一个交互式的Fortran77并行化系统*

陈文光 杨 博 王紫瑶 郑丰宙 郑纬民

(清华大学计算机科学与技术系 北京 100084)

E-mail: yangbo@est4.cs.tsinghua.edu.cn

摘要 并行化编译器可以把现有的串行程序自动或半自动地转换为并行程序.现有并行化系统的自动并行化效果与手工并行化的效果相比还有一定的差距,这是由于并行化工具的分析能力不足以及程序中所固有的语义信息无法被并行化工具所理解而造成的.TIPS(Tsinghua interactive parallelizing system)系统通过提供一些友好的交互式工具,使用户与编译器紧密协作,是提高并行化系统的能力和效率的一条有效途径.

关键词 并行化编译器,交互式系统,相关性查询,增量编译,性能预测.

中图法分类号 TP311

从80年代中期开始,随着并行计算机系统的发展,人们逐渐开始研究串行程序并行化系统.最初的研究是从扩充程序的自动向量化系统开始的,典型的例子有KAP和VAST.但是,在实际应用程序中,这些自动并行化系统的并行化效果并不理想[1].其原因在于,自动并行化与自动向量化有着本质的区别.并行化需要中粗粒度的并行性,而向量化需要细粒度的并行性.在分析粗粒度并行性时会遇到过程调用语句和符号量,使得传统的相关性分析方法遭到失败.

近年来,人们在并行化理论和实用方法两方面都取得了一些进展,并出现了一些新的并行化系统.如UIUC(University of Illinois at Urbana-Champaign)的Polaris[2,3],Stanford大学的SUIF(Stanford University intermediate format)[4],复旦大学的AFT(automatic Fortran transformer)[5]等.

这些新一代的全自动并行化工具,通过采用过程间分析、符号数据相关性分析、数组私有化、归约识别、复杂形式的归纳变量识别以及运行时分析等新技术,并行化能力与KAP等传统系统相比有了较大提高,对某些程序的并行化效果已经与手工并行化相近.但是,还有一些程序的自动并行化效果与手工并行化的效果相比还有很大差距.其原因在于,在全自动并行化系统中使用的并行化算法还不能有效地处理这些复杂应用程序.除了算法本身的能力不足以外,缺乏有关的程序语义信息更是全自动并行化算法的障碍.

Greenwich大学的CAPTools[6~9],Rice大学的Fortran D系统[10,11]和Applied Parallel Research公司的Forge90等均是交互式的并行化系统.

这些交互式并行化系统,在尽可能采取自动并行化技术的同时,允许人在并行化过程中查看和修改并行化结果,通过利用人的能力来提高并行化效果,在部分程度上弥补了全自动并行化系统的不足.但是,这些交互式的工具普遍都没有采用最新的自动并行化技术,因此,它们的自动并行化的能力比较差.

TIPS(Tsinghua interactive parallelizing system)是一个交互式的并行化系统,该系统以Polaris系统为基础,具有强大的自动并行化底层支持.TIPS具有友好的交互功能,并同时支持共享存储系统和分布存储系统.

---

* 本文研究得到国家863高科技项目基金资助.作者**陈文光**,1972年生,博士,主要研究领域为并行编译技术.**杨博**,1976年生,博士生,主要研究领域为并行编译,网络计算.**王紫瑶**,女,1974年生,硕士,主要研究领域为并行编译.**郑丰宙**,1976年生,硕士生,主要研究领域为并行编译,网络计算.**郑纬民**,1946年生,教授,博士生导师,主要研究领域为并行分布处理.

本文通讯联系人:杨博,北京100084,清华大学计算机科学与技术系系统应用教研组

本文1998-11-06收到原稿,1999-02-01收到修改稿

1 TIPS特点与结构

1.1 系统特点

(1) 提供友好的交互手段.以用户容易理解的方式向用户显示和说明并行化各个阶段的结果,紧密结合用户的知识(通过向用户查询),指导进一步的并行化.

(2) 使用先进的自动并行化技术.TIPS是基于Polaris开发的.Polaris系统中采用了数组私有化、归纳变量识别、归约变量识别、非线性相关性测试、符号分析和循环变换等多项最新的自动并行化技术.但是,在Polaris系统中缺少重要的过程间分析技术,而我们在Polaris系统上则补充了过程间分析模块.

(3) 同时支持共享存储系统和分布存储系统.

1.2 系统结构

我们设计的用于机群系统的Fortran77程序并行化系统称做TIPS,其结构如图1所示.

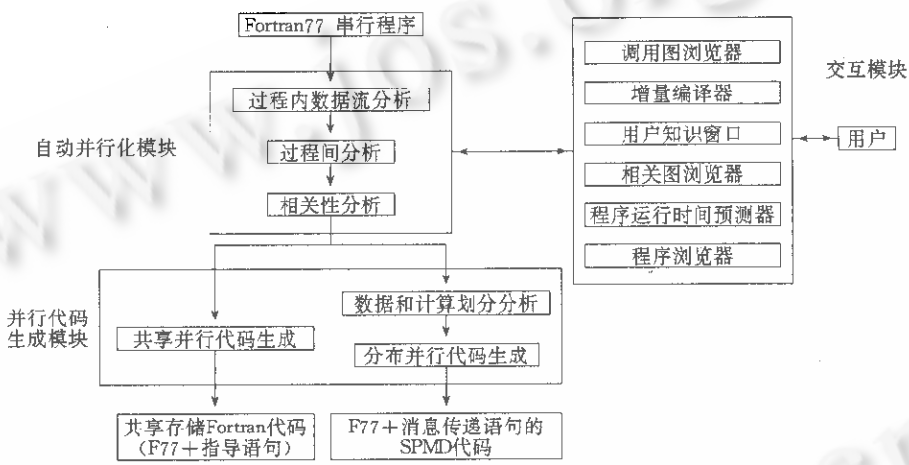

图1 TIPS系统结构图

1.2.1 自动并行化模块

自动并行化模块是TIPS系统的基础,它分为以下几部分.

(1) 过程内数据流分析

过程内数据流分析为相关性分析作准备.该阶段进行的操作包括常数传播(constant propagation)、标量私有化(scalar privatization)、归纳标量识别(inductive variables recognition)和归约识别(reduction recognition)等.

(2) 过程间分析

过程间分析阶段进行的操作包括过程繁衍、过程间数据流分析、过程间数组私有化以及过程间变换等等.分析出的过程间信息可以帮助相关性分析阶段对含有过程调用的循环进行并行化,也是增量编译器的基础.

(3) 数据相关性分析

只有不相关的语句才可以在没有同步的情况下并行执行,因此相关性分析实际上是对程序并行性的识别.相关性分析通常由一系列测试来完成,在TIPS中采用的测试有GCD(greatest common divisor)测试、Banerjee测试、Omega测试和Range测试.本系统中扩展了Range测试,可以把一类原先无法处理的相关性问题归结为一些不等式,通过交互模块向用户查询,系统根据用户的回答来确认循环是否可以并行化.

1.2.2 并行代码生成模块

(1) 共享代码生成

共享代码生成模块根据相关性分析、私有化和规约识别等模块的结果,在程序中插入相应的指导语句,并对程序进行相应的变换.所生成的并行Fortran程序在目标机上使用本机编译器就可以生成并行的执行代码.目

前，本系统的共享代码生成模块支持 SGI Power Challenge SMP 系统.

(2) 数据和计算划分分析

面向分布存储系统的程序必须显式地对数据和计算进行划分. 本模块的主要功能是对数据访问模式进行分析，尽量减少并行程序所需的通信次数和通信量，从而提高并行程序的性能.

(3) 分布并行代码生成

分布并行代码生成就是指利用分析出的并行性以及数据和计算划分方案，进行实际的数据和计算划分，然后生成通信语句. 在并行代码生成时把相应的代码插入到源程序中，生成 SPMD 的结点程序. 在本系统中生成基于 PVM 的并行代码. 并行源代码经过串行 Fortran 编译器编译，生成的执行代码在 PVM 系统的支持下就可以进行并行计算了.

1.2.3　交互模块

交互模块负责自动并行化模块与用户的信息交互. 主要任务有显示并行化结果、向用户说明该并行化结果的原因以及向用户询问程序的语义信息等. 该模块的主要组成部分如下：

(1) 调用图浏览器

显示程序的调用图. 图中的结点可以是循环或子程序. 调用图中每个结点后都跟着其执行时间在整个程序中所占的比例. 循环节点后还显示该循环是否可以并行，如果是串行循环，在调用图中还可以显示其串行的原因. 用户仔细查看程序后，如果认为这些原因是系统能力不足所造成的，可以方便地强制并行化该循环.

调用图浏览器与交互模块的其他工具紧密结合，使用户可以迅速而准确地了解程序的概况，并可以确定程序中的重要部分(运行时间长，而且无法并行的循环)，进一步对程序进行深入的研究.

(2) 增量编译器

用户在使用交互式并行化工具的时候，可能会对程序作一些修改，如手工进行一些程序变换等. 由于并行化系统一般执行时间比较长，如果在用户对程序稍加修改后还要对整个程序进行完全的重新编译，会浪费用户的很多时间，降低并行化的效率.

增量编译器通过分析，找出用户修改所影响到的程序单元，并只对这些单元进行重新编译，从而增强了系统的交互性.

(3) 用户知识窗口

在相关性分析阶段，对一些无法解决的问题，系统提取了一些不等式. 在用户知识窗口中，系统向用户询问这些不等式，用户可以回答“成立”、“不成立”或“不一定”. 系统根据用户的回答，确认相应的循环是否可以并行化.

对用户而言，系统所询问的不等式大多很容易理解，而且不难回答. 而系统通过结合用户的知识，可以得到更好的并行化效果.

(4) 相关图浏览器

相关图浏览器显示一个循环嵌套的相关区. 系统提供了多种过滤方法，使得用户可以集中于对某些相关问题的研究.

(5) 程序运行时间预测器

程序运行时间预测器可以给出每个循环的运行时间在整个程序中所占的比例. 这样就使得用户能够集中精力分析程序中那些重要的部分. 对运行时间少的循环，即使它不能并行化也不必花费太多的精力.

对程序运行时间进行预测有动态和静态两种方法. 动态法通过在源程序中加入测量时间的语句并实际运行程序来获得运行时间. 静态法通过分析程序的各种操作次数，无需运行程序就可以估计出程序各部分的运行时间. 动态法的优点是比较准确，缺点是比较费时(如程序规模比较大时)，而且有时候无法运行程序(例如，缺少输入数据). 使用静态法可以避免上面的情况. 一般来说，静态性能预测的时间与程序的长度成正比，而与问题的规模无关，其响应速度比较快，但准确性稍差.

在本系统中，我们提供了动态和静态两种方法. 用户在对程序的运行时间进行预测时，可以根据自己的需要任选一种方法.

(6) 程序浏览器

程序浏览器使得用户可以方便地浏览源程序和并行化之后的程序.在程序浏览器、相关图浏览器和调用图浏览器之间有着关联关系,在任一浏览器中选中某一循环或子程序时,其他浏览器的内容都会随之同步.这大大方便了用户对程序的研究和分析.

2 关键技术

2.1 交互技术

交互技术是交互式系统的特色,它把系统的各种分析算法与用户结合起来,从而形成功能强大的并行化系统.要让用户与分析算法进行有效的协作,必须能够在用户和算法之间有效地进行双向信息流动.交互式系统使得算法的并行化结果可以被人理解,并使得人所提供的信息可以被并行化算法利用.另外,交互式系统还应该对用户响应时间给予一定的关注,尽量减少用户操作所需的时间,提高用户的工作效率.

具体地说,在并行化过程中,交互模块应该能够完成下面的任务.

(1) 向用户显示并行化分析结果(用相关图和程序调用图的方式).用户可以查看产生某个相关性的原因,并且可以修改相关图.这就使得用户可以考察产生相关性的程序段.用户可能发现,这个相关实际上是由于算法能力不足而造成的伪相关,因此可以直接删除该相关;也可能发现,修改(使用某些程序变换技术或手工重写)该段程序可以消除相关.

(2) 向用户提出相关性分析算法在分析中所无法解决的一些问题,并利用用户的回答来改进分析结果.由于所提出问题的数量有可能十分巨大,所以要特别注意问题的选择.需要提出的是出现次数较多的问题,因为这类问题对并行化效果的影响可能较大.

(3) 提供程序各部分运行时间之比,使用户可以把注意力集中在并行化的关键步骤上.

(4) 增量编译功能,即在用户修改了程序之后只对受到影响的程序单元进行重新编译,而无需对整个程序进行重新编译.

(5) 程序浏览功能,该功能可以帮助用户方便地找到所要分析的程序段(如运行时间最长的循环,或者含有相关的循环).

程序浏览功能、相关图和程序调用图显示等功能所需的技术比较简单,限于篇幅,在本文中略去.对交互技术中3项比较复杂的技术:增量编译技术、静态程序性能预测技术和相关性查询技术,下面作一个简单介绍.

2.1.1 增量编译技术

在TIPS系统中的增量编译器是以子程序为编译单位的.其基本原理是:增量编译器通过分析源程序得到了一些过程内和过程间信息(用一些集合表示).当用户修改了程序后,增量编译器首先通过比较找出修改了的程序单元,通过重新计算这些集合,可以找出受到本次修改影响的其他程序单元.这些程序单元将被重新编译.

2.1.2 静态程序性能预测技术

本系统中静态程序性能预测技术的基本思想是:预先在目标机上运行一些基准程序,获得一些基本操作(如+,-,*,/,...)的开销;在预测一个程序的性能时,统计该程序内部各种基本操作的个数,程序运行的时间就是这些基本操作所需时间之和.

由于没有考虑有关存储器层次的信息(寄存器、Cache和内存的访问时间的差异),可能会导致性能预测和实际运行的结果有很大的差异.若要解决这个问题,需根据程序的执行过程模拟存储器的行为,这样做,开销过大.但是在TIPS系统中,使用静态程序性能预测的目的并不是为了得到准确的执行时间,而是为了得到程序各个部分的相对执行时间和整个程序的复杂程度,因此这种性能预测技术在我们的系统中是适用的.

我们的性能预测算法包括训练集的构造和测试、语法分析和时间预测计算这3部分.

通过构造和运行训练集,可以获取基本操作和内部函数在目标机上的运行时间.语法分析阶段的功能是分析源程序并构造抽象语法树,对程序进行相应的变换,以便进行后面的分析.时间计算模块根据上面两个模块的结果来计算时间.

2.1.3　相关性查询技术

相关性查询可以简单地把不能解决的相关问题都保守地估计为相关,输出相关图,然后由用户自己来分析程序,手工消除不存在的相关关系,从而达到并行化的目的.本系统也提供了这种功能.但是,这种方法对用户而言不够友好,因为相关关系的分析有时并不那么直接,而且程序中相关的数目众多,用户难以分清哪些是真相关,哪些是假的相关.

我们扩展了 Range 测试的思想,对一类无法解决的相关性问题抽取出一些不等式,然后向用户查询这些不等式是否成立,通过用户的回答来确定相关是否存在.这种方法的优点在于既增加了查询本身的友好性,又减少了用户所需分析的代码的范围,可以把人的能力很好地结合进并行化过程中.

下面,我们介绍不等式的提取算法.

由 Range 测试可知,在下面的程序中,若 $j_n \leqslant N$,则 i 循环可以并行.

```
do i=1, in
    do j=1, jn
        A(i*N+j)=...
        ...=A(i*N+j)
    enddo
enddo
```

当系统无法确定 $j_n \leqslant N$ 是否可以并行时,就可以生成一个不等式,每个不等式对应于一个相关弧.在实际系统中,通常对下标表达式进行一些变换,因此提出的问题可能不像上面的程序那么直接,但是问题仍然是容易理解和回答的.

对循环中的每一对符合上述条件的相关产生了不等式后,需要以循环为单位对这些不等式进行归结.最后以循环为单位,向用户查询.

例如,Perfect Benchmark 中的程序 APSI 中有这样的程序段:

```
IPP2=IP+2
IPPH=(IP+1)/2

DO J=2, IPPH
    JC=IPP2-J
    J2=J+J
    DO K=1, L1
        CH(1,K,J)=CC(IP0,J2-2,K)+CC(IP2,J02,K)
        CH(1,K,JC)=CC(1,J2-1,K)+CC(1,J2-1,K)
    ENDDO
ENDDO
```

系统可以抽取出两个不等式,只要其中一个成立,就可以确定该循环是并行的.

(1) IP<2

(2) 2+IP>2*IPPH

用户通过 IPPH=(IP+1)/2 可以很容易知道不等式(2)成立,从而并行化循环 J.从这个例子可以看出,本方法提出的问题对用户来讲,非常容易理解,也比较容易回答.用户甚至不需要有相关性分析的知识就可以帮助系统进行并行化.

类似的循环,在 Perfect Benchmark 中还有很多.我们初步分析了 13 个 Perfect Benchmark 中的 10 个程序,其中有 5 个程序可以提出有效的问题.因此可以认为,本方法具有一定的普遍适用性.

2.2　过程间分析技术

过程间分析技术是一项十分重要的技术,它使得含有过程调用不再成为并行的障碍,也避免了过程嵌入所

带来的代码爆炸现象.在TIPS系统中,过程间分析还是增量编译器的基础.本系统中实现的过程间分析技术主要包括过程间数据流分析、过程间相关性分析、过程间私有化和循环嵌入等.限于篇幅,本文略去了过程间分析的算法.

2.3 分布并行代码生成技术

TIPS系统的分布并行代码生成模块按并行执行模型把程序划分为若干区(region):

Region 1

...

Region 2

...

Region N

Region可以是一个并行循环,也可以是一段串行程序,Region内部没有通信,Region之间在需要的情况下可以进行通信以获得计算所需的数据或是计算结果.一般而言,在并行Region结束后要进行全局数据同步,这样,每个Region在开始执行的时候都拥有全部正确的数据.对并行循环来说,系统通过划分循环实例空间来进行计算划分,在多个结点上并行执行.数据空间没有直接划分,各结点都拥有完整的数据空间.这样做的好处是消除了地址变换操作和缓冲区管理,方便了串行程序执行,缺点则是占用内存比较大.程序的串行部分由各个节点进行冗余执行,从而减少了串行执行所带来的通信开销.I/O操作由于是在串行部分完成的,也是由各个结点冗余执行的,这样就减少了由主控结点读入数据再进行分发的通信开销.

TIPS分布代码生成模块采用的技术包括数据对准分析、临时数组消除、通信合并等,目前可以生成PVM+F77代码.限于本文的篇幅,我们略去这些算法.

3 性能测试

3.1 静态性能预测器性能测试

为了测试估计器的性能,我们进行了一系列的测试.所有的实验都是在一台拥有双CPU的Ultra 2上进行的,操作系统是Solaris 2.5.

3.1.1 准确性测试

我们使用了普渡大学的一系列程序对该预测器的估计准确性进行了测试,虽然这些例子比较短小,但是也反映了实际应用程序的特点.结果如表1所示.

表1 准确性测试结果

程序	实测时间(s)	预测时间(s)	(预测-实测)/实测*100(%)
problem01.f	1.44	1.58	9.7
problem02.f	0.46	0.46	0
problem03.f	6.42	3.57	-44.4
problem04.f	3.87	4.45	15.0
problem05.f	3.62	2.71	-25.2
problem07.f	1.86	1.82	-2.2
problem08.f	1.39	0.79	-43.2
problem09.f	0.94	0.55	-41.5

通过上面的实验可以看出,估计的误差局限在-50%~20%之间.在通常情形下,估计值将小于实际的运行时间,这主要是由于存储器访问的不确定性引起的.由于训练模块通过多次迭代的方法得到操作的平均执行时间,因此,存储器的访问基本上是对Cache进行的.而对于实际的应用程序来说,在Cache不命中的情况下,需要从内存或者甚至外存进行调页,因此其实际的执行时间将大于预测的时间.如果在一个应用程序中,当用于计算的时间比用于内存访问的时间大得多时,预测的结果将会比较准确.

3.1.2　确定重要循环

性能预测可以指导用户将注意力集中在一些比较耗时的循环上，尽量将这些循环进行并行化，以更大地提高加速比. 在静态预测时，有时虽然总的预测时间与实测时间相差较大，但是程序中各部分所占比例的差别却不是很大. 我们以 purdue-set 中的 problem09.f 各循环占全部程序运行时间的比例为例，见表 2.

表 2　各个循环预测和实测的运行时间所占比例的比较

循环	实测时间(s)	比例(%)	预测时间(s)	比例(%)
DOIT do#1	0.211	21	0.134	24
DOIT do40	0.310	31	0.237	43
DOIT do50	0.001	0	0.001	0
DOIT do60	0.305	30	0.130	24
DOIT do75	0.118	12	0.046	8

从上面的结果可以看出，实测和预测的结果在换算为各个循环所占的比例后相差较小，因此，使用性能预测可以很好地确定重要的循环，指导应用程序的并行化.

3.2　分布代码生成性能测试

我们测试了 purdue-set 中 14 个 Benchmark 程序在全自动并行后再进行自动分布代码生成的性能，运行环境是 8 台带 2 个 CPU 的 Sun Ultra2，操作系统是 Solaris 2.5，使用 100M 共享以太网连接. 串行运行时间是在 1 个CPU 下运行的时间，并行运行时间是在 8 台工作站上使用 8 个 CPU 运行的时间. 因为这些问题的运行时间都比较短，我们把每个问题运行 10 次取平均值作为其执行时间，结果见表 3.

表 3　分布式代码生成的测试结果

程序	串行运行时间(s)	8 结点并行运行时间(s)	加速比	说明
problem01.f	3.0	0.7	4.29	
problem02.f	6.4	1.3	4.92	
problem03.f	20.7	3.8	5.45	
problem04.f	—	—	—	结果不正确
problem05.f	1.3	8.2	0.159	
problem06.f	50	170	0.185	
problem07.f	1.8	2.0	0.9	
problem08.f	0.5	7	0.07	
problem09.f	—	—	—	编译未通过
problem11.f	—	—	—	编译未通过
problem12.f	0.4	—	—	并行后运行时间过长
problem13.f	2.1	0.6	3.5	
problem14.f	—	—	—	未发现并行性
problem15.f	—	—	—	编译未通过

在上述测试结果中，“编译未通过”是因为我们使用的底层自动化编译器 Polaris 不能顺利编译这些测试程序. 可以看出，对某些程序，TIPS 分布代码生成模块可以生成满意的并行代码，但是对另一些程序，生成的并行代码效率很低，甚至不正确. 这是因为分布代码生成模块生成的通讯过多，或是没有考虑到某些特殊情况造成的.

3.3　增量编译器性能测试

由于增量编译器是在用户修改程序后进行重新编译，因此用户修改程序的方式会极大地影响增量编译器的性能. 为此，我们只进行了简单的测试，测试结果表明，修改程序中的一个循环界后采用增量编译器比完全重新编译大约减少 50%的编译时间.

4　将来的工作

我们计划在以下几个方面进行更加深入的研究.

· 静态性能预测.静态性能预测目前不够准确,其主要原因是循环界估计不准确和未考虑 cache 的影响.我们计划采用动态/静态结合或是向用户查询的方法获得更加精确的循环界值.对于 cache 的影响,我们正在评估各种能够处理 cache 的性能模型.

· 查询.在目前的 TIPS 中,系统只对一类 Range 测试不能解决的相关问题进行查询.我们计划扩展查询的使用范围,包括考察其他适用于查询的相关性测试算法,或把查询技术应用于数组私有化、分布代码生成等方面.

· 增量编译.我们将考察用户修改程序的模式,并针对这些模式作特殊的处理,以达到更快的增量编译速度.

· 分布代码生成器.目前生成的分布代码在每个结点上都拥有全局数据,因此不适用于需要内存较大的程序.我们将采用新的分布程序模型,使得每个结点可以拥有小于全局数据空间的局部数据空间.同时,我们还要研究通信优化算法,并加强系统的稳定性.

5 小 结

我们认为,交互式系统是进行提高并行化质量的一个有前途的方向.TIPS 系统以最新的自动并行化编译器为基础,开发了友好的交互工具,可以有效地帮助用户进行程序并行化工作.

参考文献

1 Blume William, Eigenmann Rudolf. Performance analysis of parallelizing compilers on the perfect benchmarks programs. IEEE Transactions on Parallel and Distributed Systems, 1992,3(6):643~656

2 Blume William, Doallo Ramon *et al*. Parallel programming with polaris. Computer, 1996,29(12):78~82

3 Blume William, Doallo Ramon *et al*. Advanced Program Restructuring for High-Performance Computers with Polaris. Technical Report, 1473, University of Illinois at Urbana-Champaign, Center for Supercomputing Research & Development, 1996

4 Amarasinghe S P, Anderson J M *et al*. The SUIF compiler for scalable parallel machines. In: Bailey D H, Gilbert J R, Mascagni M *et al* eds. Proceedings of the 7th SIAM Conference on Parallel Processing for Scientific Computing. Philadelphia: Society for Industrial and Applied Mathematics, 1995

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
(Zhu Chuan-qi, Zang Bin-yu Chen Tong. An automatic parallelizer. Journal of Software, 1996,7(3),180~186)

6 Ierotheou C S, Johnson *et al*. Computer aided parallelization tools (CAPTools)——conceptual overview and performance on the parallelization of structured mesh codes. Parallel Computing, 1996,22(2):163~195

7 Johnson S P, Ierotheou C S *et al*. Automatic parallel code generation for message passing on distributed memory systems. Parallel Computing, 1996,22(2):227~258

8 Johnson S P, Cross M *et al*. Exploitation of symbolic information in interprocedural dependence analysis. Parallel Computing, 1996,22(2):197~226

9 Leggett P F, Marsh A T J *et al*. Integrating user knowledge with information from parallelization tools to facilitate the automatic generation of efficient parallel FORTRAN code. Parallel Computing, 1996,22(2):259~288

10 Hiranandani Seema, Kennedy Ken *et al*. Compiler optimization for Fortran D on MIMD distributed-memory machines. In: Proceedings of the Supercomputing'91. Washington, D C: IEEE, ACM, 1991. 86~100

11 Hiranandani Seema, Kennedy Ken *et al*. The D editor: a new interactive parallel programming tool. In: Proceedings of the Supercomputing'94. Washington, D C: IEEE, ACM, 1994. 733~742

An Interactive Fortran77 Parallelizing System

CHEN Wen-guang　YANG Bo　WANG Zi-yao　ZHENG Feng-zhou　ZHENG Wei-min

(*Department of Computer Science and Technology　Tsinghua University　Beijing*　100084)

Abstract　Parallelizing compiler can transform serial programs to parallel programs automatically or semi-automatically. The up-to-date automatic parallelizing systems cannot generate parallel codes as good as hand coding for many applications. The main reason is that they cannot manipulate the complexities of real applications. What's more, there are some semantic information in the program that the automatic tools can never know without user's knowledge. TIPS (Tsinghua interactive parallelizing system) provides some user-friendly interactive tools so that the compiler and user can cooperate with each other. It is an effective way to improve the capability and efficiency of parallelizing compiler.

Key words　Parallelizing compiler, interactive system, dependence query, incremental compilation, performance estimation.