并行编译中的区域调度问题
On Regional Scheduling of Parallel Compilation

苏运霖
Su Yunlin
(广西大学梧州分校,广西梧州 543002)
(Guangxi University Wuzhou Branch, Wuzhou, Guangxi, 543002, China)

摘要:介绍指令级并行性(ILP)中和指令级计算(EPIC)中区域的直观概念和这些概念的形式化工作,并简单介绍区域的几个调度算法,为并行编译中的区域调度问题提供一个系统的、形式化的论述。

关键词:并行编译 区域调度 形式化
中图法分类号:TP338.6

Abstract: The concepts of regions in the Instruction Level Parallelism (ILP) and the Explicitly Parallel Instruction Computing (EPIC) are explained. Sevral algorithms of regional scheduling are introduced.

Key words: parallel compilation, regional scheduling, formalization

为适应当代的各种应用领域对于计算机速度和存储容量的要求,今天的计算机厂家除了研制速度更高和容量更大的、单个的超大型计算机之外,都把注意力转移到并行计算机的研制上[1]。并行计算机通过并行运作,使计算机的性能获得指数级增长。并行是提高计算机效率的一个极其有效的途径。然而,并行把原来顺序地执行的程序转化为并行地执行或部分并行执行的程序。除了明显地可以容易地转化为并行程序执行的向量计算、矩阵计算以及联立方程组求解等问题之外,其它许多问题的并行化求解,都涉及到要针对计算机系统结构本身是指令级并行性(Instruction Level Parallelism, ILP)的处理器,还是明显地并行的指令级计算(Explicitly Parallel Instruction Computing, EPIC)的处理器,来确定实现并行的方式。然而,这两种方式的基本思想即区域调度是一致的。所不同的是,在两种处理器形式下,区域的"颗粒"大小或形式可能有不同。目前更大的问题还在于,对于区域,还没有一个普遍认同的形式化的定义。因此,也就影响了关于区域的进一步的理论研究。本文介绍ILP中和EPIC中区域的直观概念和对这些概念的形式化工作,介绍关于区域的几个调度算法,旨在把现有关于区域及区域调度的工作提升到形式化的高度,从而对整个并行编译的研究起到一定的促进作用。

2004-06-14 收稿。

1 ILP和EPIC中区域的直观概念

1.1 ILP中区域的概念及有关问题

定义1 在ILP中,区域指的是在同一时间里可以进行编译的一个程序的子部分。

根据定义,由于这些程序的子部分可以在同一时间里进行编译,因此,它们在被编译完成之后,也就可以接着在同一时间里投入运行。因此,这也就实现了并行的运行。

以下是在ILP中几种常见的区域类型。

(1)基本模块。基本模块是最普通的区域形式,是区域的"退化"形式。即它是有一个入口端和一个出口端的指令序列组成的程序节。

(2)踪迹。

定义2 一个踪迹是指通过程序代码的一个线性通路。

踪迹是区域的另一种常见类型。踪迹组成基本模块。

设 B_0、B_1、…、B_n 是程序的基本模块,而且这个顺序是既定的。一个踪迹是由这些基本模块中的操作组成的,它并有以下性质:

1) 每个基本模块是在这个序列中的下一个的前驱,即,对于每个 $k=0$、…、$n-1$,B_k 后续 B_{k+1} 或分支到 B_{k+1}。

2) 踪迹的代码无循环,除非整个区域是某个外延循环的一部分,即对于任何 i 和 j,不存在 $B_j \rightarrow B_i$

→ B_j 的路经。

踪迹并不禁止在区域内的、向前的分支，或者离开区域以及在稍后某时刻重新回到区域的流动。如图 1 就是一个多入口多出口的典型区域及踪迹的例子。

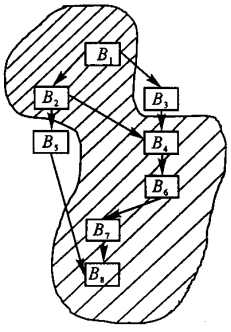

图 1　一个典型区域及踪迹例子

图 1 中 B_1、B_2、B_3、B_4、B_5、B_6、B_7、B_8 为基本模块。其中 $B_1 \rightarrow B_2 \rightarrow B_4 \rightarrow B_6 \rightarrow B_7 \rightarrow B_8$ 为通过区域（阴影部分）的一条踪迹。同理，$B_1 \rightarrow B_2 \rightarrow B_5 \rightarrow B_8$ 和 $B_1 \rightarrow B_3 \rightarrow B_4 \rightarrow B_6 \rightarrow B_7 \rightarrow B_8$ 也是踪迹。

(3)超级模块。超级模块其实也是踪迹。但加上一个限制条件，即除进入头一个模块的分支外，转入其它模块的分支都是不允许的。因此，一个超级模块是由基本模块序列 B_0、B_1、…、B_n 的操作组成，而且它具有和踪迹相同的性质，即：

1) 每一个基本模块是序列中下一个模块的前驱，即对于每个 $k = 0$、1、…、$n-1$，B_k 后续 B_{k+1}，或分支到 B_{k+1}。

2) 超级模块的代码无循环，除非整个区域是某个外延的循环的一部分，即对于任何 i 和 j，不存在 $B_i \rightarrow B_j \rightarrow B_i$ 的路径。

另外附加的性质为：

3) 在这个区域中，除对于 B_0 外，不存在转入区域中一个模块的分支。

在有关超级模块区域的参考文献[1]中，把那些非法的分支称为边门入口。

(4)树形区域。一个树形区域是在程序的控制流程内，含有基本模块的一个树形的区域。即一个树形区域是由基本模块序列 B_0、B_1、…、B_n 的操作组成，并有下列性质：

1) 除对于 B_0 外，每个基本模块恰有一个前驱，即基本模块 B_j 的前驱是序列中的基本模块 B_i，且 $i < j$，这意味着，通过树形区域的任何路径将产生一个超级模块，即一个没有边门入口的踪迹。

2) 对于任何 j 和 i，不存在 $B_j \rightarrow B_i \rightarrow B_j$ 的路径。这仅对于通过 B_0 的路径除外，即除非整个区域是某个围绕它的一个循环的一部分，否则这个代码是无循环的。

和超级模块一样，通过使用尾部复制和其它扩展技术，可以消除边门入口的限制。在一个区域内只有一个控制流动的这种情况称为线性区域。在这个意义下，踪迹和超级模块是线性区域，而树形区域是非线性区域。

(5)其它区域。除了上述区域外，专家们还提出了若干其它的区域。比如，踪迹 2 是具有一个入口的非线性区域，它有点像树形区域，但又没有对边门入口的限制，其实现却非常困难；大型模块也是一种区域类型，它是一个单入口、多个出口的区域，且有内部的控制流，它是超级模块的一种变形，它已含一些预测操作，而且可以像一个模块一样进行调度。

1.2　EPIC 中区域的概念及有关问题

实际上对于明显地并行的指令计算，其思路是相似的，理想地说，如果能知道动态踪迹，即由程序所执行的基本模块的特定顺序，就可以把它当作一个巨大的基本模块，并且极小化动态调整的长度。所以，对于明显地并行的指令级计算，区域是程序控制流程图的子集，它由一个或多个基本模块组成。包含在统计上可能是动态踪迹的极大长度段，所有高频率的模块间转移，都可归入到一个区域中。因此，每个这样的区域，也都可调度为一个单位，就像一个基本模块一样。在一个区域内，在基本模块间是没有调度障碍的。与在指令级并行性一样，踪迹允许多个入口和多个出口，超级模块是单个入口和多个出口。两者都是线性区域，在存在无偏向分支的情况下，区域必然包括离开分支的 2 个路径，因此产生非线性的区域。

一旦决定了区域的形状之后，如何把一个程序划分成为具有所述各种形状的区域，这一问题称为区域的形成问题。与这个问题相关联的还有调度的构造问题，也即区域调度的策略问题。

前一个问题，实际上是和第二个问题有关的，或者说，它是取决于第二个问题的。区域的形成必须把程序分划成为调度构造程序能够支配的，明确地定义的一些部分。这些部分同另一个基本概念描述性权值相关联。这个权值指出该部分程序的相对执行频率。在赋予执行频率即权值时，既可使用带启发性探索的方法，也可使用基于描述性的方法。对于程序中的一些重要部分，都要赋予执行频率，这包括在上下文无关文法(CFG)图中的节点或边。于是一个节

点或边的描述将指出一个具体的基本模块在它所在的程序中将被执行多少次，或者从一个基本模块到它的直接的邻居的每一个，控制流经多少次。

描述性数据的收集可以以一些不同的方法来进行。旧的技术称作制作[2]，它通过在程序文本中插入附加的代码来计算一个特殊事件的执行。但这种方法的缺点是要进行穷尽地测量，因而是不可取的。新方法是对于上下文无关文法中的边的子集进行抽样，从而仍然构造出所有边的权[2]。这种方法要抽样足够多的边来在上下文无关文法的图的无向版本中打破所有循环。这种方法的一个新版本是向前路径描述，它在一个过程中枚举所有的向前路径，然后通过在作出控制决定时，确定路径的个数。而后可以压缩向前路径调用的序列来描述整个程序的踪迹。

代替描述性，可以采用带启发性探索的描述综合。带启发性探索的描述方法基于源程序的结构对程序的每个部分赋予权值。它不要求运行程序。但这样一来，也就存在一些危险。那就是不知道实际数据的程序如何运行，就无从得知什么是普通代码，什么是例外代码。因此，也就很可能出现带启发性的探索在非普通性的代码上花费了宝贵的优化时间和存储空间。当然，也可能获得意外收获，即不必对真正的运行程序收集统计数字。

2 区域的形式化

区域的形成同程序的上下文无关文法有关，而且，只有从上下文无关文法的高度，才能把区域的概念形式化。

定义3 一个上下文无关文法 CFG G 由一个四元组确定。即 CFG $G=(N,\sum,P,S)$，其中：N 是非终端或变量字母表；$\sum$ 是终端字母表，且 $N\cap\sum=\varnothing$；$P\subseteq N\times(N\cup\sum)^*$ 是产生式规则的有限集合，P 中产生式均有 $A\to\alpha$ 的形式，A 在 N 中，α 在 $(N\cup\sum)^*$ 中；S 在 N 中，每个文法 G 均以 S 为左部的产生式开始，称为起始符号或语句符号。

定义4 一个上下文无关文法 G 所认记的语言 $L(G)$，是由该文法经过推导所得到的所有终端符号串的集合，这些终端符号串即为此文法所认记的语句。

所有程序设计语言的绝大部分成分的文法，都属于上下文无关文法，因此，由这些语言所编制的绝大部分程序的绝大多数组成就是上下文无关语言。

众所周知，一个程序说是某个语言的合法程序，如果从该语言的语句符号出发，通过使用该语言的文法产生式，最终可导出这个程序本身。这样一个过程，称为由顶向下的语法分析过程。它也就是编译的一个组成部分。另外一个方法，是从程序本身开始，运用归结的方法，或叫做由底向上的分析，最终把这个程序归结成为起始符号或语句符号。这样一个过程，可称为垂直编译。但是，程序的执行，则是对于上下文无关文法的推导图中叶节点一终端符号串自左向右的逐一实现，这样一个过程，可称为水平编译。

为并行地运行程序，就要把经过水平编译的程序，分成为区域而后通过区域调度来实现并行运行。然而，区域的形成既同水平编译有关，又同垂直编译有关。为了给出本文的结果，首先需要引进以下定义。

定义5 在上下文无关文法 CFG G 中，2个非终端符号 A 和 B 说是 $A\geqslant B$，如果 B 可以作为以 A 为左部的产生式的右部出现。

显然有，$\geqslant$ 关系是可传递的。因为如果 $A\geqslant B$、$B\geqslant C$，则利用产生式的推导关系。就会有 C 出现在以 A 为左部的产生式的右部中。因此，对于 G 中任何非终端符号 A，均有 $S\geqslant A$。若不然 A 为 S 不可到达的符号，它在文法中就没有意义。但是一般而言，N 是一个偏序集，而非全序集。

引理1 每一个上下文无关文法都等价于 Greibach 范式中的文法，即所有产生式具有如下形式 $A\to b\alpha$，其中 α 是一个(可能为空)非终结符号串。

定理1[3] 在 Greibach 范式下的文法中，如果在某个产生式 $A\to b\alpha$，中。非终端符号串 α 中包含 A 和 B 非终端符，而它们不存在 $\geqslant$ 关系，则由它们开始的推导组成不同区域。

证明 设从起始非终端符号 S 开始。将要用到这一产生式的产生式为 $S\to\alpha\beta$，而后由 $\cup$ 开始，继续使用或不使用其它产生式，而得到

$$S\to\beta\to\cdots\to\cdots b\alpha\to\cdots b\cdots A\cdots B$$

于是，我们得到图2的推导树形式。图2中往下，在对 A 和 B 进行进一步的推导时，由于它们之间不存在 $\geqslant$ 关系。因此，就不会出现冲突，即从 A 和 B 往下的推导形成分别的区域。

定理2 在一个上下文无关语言中，如果在某个产生式中，右端的非终结符号串中2次或2次以上出现同一个非终端符号，则由它们开始的推导组成不同区域。

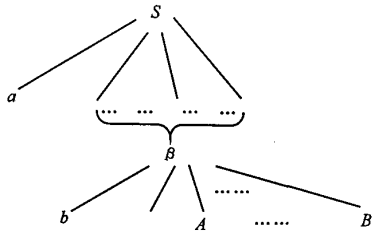

图 2 推导树

例 设 G 由

$S \rightarrow aBA|(CBA|aA|(CA|aB|(CB|a|(C$

$T \rightarrow aB|(CB|a|(C$

$A \rightarrow + s$

$B \rightarrow * T$

$C \rightarrow aBAD|(CBAD|aAD|(CAD|aBD|aD|(CD$

$D \rightarrow)$

给出，其中每个非终端符号的产生式以自左到右的顺序从 1 往上编号。显然，在此情况下，G 具有 Greibach 范式的形式。考虑图 3 中在 2 之下的((CBAD 和在 8 之下的(CBAD 便属于可以并行调度的区域。

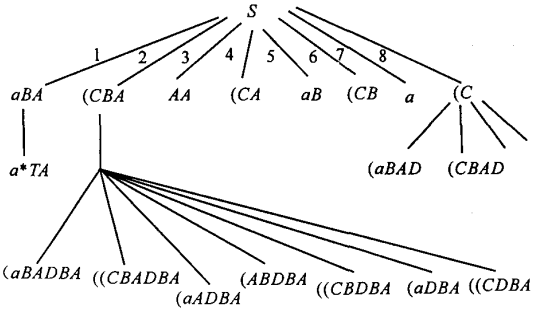

图 3 一个 Greibach 范式的推导树

定理 3 在条件语句中，2 个分支可以当成可以并行进行编译的区域，而在执行时，再依据哪个条件成立而来执行相应分支。

证明 当进行编译时，在 2 个分支中的程序都需要加以编译，以便在程序运行时，依据当时的条件判断，择一来运行，而在编译阶段，同时对 2 个分支（乃至 2 个以上分支）进行编译，不会有任何冲突发生。甚至到运行阶段，先让 2 个分支并行运行，而后才来判别条件，决定如何取舍分支。也同样可行。

例如，对于合并排序，我们有如图 4 所示的程序流程[4]。图 4 中，左右 2 个用虚线围成的部分是 2 个可并行编译的区域。编译完成后，在程序执行时可以再来执行。详见图 5。

定理 4 在 Shell 排序中，对于不同增量的比较，可作为不相交的区域而实现并行编译。

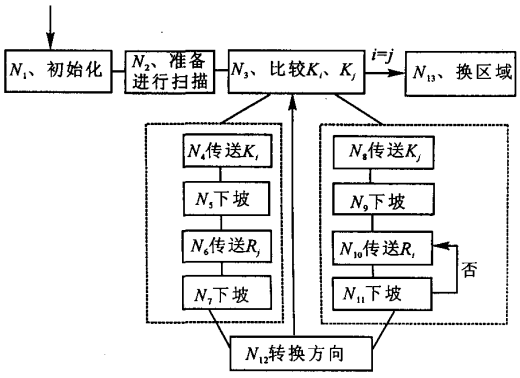

图 4 合并排序形成的区域

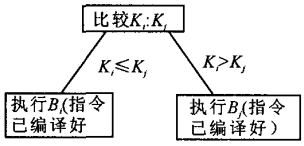

图 5 编译后的程序执行

证明 在带有增量 8,4,2,1 的 Shell 排序中，当参与排序的元素很多，开始时，是 1 与 9,17 与 25,33 与 41,…进行比较，这些比较，显然可以并行地进行。即既可以并行地编译，也可以并行地执行。同理，在增量为 4 时，1 与 5,9 与 13,…又可并行编译和执行，然后对于增量 2 和 1，同样可并行编译和执行。

定理 5 在对于表达式（既可以是数值表达式，也可以是命题表达式）进行求值时，当在表达式中分属于 2 个不相交的子表达的部分没有进行数值交换，则它们可当作并行编译的区域。

如在，$A_k = (k^{n-k}k! - (k-1)^{n-k+1}(k-1)!)/n!$ 中，$k^{n-k}k!$ 和 $(k-1)^{n-k+1}(k-1)!$，可并行地编译执行好，则有图 6 的对 B 和 C 两者的并行计算。

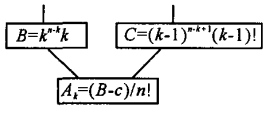

图 6 先并行计算后变成串行

3 区域的几个调度算法简介

区域调度的算法研究是形成区域之后自然地要解决的问题。形成区域的目的，就是要实现有效的并行编译和执行。

ILP 是当前并行微处理性的主流，而 EPIC 是其一种变形。因此，我们着重考虑 ILP 的区域调度算法。按照调度的体制，区域调度可分为循环调度和操作调度；按查找的风格，区域调度可分为贪婪调度和回溯调度；按流程分析，区域调度可分为线性调度和图形调度。

3.1 循环调度和操作调度

这一调度算法的目的是在极小化一个目标函数的同时，把操作分配到循环“槽”中，因此，整体的调度策略有循环调度和操作调度。循环调度是用重复地从区域选出的操作来填入一个循环，仅当穷尽了在当前的循环中可以利用所有操作之后才进行下一个循环。操作调度重复地选择在一个区域中的操作并把它放进相关性和目标资源允许的最早循环中。操作调度技术随选择方法而异，它可由一些带启发的探索或优先调度所引导。操作调度在理论上较循环调度更强有力，但其实现也要更为复杂。

3.2 线性调度和图形调度

对于由 n 个操作组成的一个区域，线性调度有 $O(n)$ 的复杂性，这是线性调度的主要优点。大多数线性调度技术使用尽可能快调度或尽可能迟调度。两者的区别在于把操作放在资源和数据限制允许的最早可能(最迟可能)的循环中。

图形调度技术，即表列调度。它重复地从准备好被调度的操作的一个数据就绪队列中选择一个操作。一个操作就绪，当它们所有数据就绪队列的前驱都已被调度时，操作就从数据就绪队列中被删除，而已变成就绪的它的后继被插入，这样迭代进行直到区域中所有操作都已被调度为止。

3.3 贪婪调度和同溯调度

在贪婪调度的情况下，从数据就绪队列贪婪地选择调度的候选者。回溯调度，则允许在一次调度失灵时，进行回溯，重新寻找合适的候选者。

4 结束语

并行编译程序技术，随着并行机的大量出现，正在日益成为计算机技术的热门话题。本文所涉及的区域及其调度问题，在并行编译技术中占有重要地位。本文旨在对区域给出形式化的处理。以推动整个并行编译技术的发展，但我们的工作也还是初步的，还有大量的形式化的工作可做。

参考文献：

1 Massimo Maresca. Scanning the special issue on microprocessor architecture and compiler technology. Proceedings of the IEEE, 2001, 89(11): 1547.

2 Farabosch P, Fisher J A, Young C. Instruction scheduling for instruction level parallel processor. Proceedings of the IEEE, 2001, 89(11): 1638～1660.

3 Robin Hunter 著. 通过 Pascal 的设计与构造. 见：苏运霖等译. 编译程序. 广州：暨南大学出版社，1992.

4 D E Knuth 著. 排序与查找. 见：苏运霖译. 计算机程序设计艺术. 第3卷. 北京：国防工业出版社，2002.

(责任编辑：邓大玉)

基于 WEB 的电子政务应用平台通过验收

南宁海蓝数据有限公司开发的基于 WEB 的电子政务应用平台，2004 年 8 月 16 日在南宁市通过了自治区信息产业局主持的验收。该项目是 2003 年广西电子信息系统应用计划项目。

系统由前台、后台、服务三大部分组成。其中前台部分包括新闻、项目库的浏览和查询、项目建议书原文的浏览、城市万事通的浏览和查询、历史人文中的原文原貌浏览等模块，后台部分包括新闻维护、用户管理、系统维护等模块，服务部分包括 WEB 服务、FTP 服务、DNS 服务、代理服务、防火墙、数据库等模块。系统以 Linux 为网络操作平台，以 WEB 技术、网络数据库和数字图书馆技术为核心，集信息发布、网络办公、自动更新、数据库应用及维护、纸质文档快速数字化及原样存储与显示等功能为一体，具有较好的适应性和通用性。

系统在政府网中应用自行开发的数字图书馆技术，实现原文原貌显示文献，很有特色。它实现了非专业人员对网页内容的简便维护和更新；应用了 perl 脚本编程技术，实现了基于 WEB 的图形化服务器维护及管理；对系统中存储的影像文件和数据库进行加密并分级管理，用技术手段实现内外网隔离和内容保密，加强了系统的安全性。

系统 2003 年 11 月在凭祥市政府投入使用，运行稳定，操作、维护简便，数据发布及时可靠，提高了工作效率和工作质量。基于 WEB 的电子政务应用平台达到国内同类技术应用的先进水平。

(广西科学院　罗海鹏)