有这种功能,不难发现图3中数组A、B、C互为别名,x、y、c、f互为别名.

```
FORTRAN :  EQUIVALENCE A(1,1),B(1,1),C(10)
      C :  union {int x;
                  int y;
                  char c;
                  float f;
                 }
```

图3　FORTRAN和C中的显式别名

·参数别名——一个实参同时传给两个或多个不同的形参所引起的两形参间的别名或

全局变量作为实参传给形参所引起的全局变量和形参间的别名. 如图4所示,X、Y 互为别名,因为 CALL 语句 S 使它们同时指向数组 A;Z 和全局变量 GV 互为别名,因为由 COMMON 语句所指的存储单元既可用 GV 来访问,也可由 Z 来访问. 另外,我们必须注意别名和结合(呸实结合)不同. 结合发生于实参赋于形参时,一个对象可和两个或多个不同的名字结合,但如果这些名字永远不会被同时访问,就无别名可言. 在图4中,X,Y 都和 A 结合,但它们和 A 不互为别名,因为 A 在子程序 P 中不可能被访问.

```
      COMMON GV
            …
S:    CALL P(A,A,GV)
            …
      SUBROUTINE P(X,Y,Z)
      COMMON GV
            …
```

图4　FORTRAN 中的参数别名

·指针别名——两个不同的指针直接或间接地指向同一对象而引起的别名,只要两个不同的指针可能包含相同的值,它们就可能间接地指向同一对象. 图5就是其中一例. 用于步进数组 a 的指针 p、q 互为别名.

```
    …
p=&a
for (q=p++;*p=0;q=p++)
    f(*q,*p);
  …
```

图5　C 中指针别名

本文主要讨论参数别名,即形参和全局变量的别名关系和两个或多个形参间的别名关系.

2　现有技术

现有的别名分析方法,对于非循环调用图的处理极其简单. 首先,通过检查每个调用点是否将多个同名的变量或将全局变量作为参数传递以确定由主过程体引起的所有可能参数别名,所有这些别名对在每个被调用过程中予以标出,然后对于每个被调用过程作同样分析并考虑当两个别名又作为实参传递时可能引起的别名,直至分析完整个调用图,标出程序中所有可能的别名. 对于循环图(含递归调用),情况就不是这样简单了. 对图6所示例子,如采用上述算法,则所检测出的别名将随着分析过程次序的变化而变化. 如 P 在 Q 前分析,则测出〈a,b〉,〈x,y〉别名对;相反,如 Q 在 P 前分析,则仅能发现〈a,b〉别名对. 而实际上 x、y、z 互为别名,同样,a、b、c 也互为别名. 为此,Banning 提出了图7所示算法[5].

```
SUBROUTINE P(x,y,z)
CALL Q(x,x,y)
        …
SUBROUTINE Q(a,b,c)
IF(…) THEN
        CALL P(a,b,c)
ENDIF
        …
```

图6　循环调用图中的别名

```
//* Input: a program with a call graph C=(N,E).
  Output: a set S of alias pairs containing all aliases in the program.
  P and Q are procedures. W is a worklist of procedures.
  a_i and a_j are actural reference parameters at a call
  site e=P→Q which are bound to formal reference
  parameters f_i^Q and f_j^Q.
  S is the set of alias pairs *//
begin
    initialize W to contain all procedures
    initialize S to contain (v,v)∀ variable v
  while W is not empty
      S_old=S
      remove procedure P from worklist W
      for each call site e=P→Q
        ∀ (a_i,a_j) ,i≠j do
          if (a_i,a_j)∈S then
            S=S ∪{(f_i^Q,f_j^Q)}
        enddo
        ∀ a_i do
          if a_i is global then
            S=S ∪{(f_i^Q,a_i)}
          else if (a_i,g)∈S for some global g then
            S=S ∪{(f_i^Q,g)}
        enddo
      endfor
      if S_old≠S then add Q to W
  endwhile
end
```

图7 检测别名的 Banning 算法

Banning 算法采用了迭代的方法精确计算了出现递归调用时的别名信息，工作表 W 中保存着需分析的所有过程. 当过程 P 从 W 中消去时，检查每一调用点 e=P→Q 看是否或因为同名变量赋给两个或多个不同的参数或因为两个不同的但可能互为别名的变量传给 Q 而引起两个或多个不同参数的别名. 后一种情况通过检查别名对集合 S 确定. 对调用点 e 处实参对的每一种组合$\langle a_i,a_j\rangle$，$i\neq j$，检查 a_i 和 a_j 是否是同名变量或$\langle a_i,a_j\rangle\in S$(两个别名的参数传给 Q)，设 f_i^Q 和 f_j^Q 分别为对应于实参 a_i 和 a_j 的形参，如果 a_i 和 a_j 为同名变量或$\langle a_i,a_j\rangle\in S$，则 f_i^Q 和 f_j^Q 也可能互为别名，这样必须将$\langle f_i^Q,f_j^Q\rangle$加入 S. 除检查不同形参间的别名关系外，还要检查形参和全局变量间的别名关系，如果 a_i 为全局变量，则 f_i^Q 和 a_i 互为别名，如果 a_i 和全局变量 g 互为别名(通过检查 S 中别名对)，则 f_i^Q 和 g 互为别名. 如果我们已知这些别名关系，则不往 S 中加入任何新的别名信息，但是如果我们以前不知道它们互为别名，则 Q 必须因这一新的信息而重新分析，这是通过加 Q 到 W 中来完成的(假设 Q 已不在 W 中). 每次新别名的发现都将受影响的过程加入工作表. 当工作表为空时，工作完成，所有可能的别名对都包含在 S 中. Cooper[6]进一步对使用传地址形式参数引起别名作了专门讨论，并提出了别名传递的一个数据流分析框架，使标准全局数据流分析算法同样适合于传递问题的求解. 因篇幅关系，我们就不详细介绍了.

3 强化别名分析

所有传统别名分析方法有一个共同的特点,那就是忽视了数组变量中的下标细节,分析结果是不完全的.不必要的数据相关性的引入,使被调用过程并行性的开发非常局限,如图8所示,参数B、C和同一实参A有地址关联,传统分析技术认为B、C互为别名,这样,因为可能存在跨迭代相关而不能对循环并行化.然而,如果我们再仔细分析一下下标就不难发现B、C实际上分别对应数组A的不同列,故不存在重叠,B、C在过程P中是两个完全不同的变量.它们在DO循环中的访问之间不可能存在数据相关性.

```
...
DIMENSION A(100,100)
...
CALL P(A,A(1,100),50)
...
SUBROUTINE P(B,C,N)
DIMENSION B(N),C(N)
...
DO I=3,N
...=C(I)
...
B(I)=...
ENDDO
RETURN
END
```

图8　假别名

直接式扩展方法回避了专门的别名分析,对扩展后代码进行分析,识别存在于含CALL直接式程序段,含CALL循环及被调用过程中的并行性.然而它具有不可忽视的缺点.首先,对过程调用的扩展要求被调用过程代码到调用程序的严格转换,有时这是无法做到的,况且递归调用通常很难处理;其次,扩展消除了程序的模块化特性,使结果代码难读难维护;再次,每个调用点都需扩展使代码段极大膨胀,分析量急剧增大;最后,所有过程内的局部变量扩展后都将变成全局变量,这样又可能会引起额外的数据相关,减少并行性.以程序并行化为目的的强化别名分析在分析数组名的同时也分析数组下标细节,识别出真假别名,以进一步开发被调用过程的并行性.在给出强化别名分析算法之前,我们先引进概念如下:

·子程序的调用点——引起某子程序执行的调用点为该子程序的调用点.

·对应调用点的原始别名实例,原始别名实例的原始调用点——在某调用点由于将两个或多个相同的变量作为参数传递或将全局变量作为参数传递所引起的被调用过程中形参间的别名或形参和全局变量间的别名被称为对应该调用点的原始别名实例,并称该调用点为这一原始别名实例的原始调用点.

·对应调用路径的传递别名实例,传递别名实例的原始调用点——在以某原始调用点为起点的某一调用路径上由两个或多个别名作为实参传递所引起的被调用过程中形参间的别名或由和全局变量别名的变量作为实参所引起的被调用过程中形参和全局变量间的别名被称为对应该调用路径的传递别名实例,并称该调用路径的原始调用点为这一对传递别名实例的原始调用点.

·潜在别名,潜在原始别名实例,潜在传递别名实例——传统别名识别算法(按名判别)识别出来的别名被称为潜在别名,相应地,称上面定义的原始、传递别名实例为潜在原始别名实例和潜在传递别名实例.

·真原始别名实例,假原始别名实例——作为实参的全局变量(标量或数组)所引起的潜在原始别名实例;调用点实参是标量变量的潜在原始别名实例;调用点实参是数组元素或数组名的,而且传递给两个或多个不同形参的实参数组区域相交的潜在原始别名实例是真原始别名实例.不是真原始别名实例的潜在原始别名实例是假原始别名实例.

·真传递别名实例,假传递别名实例——一潜在传递别名实例,如果在其原始调用点,它们所对应的实参区域间或实参和全局变量区域间相交,则是真传递别名实例,否则是假传递别名实例.

·真别名实例,假别名实例——真原始别名实例和真传递别名实例是真别名实例,假原始别名实例和假传递别名实例是假别名实例.

·对应调用点的局部真别名,对应调用点的局部假别名——我们称一个子程序的两个形参或形参和全局变量对应该子程序的某调用点局部真别名如果它们存在对应于到达该调用点的某条调用路径的真别名实例(真原始别名实例或真传递别名实例),否则它们对应该调用点局部假别名.

·全局真别名,全局假别名——一个子程序的两个形参或形参和全局变量互为全局真别名如果存在这个子程序的一个调用点(我们假设任何一个子程序调用点都是可执行到的),它们对应该调用点局部真别名;否则,它们互为全局假别名.

显然,由假别名实例(假原始别名实例或假传递别名实例)引起的传递别名实例一定是假别名实例,而由真别名实例(真原始别名实例或真传递别名实例)所引起的传递别名实例却未必是真别名实例.互为全局真别名的变量可能不是局部真别名(对于某调用点来说),但互为局部真别名一定互为全局真别名;相应,互为局部假别名不一定互为全局假别名,但互为全局假别名一定互为局部假别名.如图9所示,过程P中形参B、C对应调用点e_1的假原始别名实例引起过程Q中形参D、E对应调用路径e_1—e_3和e_1—e_4的假传递别名实例;过程P中形参B、C对应调用点e_2的真原始别名实例引起过程Q中形参D、E对应调用路径e_1—e_3的真传递别名实例和对应调用路径e_1—e_4的假传递别名实例.P中形参B、C因为它们对应调用点e_2互为局部真别名而互为全局真别名,但它们对应调用点e_1互为局部假别名;R中形参G、H互为全局假别名.

```
        ...
        DIMENSION A(10,10)
        ...
e1:     CALL P(A(1,1),A(1,2))
        ...
e2:     CALL P(A,A)
        ...
        SUBROUTINE P(B,C)
        DIMENSION B(10),C(10)
        ...
e3:     CALL Q(B,C,10)
        ...
e4:     CALL Q(B,C(6),5)
```

```
        …
e5:     CALL R(B(1),C(6))
        …
        RETURN
        END
        SUBROUTINE Q(D,E,N)
        DIMENSION D(N),E(N)
        …
        RETURN
        END
        SUBROUTINE R(G,H)
        DIMENSION G(5),H(5)
        …
        RETURN
        END
```

图9　真假别名实例,局部真假别名和全局真假别名

互为全局假别名的变量之间不存在数据相关性,不影响子程序中并行性的开发.互为局部假别名的变量之间相对其对应的调用点不存在数据相关性,不影响子程序在该调用点的并行性开发.即使存在真别名,也不应该因此而放弃被调用过程中所潜在的并行性.别名的出现虽使问题更加复杂化,然而这并不意味着不能向量化或并行化.我们认为并行性检测(或相关性分析)阶段的任务应该是:提供尽可能准确的相关信息,发现一切可开发的并行性,而被调用过程中的并行性和调用过程中的并行性同样不可忽视.为了做到这一点,我们进一步将原来的数组别名信息细化,为每一对潜在别名对〈x,y〉增加一个标志位 $TB_{x,y}$以标志它们是否全局真假别名,$TB_{x,y}=1$表示 x、y 为全局真别名,否则为全局假别名;一个标志位向量 $TV_{x,y}$以标志它们是否在某调用点处局部真假别名,$TV^{e}_{x,y}=1$($e\in E$,E 为 x、y 所在子程序的调用点集合)表示 x、y 在 e 调用处局部真别名,$TV^{e}_{x,y}=0$表示 x、y 在 e 调用处局部假别名;和一组别名相对位移向量 $OV_{x,y}$分别表示每个局部真别名的别名程度,$OV^{e}_{x,y}=\{offset_i \mid i=1,\cdots,n\}$,n 为 x、y 在调用点 e 处局部真别名的不同别名相对位移的个数},如果 e 引起 x、y 真原始别名,则别名相对位移向量 $OV^{e}_{x,y}$仅含一个相对位移 offset,它由直接计算 e 处实参的下标相对位置差得出,如果 e 引起 x,y 真传递别名,则 $OV^{e}_{x,y}$可能含多个相对位移 $Offset_i(i=1,\cdots,n)$,每个相对位移 $Offset_i$ 都是 e 处对应实参的别名相对位移量和实参下标相对位置差之和.

为了获得这些别名信息,我们提出了强化别名算法.

```
//* nput:程序、调用图 c=(N,E)和主程序 MP *//
Output:标志有强化别名信息的程序
begin
    Intensive--alias(MP)
end

Intensive-alias(P)
//* ai,aj 是调用点 e=P→Q 的实参,Nai,Naj分别是 ai,aj 的变量名,fp 和 fp 是在 e 处和 ai,aj 结合的形参 *//
begin
    for 每个调用点 e=P→Q
      ∀ 〈ai,aj〉,i≠j do
        if Nai=Naj then
```

```
        计算 f_i^P, f_j^P 的别名相对位移;
        if f_i^P 和 f_j^P 在 N_{a_i} 和 N_{a_j} 上相应区域相交 then
          TV_{i,j}^{P,Q}=1, TB_{i,j}^{P,Q}=1;
          将 f_i^P, f_j^P 的别名相对位移加入 OV_{i,j}^{P,Q}(即调用点 e 处的别名相对位移向量)
      elseif TB_{a_i,a_j}=1 then
        (∀ e_1)(TV_{a_i,a_j}^{e_1})→由 OV_{a_i,a_j}^{e_1} 中的别名相对位移计算出 f_i^P, f_j^P 在 e 处的别名相对位移;
        if f_i^P, f_j^P 在对应原始调用点的实参区域相交 then
          TV_{i,j}^{P,Q}=1, TB_{i,j}^{P,Q}=1;
          f_i^P, f_j^P 在 e 处的别名相对位移加入 OV_{i,j}^{P,Q}
    enddo
    ∀ a_i do
      if a_i 是全局变量 then
        TV_{i,a_i}^{P,Q}=1, TB_{i,a_i}^{P,Q}=1;
        计算 f_i^P 和 a_i 的别名相对位移加入 OV_{i,a_i}^{P,Q}
      elseif(∃ g),(g 是全局变量且 TB_{a_i,g}=1)then
        (∀ e_2)(TV_{a_i,g}^{e_2}=1)→由 OV_{a_i,g}^{e_2} 中别名相对位移计算出 f_i^P, g 在 e 处的别名相对位移;
        if f_i^P, g 在对应原始调用点的实参区域相交 then
          TV_{i,g}^{P,Q}=1, TB_{i,g}^{P,Q}=1;
          f_i^P, g 在 e 处的别名相对位移加入 OV_{i,g}^{P,Q}
    enddo
  enddo
  // * 分析每个被调用过程 * //
  for 每个过程 Q(P→Q ∈E)
    Intensive-alias(Q)
  end
end
```

算法只考虑了非循环调用情况,对于循环调用图我们可以再作进一步扩充.

有了这些强化别名信息,我们可以判别一个被调用过程是否存在全局真假别名,通过收集并分析其调用点的别名程度结合被调用过程中的数据相关性分析找到一种一致的方法来向量化或并行化被调用过程段,对过程段进行向量化或并行化改写.在这种一致方法由于调用点执行环境的不同所能开发的并行性有限的情况下,我们可以分析被调用过程段各调用点的情况,针对各调用点开发其并行性,如图10所示.图10(a)中有两处调用子程序 P.经分析,尽管 P 中形参 B、C 对应两个调用点均局部真别名,第一次调用,P 中循环无跨迭代相关,故可转换成并行循环 DOALL,第二次调用虽引入了跨迭代相关,但所有相关均为向下相关(即按词法序在后的语句依赖于在前的语句)满足向量化条件,可表示成向量指令形式.转换后的结果见图10(b)所示.

```
...
DIMENSION A(100,100)
...
CALL P(A(2,100),A(1,100),20)
CALL P(A(1,100),A(1,100),50)
...
SUBROUTINE P(B,C,N)
DIMENSION B(N),C(N)
DO I=3,N
    ...=C(I)
    B(I-1)=...
ENDDO
RETURN
```

a) 转换前代码

```
…
DIMENSION A(100,100)
…
DOALL I=3,20
    …=A(I,100)
    A(I,100)=…
ENDDOALL
…=A(3:50,100)
A(2:49,100)=…
…
```

b)转换后代码

图10　借助强化别名分析开发被调用过程的并行性

4　结束语

开发最大可能的并行性是并行性分析的主要目的,虽然过程内相关性分析取得了一些令人满意的,然而过程调用的引入使调用过程和被调用过程的并行性开发非常困难.别名分析,作为过程间相关性分析的一个重要组成部分,正是为了尽量开发被调用过程的并行性.本文深入研究了别名对程序并行化的影响,提出了强化别名分析的方法,使被调用过程段的并行化成为可能,被调用过程段往往是执行频率较高的程序段,对它的并行性开发是非常有意义的.然而它又是复杂的,还有待作进一步广泛而深入的研究.

参考文献

1　Polychronopoulos C D, Girkar M, Haghighat M R *et al*. Parafrase—2: an environment for parallelizing, partitioning, synchronizing and scheduling programs on multiprocessors. ICPP'89, 1989.

2　Sridharam K, Mcshea M, Denton C *et al*. An environment for parallel structuring of Fortran programs. ICPP'89, 1989.

3　Allen F E *et al*. An overview of the PTRAN analysis system. ICS'87, 1987.

4　Allen J R, Kennedy K. PFC:a program to convert Fortran to parallel form. TR82—6, Rice Univ., 1982.

5　Banning J. A method for determining the side effects of procedure calls. Ph. D. thesis, Stanford Univ., 1978.

6　Cooper K D. Analyzing aliases of reference formal parameters. Proc. of the 11th Acm Symp. on Prin. of Programming Languages, 1984.

7　Allen F E, Burke M, Cytron R *et al*. A framework for determining useful parallelism. ICS'88, 1988.

8　Banerjee U. Dependence analysis for supercomputing. Kluwer Academic Publishers, 1988.

9　Kuck D, Davidson E S, Lawrie D H *et al*. Parallel supercomputing today and the cedar approach. Science 231 (4740) Feb. 28, 1986:967—974.

10　Padua D A, Kuck D J, Lawrie D H. High—speed multiprocessors and compilation techniques. IEEE Trans. on Computers, 1980,**C—29(9)**.

11　Polychronopoulos C D. Parallel programming and compilers. Kluwer Academic Publishers, 1988.

12　Padua D A, Wolfe M. Advanced compiler optimizations for supercomputers. CACM, 1986,**29(12)**.

13　Triolet R, Irigoin F, Feautrier P. Direct parallelization of call statements. ACM SIGPLAN 86 Symp. on Compiler Construction, 1986.

14　Weihl E W. Interprocedural data flow analysis in the presence of pointers, procedure variables and label variables. 7th ACM Symp. on Prin. of Programming Languages, 1980.

A NEW ALGORITHM FOR INTENSIVE ALIAS ANALYSIS

Jin Guohua and Chen Fujie

(*Department of Computer Science, Changsha Institute of Technology, Changsha* 410073)

Abstract Although much more attention have been paid, data dependence analysis and program parallelization become very difficult when the analyzed program has procedure call. Based on thorough study of the effect of alias on program parallelization, in this paper, they present an algorithm which intensively analyzes the aliases, and as the result, they make the parallelization of the called procedures possible.

Key words Alias, procedure call, parallelization, data dependence.

中国计算机学会1994、1995年国际学术活动计划

会议名称	时间	地点	主办单位	联系人
The 3rd Pacific RIM International Conference on Artificial Intelligence 第三届太平洋地区国际人工智能会议	8.16—8.18	北京	CCF CAA	史忠植 2565533—419 Fax. 2567724
世界华人计算机科学技术及产业研讨会	8.28—9.1	北京	CCF CAS	李光华 2560911
HKICC(GZ)	10.3—10.4	广州	CCF HKCS	王介生 2565533—818
ICA'94展览会	10.6—10.10	上海	CCF SCS B/I	徐桂珍 3726055
国际软件测试和软件可靠性研讨会	10.18—10.20	北京	IEEE SE 专委 CCF 容错专委	闵应骅 2565533—836
The 19th. Computer Software & Applications Conference 第19届国际计算机软件及应用会议	1995.10	北京	IEEE CS CCF	朱明远 6756956 Fax. 4919213
ICYCS'95 第四届国际青年计算机学术会议	1995.8	北京	CCF 主办 智能中心承办	白硕 2565533—162
第四届CAD & CG 国际学术会议	1995.10	武汉	CCF 主办 华中理工大学 (430074) 承办	华中理工大学 机一系 刘健