ISSN1000-9825 Journal of Software 软件学报 2000,11(9):1268～1275

自动并行编译新技术*

阳雪林 于勐 陈道蓄 谢立

(南京大学计算机软件新技术国家重点实验室 南京 210093)

(南京大学计算机科学与技术系 南京 210093)

E-mail: yxl@dislab.nju.edu.cn

摘要 自动并行编译为并行化现有的串行程序及编写新的并行程序提供了重要的支持,因此20多年来一直受到重视.近几年来,自动并行编译技术的研究进展,包括在依赖关系分析、程序变换、数据分布和重分布及调度等方面的进展,将自动并行编译进一步推向了实用化.该文介绍了自动并行编译技术的最新进展,并提出了进一步的研究所要解决的问题.

关键词 自动并行编译,依赖分析,程序变换,数据分布,调度.

中图法分类号 TP314

自动并行编译为充分发挥并行计算环境不断增强的计算能力提供了一条重要的途径.它有以下优点:

(1) 在缺乏被普遍接受的并行程序设计语言的情况下,自动并行化工具能有效地解决代码的可重用和可移植问题;

(2) 解决了用一种语言难以显式地表达从指令层到任务层各个层次的并行性,并进行基于体系结构的优化问题;

(3) 自动并行化工具为大量存在于应用领域的,并经过长时间的设计、使用和测试的成熟大型串行应用程序的并行化提供了唯一可行的选择.

但设计自动并行化系统十分复杂.自动并行编译牵涉到数据依赖关系分析、程序变换、数据分布及调度等许多技术.如果说基于共享内存的自动并行编译技术已具有一定程度的实用性,基于分布内存的技术则由于数据分布的困难而离实用化还有相当一段距离.自90年代以来,对自动并行编译技术的研究有了很大的进展,产生了一批有代表性的自动并行化系统,国外的有SUIF(Stanford university intermediate format),Polaris;国内的有KD-PARPRO[1],FAT,Autopar[2]和JAPS[3].通过对这些系统的开发,人们提出了很多实用技术,涵盖了数据依赖关系分析、程序变换、数据分布及调度等方面的内容.下面依次对这些技术作简要的介绍.

1 依赖关系分析

循环体中的依赖关系分析和过程间依赖关系分析的质量决定了数据并行性和任务并行性的开发程度,多年来,它们一直是研究的热点.近几年的研究成果更是进一步地推动了自动并行的实用化.指针分析是对C,C++程序并行化不可缺少的工作.

1.1 循环体中的依赖关系分析

循环占用了串行程序的绝大部分执行时间,因此,对循环体的依赖关系分析一直是最受关注的[4,5].早期的

* 本文研究得到国家863高科技项目基金(No.863-306-ZT02-0301)资助.作者阳雪林,1967年生,博士生,助教,主要研究领域为并行与分布式计算.于勐,1972年生,博士生,助教,主要研究领域为并行与分布式计算.陈道蓄,1947年生,教授,博士生导师,主要研究领域为并行与分布式计算.谢立,1942年生,教授,博士生导师,主要研究领域为并行与分布式计算,计算机网络.

本文通讯联系人:阳雪林,南京 210093,南京大学计算机科学与技术系

本文2000-01-26收到原稿,2000-06-12收到修改稿

工作主要集中于数组是线性下标的精确方法和运行速度足够快的近似方法,分析也都是静态的.因此,在大多数情况下只能得到保守的结果.Omega Test[6]使用整数规划,综合得到了速度快且精确的结果.Range Test[7]突破了线性下标的限制.RPD(reduction privatizing doall) Test 和 LRPD(lazy reduction privatizing doall) Test[8]对依赖于输入、静态所不能确定依赖关系的循环进行了动态并行化.

1.1.1 Omega Test

Omega Test 的原理是:依赖关系分析可以归结为求解一组任意的线性等式和不等式集合的整数解,即整数规划的问题.传统上认为用整数规划来求解依赖关系代价过高,Omega Test 的速度则表明这种看法是不正确的.

Omega Test 由一种消去整数约束的新方法和 Fourier-Motzki 变量消去法的一种变形组合而成.它可以给出精确解.它的输入是形如 $\sum_{0\leqslant i\leqslant n} a_i x_i = 0$, $\sum_{0\leqslant i\leqslant n} a_i x_i \geqslant 0$ 的等式和不等式约束的集合,其中 $x_0 = 1$,定义 V 为变量的索引集合.它的工作步骤可分为 3 步:

(1) 将约束标准化.标准化的约束是指所有的系数是整数且所有系数的最大公约数(不包括 a_0)是 1 的等式和不等式.

(2) 消减等式约束.为消减等式 $\sum_{0\leqslant i\leqslant n} a_i x_i = 0$,首先检查是否存在 $j\neq 0$ 使得 $|a_j| = 1$.如果存在,我们解出 x_j 并代入其他约束以消减此约束.若不存在,则进行变换,以使至少一个变量的系数为 1.反复进行此过程则可以消除所有等式约束.

(3) 消减不等式约束.在所有等式约束消减后,即开始消减不等式.由于处理等式比处理不等式效率更高,若存在一对紧连(tight)不等式(如 $6\leqslant 3x+2y$ 和 $3x+2y\leqslant 6$),即用适当的等式约束取代,并用处理等式约束的方法来处理,删除冗余的不等式.若此时不等式只包含一个变量,则问题有解.否则,用扩展的 Fourier-Motzkin 方法继续处理.

Omega Test 可以产生一组描述了所有可能的依赖于距离矢量的解.虽然由于整数规划问题是 NP 完全问题,在最坏的情况下,Omega Test 具有指数时间复杂度,但很多情况下,在能用其他方法于多项式时间内精确求解的时候,Omega Test 具有低阶的多项式最坏时间复杂度.

1.1.2 Range Test

当前,绝大多数依赖关系分析方法要求循环边界和数组下标用循环索引变量的线性(仿射)函数表示,而不能处理循环边界和数组下标中出现非线性表达式的情况,因而对一部分程序不能进行分析.

Blume 的 Range Test[7]通过确定一组符号不等式是否得到保持的符号处理方法,可以处理循环边界和数组下标是符号和非线性表达式的情况.

Range Test 的原理如下:对一个给定的循环 L 中的迭代 i,将被存取的数组下标值域 range(i)看作一个符号表达式.如果我们能证明 range(i)是单调的,且与下一个迭代 $i+1$ 的数组下标值域不相重叠,则 L 不存在跨迭代依赖关系.对单调递增函数,只需证明 $\max(\text{range}(i)) < \min(\text{range}(i+1))$ 即可.对单调递减函数,只需证明 $\max(\text{range}(i+1)) < \min(\text{range}(i))$ 即可.

Range Test 用于 Porilas 编译器后,大大提高了对重要程序并行性的开发.

1.1.3 RPD Test 和 LRPD Test

依赖关系分析方法按分析的时刻可分为静态分析和运行时分析两种.静态分析方法用于内存存取模式静态确定的规则程序.运行时分析方法用于内存存取模式运行时才知道的不规则程序,以及由于内存存取模式的复杂而不能用目前的静态分析算法分析的内存存取模式静态确定的规则程序.

Rauchwerger 提出的 RPD Test 和 LRPD Test 都是在执行时确定循环能否完全并行化的方法.RPD Test 从循环中抽取一个观察(inspector)循环,通过它的执行来判断在考虑数组私有化和归约并行后循环能否并行化.观察循环的执行要求不能改变程序的状态,若不能并行化,则串行执行循环,即 RPD Test 要执行两个循环.

由于抽取观察循环的困难以及在某些情况下不可能在不影响程序状态的情况下执行观察循环,Rauchwerger 又提出了 LRPD Test.LRPD Test 不抽取观察循环,它首先保存程序状态,在执行循环的同时判定循环的可

并行性.循环执行结束后若发现循环不能并行,则恢复循环执行前状态,重新串行执行循环.

理论分析和实验结果表明,在充分利用静态信息的情况下,Rauchwerger 提出的动态并行化技术在能并行时,可得到可扩性好的高加速比;在不能并行时,也只招致较小的运行开销.

1.2 过程间的依赖关系分析

为了开发大粒度并行性以及全程优化,需要利用过程间依赖关系分析技术.过程繁衍(cloning)技术[9]提高了过程间依赖关系分析的精度和效率.流敏感的过程间数据流分析框架的引入,使得分析的精度和效率及可分析的程序的大小都得到了提高.精确而又易于操作的数组表示,为精确的过程间的数组引用分析奠定了基础.

1.2.1 过程繁衍

传统上有两种方式用于处理程序中的过程调用.

最简单的是过程嵌入(inline)的方法.过程嵌入是在过程被调用处直接加入过程体的一个副本.这种方法的优点是,对过程的优化可以根据调用点的上下文来进行.它的缺点是可能导致程序代码的过度膨胀及编译时间过分延长,而且一些过程(如递归过程)不能被嵌入.

第 2 种是上下文不敏感的过程间的数据流分析技术.在分析一个过程时,它将所有调用点信息的综合作为入口信息,因此,每个调用点所获得的被调用过程的信息将是一致的.采用这种技术的优点是只需为每一个过程生成一个实现(implementation).它的缺点主要是对具体调用点而言,信息是不精确的,影响过程和整个程序的优化.

Cooper 提出的过程繁衍技术[9]集中了这两种方式的优点.它的基本思想是,对源程序中的任意一个过程生成多个实现,根据过程调用的上下文对同一过程的不同调用点进行分类,每一个实现对应于一类调用点,根据各类调用点所获得的程序属性信息的不同,对过程施加不同的程序变换、优化措施,从而生成不同的实现.程序中的调用点将根据其上下文的属性信息来确定调用过程的某个实现.

此方法的优点一是过程副本少于过程嵌入,程序不致过度膨胀;二是可以有针对性地对过程实现进行优化.

1.2.2 流敏感的过程间数据流分析框架

传统的过程间数据流分析,或者是由于全部采用过程嵌入技术而难以分析大程序,或者是由于缺少特定路径的过程间信息而精度不够,或者是分析效率太低.Hall 等人提出的流敏感的过程间数据流分析框架[10]具有分析效率高、分析结果精确以及可以容易地分析大程序等特点,还集成了支持过程边界数组变形(array reshape)的公式.这主要是通过组合下面两项技术而得到的.

基于区域(region-based)的流敏感分析.为了获得精确的过程间信息,需要使用流敏感的分析方法,即根据程序中每一条可能的控制流路径导出分析结果.通过使用基于以表示嵌套关系为基础的区域图的分析,Hall 等人的框架解决了其他方法可能会出现的不可能存在的路径(unrealizable paths)的影响以及慢收敛问题.

选择性过程繁衍.对于程序中由多条路径激活的过程,Hall 等人的框架采用了选择性过程繁衍技术,即只有在能提供优化机会时才使用路径特定的信息.这样既可以得到与完全嵌入相同的精度,也避免了不必要的复制.

1.2.3 数组区域的表示

精确的数组的数据流概要分析首先要用合适的形式表达和描述数组区域.在选择表达方式时,要考虑表达的精确性、区域合并的能力以及检测区域相交的能力.

传统的描述方法都只描述单区域.由于合并操作的不封闭性,为了用单区域描述合并操作的结果,只能采用近似的方法,这会丢失概要信息.GAR's(guarded array regions)表示方法[11]使用规则数组区域的链表来表示访问区域,还为规则的数组区域增加了卫哨(guards),以更好地处理 IF 条件.GAR's 的使用为数组概要信息提供了更精确的表示,有利于数组私有化等优化的进行.

1.3 指针分析

指针分析的目的是在程序中的每一个语句位置识别出指针的可能值[12],以便为指针别名分析提供更精确的信息.目前,指针分析的研究主要集中于面向过程的语言,如 C,Fortran90.提出的方法主要有 Emami 等人的 Points-to 函数[13]、Wilson 等人的部分转换函数(partial transform function,简称 PTF)[12]等方法,以及以此为基础,黄波等人对指针向量的分析方法[14].

在 Emami 等人提出的方法中,由于对过程间分析是采用嵌入取代实现的,因而很难分析大程序. Wilson 等人在转换函数的基础上提出了部分转换函数的概念,即将每个过程的输入参数所确定的域上的 Points-to 函数部分映射到过程返回的 Points-to 函数. 由于这种映射不必是完全的,从而大大减少了映射函数的分析计算. 经验表明,一个过程一般只需要一个 PTF 即可得到完全的上下文敏感的结果.

2 程序转换

Eigenmann 等人通过手工并行化标准测试程序包所进行的大量实验研究[15]表明,数组私有化、归约操作并行化和广义归纳变量(generalized induction variables)替换是开发并行性最重要的程序转换技术.

2.1 数组私有化

标量或数组的私有化是指将标量或数组的拷贝分配于每一个处理节点. 私有化可以消除循环反依赖和输出依赖,开发并行性. 标量的私有化技术已被普遍采用,而由于确认可私有化数组的复杂性,数组私有化技术还未被普遍采用.

数组私有化判定的条件是简单的:如果数据项在循环迭代中定值先于引用,则可以私有化. 为了发现这些信息,需要对数组或数组块的定值和引用进行复杂的分析[10,11]. 由于可私有化的数据结构常常是数组的一部分,它的范围一般由标量变量的值确定,确定这些标量变量的值常常需要进行过程间分析,包括对变量的值、变量间关系、关系保持的条件的分析及过程间常量传播、符号值传播[7]. 分析的精确程度也与编译器对数组区域的表示能力有关[11]. 因此,数组私有化能力的开发是紧密依赖于过程间分析的能力的.

2.2 归约操作并行化

归约变量 x 是指循环体中这样的一个变量,它出现在形如 $x=x\oplus \exp$ 的结合操作(associative operation)中,其中$\oplus$是结合操作符,并且 x 不出现在表达式 exp 或循环的其他位置. 归约变量引起的数据依赖妨碍了循环的并行性开发. 归约变换可删除由归约操作所引起的数据依赖.

开发归约并行性需要做两步工作:识别归约变量和并行化归约操作.

归约变量的识别过程是,首先将循环体中语句与归约模板进行语法匹配,然后对候选变量进行数据依赖分析.

并行化归约操作可用以下几种不同的方式来实现:

(1) 原位同步. 每一归约操作用 lock/unlock 对加以保护. 这种变换需要改变的代码最少.

(2) 私有化并行归约. 这种方法将部分和加入循环私有的变量,这些变量以后用于在临界段中更新初始变量. 使用私有部分和变量提高了局部性,但仍然需要同步段.

(3) 扩展并行归约. 这种方法将部分和变量扩展为具有与处理器相同数量的元素的一维数组,并赋予全程域. 每一处理器使用处理器号为索引使用数组的一个元素. 并行循环结束后,部分和被加入初始变量. 这种变换产生了可完全并行的循环.

如何挑选合适的并行化方法以及如何将这些方法组合使用,需要综合考虑循环迭代的数量和处理机环境.

2.3 广义归纳变量替换

循环中的数组下标经常使用形如 $V=V\ \mathrm{op}\ K$(其中 K 的值是循环不变式)归纳变量的值,这样的递归赋值引起跨迭代数据依赖. 如果能将归纳变量的值表达为循环索引的函数,即可删除由此引起的数据依赖. 对 op 操作为加且 V 的值构成算术级数的归纳变量的识别和替换是成熟技术.

除此之外,还存在两类广义归纳变量:一类用乘操作取代加操作,其值形成几何级数;另一类虽然是加操作,但由于循环是三角的,其值不能在循环的所有点上构成算术级数.

对广义归纳变量的识别和替换并不困难. 它所引起的问题主要是替换后常常会引入非线性表达式,需要使用相应的非线性依赖分析方法,如前面提到的 Range Test[7],才能分析依赖关系.

3 数据分布和重分布技术

对基于分布内存的自动并行化系统而言,数据分布的选择对程序性能的影响多高都不过分. 数据分布的选

择依赖于目标机器、问题大小以及可用的处理器数量,一般情况下,这是 NP 难题. Kremer 等人指出,利用最新的 0-1 整数规划技术可以有效地求这些 NP-难题的优化解[16]. Guo 等人从无通信的分布入手,研究了线性数据分布的问题[17]. 为了回避分布内存系统的数据分布的复杂性,Cox 等人研究了将软件分布共享内存(distributed shared memory,简称 DSM)作为分布内存系统并行编译的目标的可能性[18].

3.1 数据分布

数据分布的一个研究热点是自动数据分布. 自动数据分布的目标是在程序分析的基础上,全盘考虑数据分布与计算分割的需要,推导出一个全局较优的数据分布模式,从而在尽可能大地开发程序中的并行性的同时,减少程序中通信、同步的开销.

关于自动数据分布问题已有很多的研究,但由于确定数据分布策略是 NP-难题,已经提出的算法大多是启发性算法. Kremer 等人提出的算法[16]没有过早地求助于启发信息,而是利用 0-1 整数规划技术来求这些 NP-难题的优化解.

Kremer 等人提出的自动数据分布框架由 4 步构成. 第 1 步,输入的串行程序被划分为程序段. 第 2 步,为每一程序段构建有希望的候选分布方案搜索空间. 每一程序段的候选分布方案是程序段中被引用的数组到目标体系结构的映射. 第 3 步,根据估计执行时间对每一候选方案进行评价. 最后一步,根据估计的候选方案代价,从每一候选空间挑选一个使全局代价最小的分布方案.

构建候选分布方案空间以及最终从每一候选搜索空间确定分布都是 NP 问题. 框架的引人之处在于,没有过早地求助于启发信息,而是利用 0-1 整数规划技术来求这些 NP-难题的优化解. 实验结果表明这是有效的.

数据分布研究的另一个热点是新的数组分布模式. 传统的数据分布都是数组沿着维内以 BLOCK 或 CYCLIC 分布,但对一些应用程序,这样的分布模式不能保证通信开销最小化. 例如下面的循环:

```
do i=1,n1
   do j=1,n2
      A(i,j)=F(A(i+1,j-1),B(i+1,j+1))
   enddo
enddo
```

不管数组 A 沿着行还是列分布都不可能避免处理器间通信,但如果数组 A 沿着反对角线(anti-diagonal)分布,则不会引起处理器间通信. Guo 等人研究了线性数据分布[17],即沿着平行超平面划分数组并映射到处理器,传统的 BLOCK 和 CYCLIC 以及对角线分布都是它的特例. Guo 等人的研究给出了无通信的数组分布和对齐的基于索引分析的方法以及数据分布的策略. 由于线性分布不存在简单的对全程和局部地址进行互相转换的代数公式,他们还提出了全程和局部地址的相互转换的算法以及转换全程迭代空间到局部迭代空间的转换算法.

3.2 数据重分布

重分布的一个研究热点是减少重分布开销. 重分布开销由索引计算和处理器间通信两部分组成. Guo 等人提出了一种有效的索引计算方法来降低索引计算时间,并提出一种无竞争的通信调度方法以减少通信时间[17].

Guo 引入一个称为局部数据描述符(local data descriptor,简称 LDD)的 4 元组来表达划分到一个局部内存的数组元素集合,在两个处理器间重分布的数据可由两个处理器 LDDs 的交集表示. 由此提出的算法可以处理任意的源和目标的分布方案为 block-cyclic 的数组重分布.

如果在重分布算法中不使用通信调度,则可能发生通信竞争,增加通信等待时间. 为解决这个问题,Guo 等人为重分布算法中广泛使用的 all-to-many 通信开发了一个通信调度算法. 算法基于 LDDs,首先产生表示发送结点和接受结点之间的通信关系的通信表. 根据通信表,产生通信调度表. 通信调度表的每一列是每一通信步接收结点编号的排列. 这样的通信是无竞争的.

3.3 基于软件 DSM 的并行编译

目前对数据分布和重分布所做的研究工作,基本上都是针对规则的程序. 由于不规则程序编译时不能获得足够的数据存取信息,因此产生的消息传递代码或者是低效的,或者相当复杂. 编译到软件 DSM 系统则避免了

这个问题,因为软件 DSM 系统提供了请调(on-demand)数据通信和自动数据缓存(caching)的功能.

Cox 等人估价了以软件 DSM 系统作为消息传递机器上并行化编译器的目标的可能性[18]. 他们的实验结果表明,软件 DSM 系统有希望作为不规则的应用程序的并行化目标. 如果适当地扩展编译器和 DSM 系统,对规则的应用程序,可以做到编译器产生的 DSM 程序性能与编译器产生的消息传递程序相当,这使得软件 DSM 有希望作为所有应用程序的一个通用的并行化平台.

4 调 度

并行编译中的调度研究的是如何分配给处理器一个作业的任务,以使得作业的执行时间最短. 这个工作一般由编译系统和运行环境负责. 在编译时刻进行的调度工作称为静态调度,在运行时刻进行的调度工作称为动态调度. 静态调度已有很多研究成果,但由于在很多情况下程序任务图的计算粒度和通信代价只有在程序运行时才知道,如循环结点的迭代次数,因此,根据运行时的粒度大小和通信代价进行动态调度很受重视.

Voss 等人研究了动态决定单个循环的串并行问题[19]. Nguyen 等人研究了整个程序的执行过程中自动调节处理器分配数量以最大化加速比的问题[20]. 由于工作站网络的可用性、廉价性和不可独占性,Blumofe 等人研究了工作站网络的的动态调度[21].

4.1 动态分配处理器

Voss 等人观察到某些可并行循环由于粒度不够大而导致并行执行反而比串行执行慢的现象[19],他们认为这主要是由于并行开销的存在. 考虑到并行开销是程序、程序输入以及机器配置的函数,只有在运行时才能确定,又考虑到程序的可移植性,Voss 等人提出了采用观察-执行方法(inspector-executor scheme)来动态地识别不宜并行的循环并动态地取消其并行执行的方法.

这个方法的基本原理是:对每一循环,首先并行执行并获取其执行时间,此时间用于判定算法判定此循环的并行执行是否由并行开销支配,若是,则此循环应串行执行. 在程序状态发生较大变化时重新执行判定算法.

测试表明,Voss 的方法可以有效地提高加速比. 此方法考虑的是单个循环,粒度较小,并且只考虑串行执行或在所有处理器上并行执行这两种情况.

Nguyen 等人则研究了循环并行应用程序(iterative parallel applications)的执行过程中自动调节处理器分配数量以最大化加速比的问题[20]. 由于观察到很多应用程序并不具有加速比随处理器数量增加而单调增加的现象,他们研制了这样的一个系统:(a) 动态地测量作业(job)在分配了不同处理器数量时的效率;(b) 使用这些测量数据计算相应的加速比;(c) 自动调节处理器分配数量以最大化加速比. 它除了具有不需用户静态分配处理器的优点以外,还具有一些显而易见的优点:

(1) 由于应用程序的性能可能随着输入数据的变化而有明显的变化,事先确定最佳数量的处理器有时是不可能的.

(2) 由于作业的加速比可能随着时间的变化而有明显的变化,可能不存在对一个作业的执行生命周期来说都是最优的静态处理器分配方案.

(3) 自调节的开销很小.

Voss 和 Nguyen 等人的工作的一个共同之处是基于共享处理器,Nguyen 等人的工作还假定处理器数量足够多.

4.2 工作站网络的动态调度

由于典型的工作站大量时间是空闲的,且单机价格比超级计算机的一个结点便宜得多,工作站网络为大规模的应用程序提供了强大的廉价计算资源. 由于工作站加入和退出计算的随意性,工作站网络的调度适于采用动态调度.

工作站网络的调度面临的主要挑战是其通信启动开销远大于超级计算机,而通信带宽远小于超级计算机. 因此,其调度目标是针对一些应用程序,调度器能将处理器间的通信降低到使得低下的通信性能并不导致整个应用程序性能的降低.

Blumofe 等人提出的 Phish[21]是一个满足此目标的调度器.为在工作站网络上有效地调度大尺度并行计算的任务,Phish 采用了"空闲启动(idle-initiated)"技术,即空闲计算机主动寻找作业而不是等待分配作业的技术.Phish 由宏层(macro-level)和微层(micro-level)两层调度构成.宏层负责将处理器分派给并行作业,它采用如下策略:(1) 分用处理器;(2) 处理器随作业并行性的增减而动态地加入或退出;(3) 允许工作站所有者对机器保留特权.微层负责将构成特定并行作业的任务分派给参与此作业处理的处理器,它采用如下策略:(1) 维护通信和内存局部性;(2) 适应动态并行性.原型系统的实验表明,Phish 实现了预设的目标.

5 需进一步研究的问题

研究和实践都表明,在目前设计出基于共享内存的可实用的自动并行化系统是可能的,但基于分布内存的可实用的自动并行化系统还需假以时日.为了设计更加实用和强大的自动并行化系统,还需要人们在从依赖关系分析、程序转换、数据分布优化到调度的各个环节进行研究,推出更好的新技术[2,16,17,22,23].

为了开发更加实用和强大的自动并行化系统,下面这些问题是特别需要加以研究的:

(1) 开发通用的编译器研究平台.编译技术如果脱离了编译器是没有意义的.为了使新的编译技术能容易地集成于编译器上,以便测试和评价其性能,从而提高编译器技术的研究质量及加快研究的进展,需要可靠的模块、独立的编译器平台.

(2) 收集大量的与实际应用有关的测试程序(benchmark).测试编译技术离不开测试程序.能够很好地刻画实际程序性质的测试程序对推进编译器技术的发展具有不可取代的作用.

(3) 开发高效的编译算法.为了将编译器的运行时间限制在可接受的范围内,包括依赖关系分析问题、多种循环重构技术的选择使用和组合问题、数据分布的选择、通信的优化、任务粒度的动态控制等问题,都需要更高效的算法处理,可以考虑采用并行算法.

(4) 开发自适应代码的产生算法.由于运行系统的变化性和程序对输入的依赖,为了提高加速比,需要开发产生自动适应系统变化和程序输入的代码的算法,包括运行时刻的依赖关系分析问题、运行时的数据分布和动态调度的算法.

参考文献

1 Jin Guo-hua, Chen Fu-jie. The Theory and Technology of Program Parallelizing in Massively Parallel Processing Systems. Beijing: Science Press. 1995
(金国华,陈福接.大规模并行机程序并行化理论与技术.北京:科学出版社.1995)

2 Feng Xiao-bing. Global optimizing technology of data distribution [Ph.D. Thesis]. Beijing: Institute of Computing Technology, The Chinese Academy of Sciences, 1999
(冯晓兵.数据分布全局优化技术[博士学位论文].北京:中国科学院计算技术研究所,1999)

3 Du J, Chen D, Xie Li. JAPS: an automatic parallelizing system based on Java. Science in China (Series E), 1999,42(4): 396～406

4 Banerjee U. Dependence Analysis for Supercomputing. Shen Zhi-yu, Zhao Ke-jia Trans. Changsha: Press of Science and Technology of Hu'nan, 1991
(Banerjee U. 超级计算中的依赖关系分析.沈志宇,赵克佳译.长沙:湖南科学技术出版社,1991)

5 Wolfe M. High Performance Compilers for Parallel Computing. Redwood City: Addison-Wesley Publishing Company. 1996

6 Pugh W. A practical algorithm for exact array dependence analysis. Communications of the ACM, 1992,35(8):102～114

7 Blume W, Eigenmann R. Nonlinear and symbolic data dependence testing. IEEE Transactions on Parallel and Distributed Systems, 1998,9(12):1180～1194

8 Rauchwerger L. Run-Time parallelization: a framework for parallel computation [Ph.D. Thesis]. University of Illinois at Urbana-Champaign, 1995

9 Cooper K D, Hall M W, Kennedy K. A methodology for procedure cloning. Computer Languages, 1993,19(2):105～117

10 Hall M W, Murphy B R, Amarasinghe S P *et al*. Interprocedural analysis for parallelization. In: Huang C H, Sadayappan

P, Banerjee U *et al* eds. Proceedings of the 8th Workshop on Languages and Compilers for Parallel Computing. Berlin: Springer, 1995. 61～80

11 Gu J, Li Z, Lee G. Experience with efficient array data flow analysis for array privatization. In: Proceedings of the 6th ACM SIGPLAY Symposium on Principles & Practice of Parallel Programming. New York: ACM, 1997. 157～167

12 Wilson R P, Lam M S. Efficient context-sensitive pointer analysis for C programs. ACM SIGPLAN Notices, 1995,30(6):1～12

13 Emami M, Ghiya R, Hendren L J. Context-sensitive interprocedural points-to analysis in the presence of function pointers. In: Proceedings of the ACM SIGPLAY'94 Conference on Programming Language Design and Implementation. New York: ACM, 1994. 242～256

14 Huang Bo, Zang Bin-yu, Yu Yi-jun *et al*.. Intraprocedural alias analysis for pointer array. Journal of Software, 1999,10(6):600～607
(黄波,臧斌宇,俞一峻等.指针数组的过程内别名分析.软件学报,1999,10(6):600～607)

15 Eigenmann R, Hoeflinger J, Padua D. On the automatic parallelization of the perfect benchmarks. IEEE Transactions on Parallel and Distributed Systems, 1998,9(1):5～23

16 Kremer U. Automatic data layout for distributed memory machines [Ph. D. Thesis]. Rice University, 1995

17 Guo M. Efficient techniques for data distribution and redistribution in parallelizing compilers [Ph. D. Thesis]. University of Tsukuba, 1998

18 Cox A L, Dwarkadas S, Lu H *et al*. Evaluating the performance of software distributed shared memory as a target for parallelizing compilers. In: Proceedings of the 11th International Parallel Processing Symposium. Los Alamitos, CA: IEEE Computer Society, 1997. 474～482

19 Voss M, Eigenmann R. Reducing parallel overheads through dynamic serialization. In: Processings of the 13th International Parallel Processing Symposium & the 10th Symposium on Parallel and Distributed Processing. Los Alamitos, CA: IEEE Computer Society, 1999. 88～92

20 Nguyen T D, Vaswani R, Zahorjan J. Maximizing speedup through self-tuning of processor allocation. In: Proceedings of the 10th International Parallel Processing Symposium. Los Alamitos, CA: IEEE Computer Society, 1996. 463～468

21 Blumofe R D, Park D S. Scheduling large-scale parallel computations on networks of workstations. In: Proceedings of the 3rd IEEE International Symposium on High Performance Distributed Computing. Los Alamitos, CA: IEEE Computer Society, 1994. 96～105

22 Padua D. Outline of a roadmap for compiler technology. IEEE Computational Science & Engineering, 1996,3(3):65～66

23 Chen Hua-ping. Heuristic task scheduling in parallel and distributed computing [Ph. D. Thesis]. Hefei: University of Science & Technology of China, 1997
(陈华平.并行分布计算中的启发式任务调度[博士学位论文].合肥:中国科学技术大学,1997)

New Development of Automatic Parallel Compilation

YANG Xue-lin YU Meng CHEN Dao-xu XIE Li

(*State Key Laboratory for Novel Software Technology Nanjing University Nanjing* 210093)
(*Department of Computer Science and Technology Nanjing University Nanjing* 210093)

Abstract Automatic parallel compilation provides important supports for parallelizing "legacy" programs and designing new parallel programs. It has been focused on for two decades. In recent years, the developments of automatic parallel compilation technique, including dependence analysis, program transformation, data distribution and redistribution, and scheduling, have gradually made it practically applicable. In this paper, the new development of automatic parallel compilation technique and the challenges of future research are presented.

Key words Automatic parallel compilation, dependence analysis, program transformation, data distribution, schedule.