逻辑程序的并行性粒度的分析*

黄志毅 胡守仁

(长沙工学院计算机系,长沙 410073)

摘要 本文提出了开发逻辑程序AND并行性中的粒度问题,给出了在我们的计算模型下粒度的计算公式,同时还给出了用粒度信息优化执行图表达式的具体例子.

关键词 并行性粒度,逻辑程序,并行处理,AND并行性.

并行性粒度分析是并行处理领域内一个重要的研究问题.并行性粒度即是指并行执行的某个任务的大小(即工作量).粒度太小,则会引起任务分配、调度开销增大,反而抵消了开发并行性带来的好处.粒度太大,则不利于加速程序的执行.因此,开发适度大小的并行性是提高系统和程序执行效率的关键.

我们在文献[1]提到开发并行性时,总是强调最大限度地开发并行性,这对于一种单纯的并行性开发技术而言是无可非议的.然而,在开发并行性时,我们所面临的困境是,虽然有些并行性可以开发,但对其开发所需的任务分配和调度开销远远抵销了开发并行性带来的好处.因此,并不是所有的并行性都值得开发.我们称并行性开发技术识别的并行性为Can(能够)并行性,而称那些开发后能够带来效益的并行性为Should(应该)并行性.并行性粒度分析这一任务就是要从Can并行性中区分出Should并行性,使系统只开发那些Should并行性.

并行性粒度分析目前是并行推理机编译技术中尚待深入研究的问题,国际上在这方面所做的工作较少[2,3].Tick所提出的粒度分析方案主要是针对并行语言FGHC的.该方案对每个过程给出一个权值.在执行程序时,系统根据权值对各并行执行的过程进行调度.严格地说来,该方案分析出来的过程的粒度信息是不精确的.Debray提出了一个非常精确的计算各过程粒度的方法:该方法首先分析每个子句中各变元的相对大小,然后根据这些相对大小建立各过程的粒度公式.最后通过动态测试的方法确定各个目标的实际粒度.该方法所需编译开销较大,而且需要增加一些动态开销.

在我们的模型中,由于开发AND、OR两种并行性,AND并行性开发得较充分,且抽象机所含的处理机数目不大,因此,在粒度分析时,我们无须过分考虑并行性的损失.基于这个原则,我们所提出的粒度方法应该估计出各过程的粒度下界(最好是最大下界),这样可保证

* 本文1992-01-03收到,1992-04-07定稿

作者黄志毅,29岁,1992年博士毕业于国防科技大学,主要研究领域为并行处理,体系结构,逻辑程序.胡守仁,68岁,教授,博士导师,主要研究领域为神经网络计算,新一代计算机体系结构.

本文通讯联系人:胡守仁,长沙 410073,长沙工学院计算机系

开发出的并行性一定是得大于失的.下面我们给出具体的粒度计算方法.

1　下界粒度分析法

对于粒度进行分析时,我们有多种度量单位作为基本粒度单位,如一致化次数,归结次数,指令条数.为了方便起见,我们采用归结次数作为粒度的度量单位.

1.1　粒度计算公式

对于非递归过程,我们有如下公式:

$$Cost_{非递归过程} = \sum_{i=1}^{n} Cost_{子句i}/n$$

n 为过程中子句个数.

每个过程的粒度大小为其子句粒度的平均值.其中子句的粒度按如下方式计算:

$$Cost_{子句} = \sum_{i=1}^{m} Cost_{子目标i} + 1$$

其中 m 为子句体中子目标个数.

每个子目标粒度的计算方式与相应的过程相同.

对于递归过程,有如下计算公式:

$$Cost_{递归过程} = (\sum_{i=1}^{l} Cost_{递归过程i}/l) \times 递归深度 + 1$$

上述公式中的递归深度指该过程递归调用的次数.而递归子句的粒度计算公式如下:

$$Cost_{递归子句} = \sum Cost_{子目标j} + 1.\quad 子目标\ j \neq 子句头$$

从上述公式我们可以看出,粒度计算方法在全面开发 OR 并行的模型下是比较准确的.由于它们只求出各分枝粒度的平均值,因此,当只开发部分 OR 并行时,这些计算公式只是求出了粒度的下界.因此,在 RAP/LOP－WAM 模型[4]中,该计算方法所得到的粒度大小也只是下界.

对于递归过程计算公式中的递归深度,我们可通过动态运行时的一些统计信息调整该值,使其能反映实际程序的平均递归深度.当然,该递归深度也应当是偏向于取下界的.

上述粒度计算方法的最大优点是,计算方法简单,采用下界计算法适合于并行性开发较多的模型,从而在一定程度上限制小粒度并行性的开发,减少任务分配,通讯等开销.这种方法尤其适合处理机台数不多的情况.

1.2　优化执行图表达式

在文献[1,5]中,通过开发 AND 并行性,产生出的执行图表达式的语法规则如下:

执行图表达式:==SEQ(子表达式$_1$,…,子表达式$_n$)

　　|子表达式

子表达式:==PAR(子目标$_1$,…,子目标$_n$)

　　|IPAR[变量集合 1,…,变量集合 n](子目标$_1$,…,子目标$_n$)

　　|GPAR[变量集合](子目标,…,子目标 n)

　　|子目标

我们采用粒度分析的结果对上述表达式进行优化,即若 PAR,IPAR,GPAR 表达式中

含有粒度小于等于基本粒度的子目标时，将该表达式变成顺序执行的表达式 SEQ.

定义. 基本粒度

我们将派生并行任务带来的任务分配，通讯，同步控制开销折合成 B 次归结的开销，称 B 为基本粒度.

对于每个执行图表达式，我们需对其中的 PAR，IPAR，GPAR 子表达式进行检查. 若某子目标的粒度小于等于 B，则将相应的子表达式变成 SEQ 表达式，从而优化了执行图表达式.

2 举 例

下面我们以快速分类程序说明我们的粒度计算和优化的思想.

例 1：快速分类程序

```
quicksort([H|T],S):-
                    split(H,T,X,Y),
                    quicksort(X,X1),
                    quicksort(Y,Y1),
                    append(X1,[H|Y1],S).
                    quicksort([],[]).
split(H,[E|T],[E|X],Y):-
                        E<H,
                        split(H,T,X,Y).
split(H,[E|T],X,[E|Y]):-
                        E>=H,
                        split(H,T,X,Y).
split(H,[],[],[]).
append([],L,L).
append([H|T],L,[H|L1]):-
                         append(T,L,L1).
? -quicksort([5,3,2,4,1,3,6,7,9,10],L).
```

通过文献[1]中的 ICAAP 方法我们可以求得如下执行图表达式：

```
quicksort(GN,[H|T],S):-
        SEQ(split(GGNN,H,T,X,Y),
            PAR(quicksort(GN,X,X1),
                quicksort(GN,Y,Y1)),
            append(GGN,X1,[H|Y1],S)).
quicksort(GG,[H|T],S):-
        SEQ(split(GGNN,H,T,X,Y),
            PAR(quicksort(GN,X,X1),
                quicksort(GN,Y,Y1),
            append(GGG,X1,[H|Y1],S)).
split(GGNN,H,[E|T],[E|X],Y):-
            PAR((GG,E<H),
                split(GGNN,H,T,X,Y).
```

```
split(G???,H,[E|T],[E|X],Y):-
        PAR((GG,E<H),
            split(G???,H,T,X,T)).
split(GGNN,H,[E|T],[E|X],Y):-
        PAR((GG,E>=H),
            split(GGNN,H,T,X,Y)).
split(G???,H,[E|T],[E|X],Y):-
        PAR((GG,E>=H),
            split(G???,H,T,X,Y)).
append(GGN,[H|T],L,[H|L1]):-
            append(GGN,T,L,L1).
append(GGG,[H|T],L,[H|L1]):-
            append(GGG,T,L,L1).
append(???,[H|T],L,[H|L1]):-
            append(???,T,L,L1).
```

首先我们要利用前面的公式求出每个过程的粒度大小.计算按从底至顶的顺序进行.假设我们的递归深度为 10.

对于 append 过程

$$Cost_{append}=1\times 10+1=11$$

对于 split 过程,假设内部谓词<,≥的执行也相当于一次归结的开销,则有

$$Cost_{split}=((2+2)/2)\times 10+1=21$$

对于 quicksort 过程

$$Cost_{quicksort}=(21+11+1)\times 10+1=331$$

假设基本粒度 B 为 50,则可用上述粒度对前面的执行图表达式进行优化.优化后的执行图表达式如下:

```
quicksort(GN,[H|T],S):-
                SEQ(split(GGNN,H,T,X,Y),
                    PAR(quicksort(GN,X,X1),
                        quicksort(GN,Y,Y1)),
                    append(GGN,X1,[H|Y1],S)).
quicksort(GG,[H|T],S):-
                SEQ(split(GGNN,H,T,X,Y),
                    PAR(quicksort(GN,X,X1),
                        quicksort(GN,Y,Y1)),
                    append(GGG,X1,[H|Y1],S)).
split(GGNN,H,[E|T],[E|X],Y):-
                SEQ((GG,E<H),
                    split(GGNN,H,T,X,Y)).
split(GG??,H,[E|T],[E|X],Y):-
                SEQ((GG,E<H),
                    split(GG??,H,T,X,Y)).
split(GGNN,H,[E|T],X,[E|Y]):-
                SEQ((GG,E>=H),
```

```
                    split(GGNN,H,T,X,Y)).
split(GG??,H,[E|T],X,[E|Y]):-
                SEQ((GG,E>=H),
                    split(GG??,H,T,X,Y)).
append(GGN,[H|T],L,[H|L1]):-
                    append(GGN,T,L,L1).
append(GGG,[H|T],L,[H|L1]):-
                    append(GGG,T,L,L1).
append(???,[H|T],L,[H|L1]):-
                    append(???,T,L,L1).
```

3 小 结

本文给出了一种逻辑程序的粒度分析方法——下界粒度分析法.该方法适合并行处理机台数较少的情况,特别是RAP/LOP—WAM抽象机模型.作者还将对更精确的粒度分析方法进行研究,即对于递归过程的递归深度设定一个参数,过程的粒度大小通过该参数给出计算公式,动态运行时根据该参数值计算准确的粒度大小.

参考文献

1 Huang Zhiyi, Hu Shouren. A compiling approach for exploiting AND—parallelism in parallel logic programming systems. Proc. of Parallel Architecture and Language Europe, Netherlands, 1989:335—345.

2 Debray S K, Lin Naiwei. Task granularity analysis in logic programs. Technical Report, Dept. of Comp. Sci., Univ. of Arizona, 1990.

3 Tick E. Compile—time granularity analysis for parallel logic programming languages. Proc. of Inte. Conf. on 5th Generation Computer Systems, Japan, 1988:994—1000.

4 Gao Yaoqing, Hu Shouren, Sun Chengzheng. Design and implementation of RAP/LOP parallel abstract machine. Inter. Conf. on Info. Processing, Japan, 1990.

5 Huang Zhiyi, Hu Shouren, Sun Chengzheng *et al*. Reduction of code space in parallel logic programming systems. Proc. of Parallel Architecture and Language Europe, 1991.

GRANULARITY ANALYSIS FOR LOGIC PROGRAMS

Huang Zhiyi and Hu Shouren

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

Abstract This paper discusses the problem of granularity while exploiting AND—parallelism in logic programs. The authors give evaluation formulas of granularity under their computation model. An example is given to show how the execution graph expressions are optimized with the size of grain.

Key words Task granularity, logic program, parallel processing, AND—parallelism.