构造并行化系统交互环境的若干关键技术*

杨 博, 王鼎兴, 郑纬民

(清华大学 计算机科学与技术系,北京 100084)

E-mail: yangbo@est1.cs.tsinghua.edu.cn

http://hpc.cs.tsinghua.edu.cn

摘要: 交互式并行化系统通过提供友好的交互功能并引入用户知识来协助程序的并行化,是解决自动并行化能力不足的一条有效途径.描述了一个并行化系统交互环境TIPSIE(interactive environment of Tsinghua interactive parallelizing system),并就构造该环境的性能预测、增量编译和数据相关查询等关键技术进行了讨论.实验结果表明,这些技术能够有效地提高系统的并行化能力和效率.

关键词: 并行编译;交互系统;增量编译;数据相关查询;性能预测

中图法分类号: TP315 **文献标识码**: A

从20世纪80年代中期开始,随着并行计算机的发展,人们开始研究串行程序并行化系统.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Illinois大学的Polaris[1]、Stanford大学的SUIF(Stanford University intermediate format)[2]、复旦大学的AFT(automatic Fortran transformer)[3]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,并行化能力有了较大提高.

虽然自动并行化技术取得了很大的进展,但是其并行化能力同手工并行化相比还有很大差距.为了克服自动化并行编译器算法能力的不足,出现了一些半自动的并行化系统,如Greenwich大学的CAPTools[4]、Rice大学的Fortran D系统[5]以及Applied Parallel Research公司的Forge90,这些系统通过提供一些交互手段,使程序员可以向并行化系统输入自己的知识,以协助程序的并行化,在部分程度上弥补了全自动并行化系统的不足.

TIPS(Tsinghua interactive parallelizing system)是一个交互式的并行化系统.该系统采用了大量的并行化新技术,具有强大的自动并行化底层支持.另外,我们针对底层自动并行化系统的特点,设计并实现了一个交互式的并行化环境TIPSIE(interactive environment of TIPS),使用若干交互技术来协助程序的并行化,其中包括:

(1) 使用可视化技术,将程序信息以易于理解的方式提供给用户;

(2) 提出了一种动静结合的程序性能预测方法,可以快速而准确地获取程序性能,使用户将注意力集中在程序最重要的部分和计算最为密集的循环上;

(3) 使用增量编译技术使得在用户对源代码进行修改后,系统能够通过重编译测试,确定需要

* 收稿日期: 1999-11-25; 修改日期: 2000-02-23

基金项目: 国家自然科学基金资助项目(69933020);国家863高科技发展计划资助项目(863-306-ZD-02)

作者简介: 杨博(1976-),男,陕西商州人,博士生,主要研究领域为并行处理,网络计算技术;王鼎兴(1937-),男,江苏吴县人,教授,博士生导师,主要研究领域为并行/分布计算,网络计算技术;郑纬民(1945-),男,浙江鄞县人,教授,博士生导师,主要研究领域为并行处理.

重新编译的程序单元,从而缩短重新编译的时间,提高编译系统的效率;

(4) 提出了一种基于表达式范围的数据相关查询方法,将自动数据相关分析所得到的不确定性结果转化成一些简单的问题提交给用户,以改进相关性分析的结果,提高并行化系统的能力.

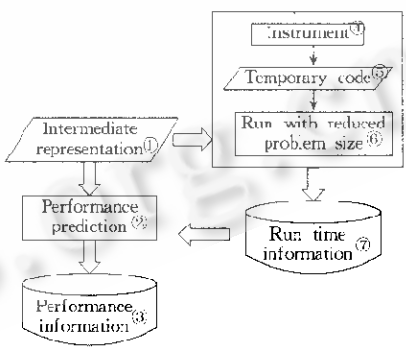
对于一个交互式系统来说,用户接口的友好性十分重要.可视化技术通过交互信息到图形方式的映射使得并行化信息能够更好地被用户所理解,同时使用户输入自己的知识更加方便.由于这部分工作比较琐碎,这里就不再详细给出,有关 TIPSIE 的结构和可视化技术请参阅文献[6].本文重点针对 TIPSIE 中的其他交互技术进行讨论.第 1 节主要给出 TIPSIE 中使用的动/静结合的程序性能预测方法.第 2 节和第 3 节分别就数据相关查询技术和过程级增量编译问题进行讨论.第 4 节是实验结果与分析.第 5 节是对全文的总结.

1 性能预测技术

在交互式并行化系统中,性能分析是一项非常重要的交互技术,准确、有效的程序性能分析与并行化系统的紧密结合能够提高系统的并行化能力.并行化系统中的性能分析工具通常使用两种方法:动态监控和静态预测.为了提高准确性和实现方便,大部分并行化系统都使用动态监控方法,通过实际运行程序来获取程序信息.静态预测方法不需要程序的实际执行,可以在编译时迅速地对程序性能作出预测,比较适合在交互式系统中使用,但是,由于在预测过程无法确定运行时的程序行为,因此准确性较差.

1.1 动/静结合的程序性能预测方法

可扩展的数据并行性程序是科学和工程计算程序的非常重要的一类.统计结果表明,90%的科学和工程计算程序属于这一类型[7].这些程序可以通过改变程序访问的数据元素的规模来进行扩展.根据这一特点,TIPSIE 除了提供上述两种性能分析方法之外,还提出一种动/静结合的程序性能预测方法 SPEASE(static performance estimation after single execution),使用该方法的性能预测器的简化结构如图 1 所示.在动态分析时,我们选用了不含分支、跳转和过程调用的程序段作为计时对象,通过自动插入计时语句生成动态分析时的临时代码.产生的临时代码经过修改后在目标机上运行,以此获取其在小规模下与问题规模无关的程序段的运行时间信息.时间预测计算在 IR 的基础上通过静态分析确定在原规模下程序段的运行次数以计算不同层次的循环、分支和过程的执行时间信息.由于动态分析在小规模下运行程序,因而开销较小.另外,这些运行时信息可以存储起来,供静态预测时多次使用,以获取程序在不同问题规模下的运行时间,从而分析程序性能随问题规模的变化关系.SPEASE 通过在静态预测中引入运行时信息,在提高性能预测速度的同时,在一定程度上保证了预测的准确性.



①中间表示IR,②时间预测计算,③程序性能信息,④插入计时语句,⑤临时代码,⑥小规模运行,⑦运行时信息.

Fig. 1 Simplified structure of SPEASE
图1 SPEASE的简化结构

1.2 循环界和分支概率的确定

在性能预测过程中,循环的界和分支选取的确定将直接影响到预测的准确性.但是,在许多情

况下,使用静态分析则很难得到,这主要是由于循环的上、下界和分支的条件表达式中变量的存在所造成的.

在经过对实际程序的分析后,我们发现,循环界不确定的情况主要出现在多重循环的内层循环,而且在相当一部分情况下,内层循环的界为外层循环变量的表达式.对于这种情况,可以使用符号表达式求和与化简技术来准确计算循环的运行时间.另外,还有其他一些循环,可以通过编译器分析和变量替代,转化为这种类型的循环,让内层循环的上、下界仅含有外层循环变量.同确定循环的界一样,分支选取的确定对于性能预测的准确性来说也是非常重要的.考虑到开销和准确性两个方面,我们使用了范围传播技术[8]来获取变量在程序中某点的范围,然后以此为依据来计算分支条件中的表达式范围,并通过假定该表达式的取值在其范围内均匀分布来计算条件为真的概率,从而得到近似的分支概率.限于篇幅,使用SPEASE方法的性能预测器的设计和具体实现就不再给出,有关细节请参阅文献[9].

2 数据相关查询技术

交互式数据相关性分析[4]通过与用户的交互引入用户所掌握的语义信息,从而改进相关性分析的结果.在TIPS系统中,我们借用了Range Test[11]的思想,即通过判定数组访问区域不发生重叠来确定不存在相关,提出了一种基于表达式范围的相关性提取算法RBDDIE(range based data dependence information extraction),从自动数据相关性分析得出的不确定性结论中归纳出一些简单明了的问题,根据用户对这些问题的回答情况最终确定是否存在数据相关.

```
      do i=1 to N
       do j=1 to Nj
S:       A[i*N+j]=...
T:     ...=... A[i*N+j]...
       enddo
      enddo
```

Fig. 2 Sample of programs that RBDDIE deals with

图2 RBDDIE针对的程序示例

首先考察如图2所示的程序,通过分析我们可以发现,当$N \geqslant N_j$时,i循环的迭代实例对数组A的访问区域之间不会发生重叠,因而不携带相关.我们可以对此向用户提问,如果能够得到肯定的回答,则循环i就能够并行化.RBDDIE就是针对这类情况来构造问题的.图2中的程序是一个一维数组的例子.对于多维数组,RBDDIE按一维的情况分别处理各维,在今后的论述中,我们将仅以一维数组为例.另外,在目前的版本中,RBDDIE仅对数组下标为循环变量的线性表达式的情况进行处理.在描述RBDDIE的算法之前,先给出一些定义.

定义1. 相关对的公共循环是指同时包含相关对的循环.相关对的局部循环是指只包含相关对中的一个的循环.检测并行性时只对相关对的公共循环进行.

定义2. 当前正在检测其并行性的公共循环称为待测循环.凡是位于待测循环之外的公共循环称为受限循环,所谓受限是指在相关对中该循环的循环变量要取相同的值.凡是位于待测循环之内的循环,包括公共循环和局部循环,都称为自由循环.

定义3. 由某重循环造成的相关对数组下标之间的最小差值,称为相关对关于该重循环的界.根据定义2中的不同循环类型,界可分为受限界、自由界和待测界.

定义4. 由某重循环造成的相关对数组下标之间的最大差值,称为相关对关于该重循环的覆盖.覆盖也可根据不同的循环类型分为受限覆盖、自由覆盖和待测覆盖.

定义5. 相关对数组下标中常数项的差值既可作为界又可作为覆盖.由于它没有相应的循环,为了算法实现的方便,假设它也对应于某重循环——第0重循环,而该差值作为第0重循环的受限

界和受限覆盖.

在构造问题之前,RBDDIE 首先计算所有的界和覆盖,如算法 1 所示.

算法 1. 计算界与覆盖的算法

```
输入:相关对的数组下标表达式 X 和 Y 关于循环变量的信息
输出:受限界、受限覆盖、自由界、自由覆盖、待测界和待测覆盖
    自由界=NULL;受限界=NULL;待测界=NULL;
    自由覆盖=NULL;受限覆盖=NULL;待测覆盖=NULL;
    if 是第 0 重循环){
      受限界=常数项的差值;受限覆盖=常数项的差值;
    }
    else if(是公共循环){
        if (X,Y 关于该循环变量的系数相同且不为 0){
        自由界=0;自由覆盖=系数 * 循环变量的(上界-下界);
        受限界=0;受限覆盖=0;
        待测界=系数 * 循环变量的步长;待测覆盖=自由覆盖;
    }
    else if (X,Y 关于该循环变量的系数有一个为 0){
        自由界=系数差 * 循环变量的下界;自由覆盖=系数差 * 循环变量的上界;
        受限界=系数差 * 循环变量的下界;受限覆盖=系数差 * 循环变量的上界;
        待测界=自由界;待测覆盖=自由覆盖;
    }
    else 受限界=系数差 * 循环变量的下界;受限覆盖=系数差 * 循环变量的上界;
    }
    else {/ * 局部循环 * /
        自由界=系数 * 循环变量的下界;自由覆盖=系数 * 循环变量的上界;
        待测界=自由界;待测覆盖=自由覆盖;
    }
```

如果某个循环的界(该循环可以是公共循环、局部循环,也可以是第 0 重循环.至于用哪个界,则由该循环的当前类型决定)大于其他所有循环的覆盖之和(用哪个覆盖同样由相应循环的当前类型决定),则被测循环不携带由所给相关对产生的相关.RBDDIE 就是据此来提出问题的.

3 增量编译技术

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.如果在用户对程序稍加修改后还要对整个程序完全进行重新编译,则会浪费用户的很多时间,降低并行化的效率.TIPS 系统在文献[11]中所述原理的基础上设计并实现了一种过程级的并行化增量编译技术,可以通过分析找出用户修改所影响到的过程,并只对这些过程进行重新编译,其他未受影响的过程可以直接使用上次编译的结果,从而减少了重新并行化所需的时间,增强了系统的交互性.重编译测试方案的思路是:对于每一个过程,入口的别名信息以及过程调用语句的副作用信息(MOD 和 USE)的增加将引起该过程的重编译,而减少则不会引起重编译;对于入口的常量信息,则是减少引起重编译.

重编译测试过程可以分为以下几个组成部分,如算法 2 所示.

算法 2. TIPS 中的重编译测试算法

(1) 通过分析确定被修改的过程;

(2) 构造测试用的过程间信息集合

(3) 对于未修改的过程,测试重编译条件,确定是否需要重新编译;

(4) 进行中间表示的重组,将需要进行重编译的过程和这些过程中新的 IR 提供给下面的模块,重新编译这些程序单元.

```
for 程序中的每一个过程 p do
    recompile_flag[p]=False;
od
for 程序中的每一个过程 p do
    比较新代码和旧代码;
    if 被修改 then recompile_flag[p]=True fi;
od
if 存在过程 p,recompile_flag[p]=True then
        计算重编译测试用数据
        for 每一个过程 p do
            if recompile_flag[p]=False then
                测试重编译条件;
                if 条件成立 then    recompile_flag[p]=True;  fi
            fi
        od
fi
for 程序中的每一个过程 p do
    if recompile_flag[p]=False then
        使用过程 p 的旧的中间表示替代新的中间表示
    fi
od
```

在重编译测试后,过程内的分析和变换仅对 recompile_flag=True 的过程进行.需要指出的是,在每一次编译时,全局的分析和优化都需要重新进行,如过程间私有化分析等.

4 实验结果与分析

我们对 TIPSIE 中性能预测方法 SPEASE 的准确性、数据相关查询技术 RBDDIE 和增量编译的有效性进行了测试.所有这些实验是在一台拥有双 CPU 的 Ultra 2上进行的,操作系统是 Solaris 2.5.1.

首先,我们给出 SPEASE 的测试结果.测试程序选自普渡大学的 purdue-set,结果见表1,其中"最小误差"和"最大误差"为不同问题规模下|(预测时间-实测时间)/实测时间|*100%的最大和最小值.从测试结果可以看出,许多程序的最小误差非常小,而最大误差绝大部分控制在8%以内,其准确性能够交互式并行化系统的需要.

Table 1 Accuracy testing result of SPEASE

表1 性能预测方法 SPEASE 的准确性测试结果

Program①	Max problem size②	Min problem size③	Max error④(%)	Min error⑤(%)
Problem01.f	1 024	524 288	0.38	0.60
Problem02.f	128*128	1024*1024	0.24	5.16
Problem03.f	128*128	1024*1024	0.38	6.53
Problem04.f	16 384	65 535	4.60	7.05
Problem05.f	64*64	256*256	2.53	8.25
Problem06.f	8 196	262 144	3.12	7.35
Problem07.f	2 048	32 768	4.69	4.98
Problem08.f	8 192	130 172	1.66	4.60

①程序,②最大问题规模,③最小问题规模,④最大误差,⑤最小误差.

下面,我们给出数据相关查询技术 RBDDIE 的提问能力.实验采用了 Perfect Benchmark 作为测试实例.该测试集共有13个程序,在 TIPS 能成功编译的11个程序中(SMSI.f 和 CMSI.f 除外),RBDDIE 对其中的7个提出了问题.其提问情况见表2.

Table 2 Experimental result of RBDDIE on PERFECT Benchmark

表2 RBDDIE 针对 PERFECT 基准程序集的测试结果

Program①	Total number of loops②	Number of parallel loops③	Number of Loops that have questions④	Number of loops whose questions can be answered⑤
APSI.f	298	223	11	1
LWSI.f	52	38	2	2
NASI.f	233	138	11	2
OCSI.f	289	250	3	0
SRSI.f	224	203	9	0
TFSI.f	176	171	3	0
WSSI.f	389	356	2	2

①程序,②总循环数,③并行循环数,④提问循环数,⑤用户可回答的循环数.

表2中的用户可回答循环是指经过我们分析,认为经过用户回答能够得以并行的循环.从表中可以看出,RBDDIE 有较强的提问能力,只是用户可以回答的循环数相对较小.这一方面是由于 TIPS 系统底层的自动数据相关性分析能力原本就比较强大,在一定程度上使得能够并行而又未被它们分析出来的循环数比较少;另一方面,上面关于循环是否可回答的判断是我们在简单地对程序进行浏览后作出的,我们对程序背景知识的了解当然无法与真正的编程者相比,这也可能造成一些原本可回答的循环被我们误认为是不能回答的;再有,当前所实现的交互式数据相关性分析对一些情况没有考虑,如数组下标中又出现数组变量的情况,这也是我们进一步工作的方向.

由于增量编译的性能与用户的行为有关,因此,我们在测试增量编译的性能时侧重于当程序中被改变了一个过程且仅仅需要重新编译这个过程时的结果,以说明增量编译的有效性.表3给出了 Perfect Benchmark 中6个程序的完全编译和增量编译所用的时间以及效率提高比例,其他的程序结果与此类似,表中不再给出(表中"效率提高"定义为$(T_{完全编译}-T_{增量编译})/T_{完全编译}*100\%$,"其他"是指重编译测试和过程内分析和优化之外的时间,包括常量传播、过程间分析与优化、后处理等).

Table 3 Efficiency improvement achieved by incremental compilation (s)

表3 采用增量编译技术获得的效率提高比例 (秒)

	APSI.f		LWSI.f		MTSI.f		SRSI.f		TISI.f		WSSI.f	
	Complete①	Incremental②	Complete	Incremental	Complete	Incremental	Complete	Incremental	Complete	Incremental	Complete	Incremental
Recompilation test③	—	9	—	1	—	3	—	4	—	1	—	6
Intraprocecural analysis and optimization④	217	18	68	3	45	2	82	2	176	2	1 847	13
Other passes⑤	199	191	80	80	83	83	96	90	81	83	221	228
Total time⑥	416	218	148	84	128	88	178	96	257	86	2 068	247
Efficiency improvement⑦	47.6%		43.2%		31.3%		46.1%		66.5%		88.1%	

①完全,②增量,③重编译测试,④过程内分析与优化,⑤其他,⑥总时间,⑦效率提高.

从表3可以看出,在用户有少量改动并且该改动对其他过程影响较小的情况下,采用增量编译技术能够减少不必要的重复分析,提高交互式系统的效率.通过实验结果我们可以发现,增量编译

本身进行重编译测试的时间很小,没有为编译器带来额外的负担,节省的主要时间为过程内的分析与优化时间,而其他部分没有发生明显的变化.在今后的工作中,我们将加强过程级增量编译与其他过程间的分析与优化的结合,进一步减少不必要的重复分析.

5 总 结

交互式并行编译系统通过为用户提供友好的接口引入用户的知识,是提高并行化系统能力的一项重要途径.在交互式并行化系统 TIPS 中,我们在自动并行化模块的基础上建立了一个交互环境 TIPSIE,并实现了信息可视化、性能分析、数据相关查询,这些交互式技术能够使用户对程序的并行化信息有更深入的了解,并能以友好的方式输入用户的知识,从而提高系统的并行化能力.另外,同其他一些交互式系统相比,TIPSIE 还对效率给予关注,使用了动/静结合的程序性能预测方法 SPEASE 和过程级增量编译技术,以缩短交互式系统的响应时间.在今后的工作中,我们将进一步提高系统的交互能力,尽可能使得在并行化的各个阶段都能够引入用户的知识,并提高系统的稳定性和实用性.

References:

[1] Blume, W., Doallo, R., Eigenmann, R. Parallel programming with polaris. Computer, 1996,29(12):78~82.

[2] Wilson, R. P., French, R. S., Wilson, C. S. SUIF: an infrastructure for research on parallelizing and optimizing compilers. Sigplan Notices, 1994,29(12):31~37.

[3] Zhu, Chuan-qi, Zang, Bin-yu, Chen, Tong. Automatic parallelizing system. Journal of Software. 1996,7(3):180~186 (in Chinese).

[4] Ierotheou, C. S., Johnson, S. P., Cross, M., *et al*. Computer aided parallelisation tools (CAPTools)——conceptual overview and performance on the parallelisation of structured mesh codes. Parallel Computing, 1996,22(2):163~195.

[5] Hiranandani, S., Kennedy, K., Tseng, Chau-wen, *et al*. The D editor: a new interactive parallel programming tool. In: Proceedings of the Supercomputing'94. Los Alamitos, CA: IEEE Computer Society Press, 1994. 733~742

[6] Chen, Wen-guang, Yang, Bo, Wang, Zi-yao, *et al*. An interactive Fortran77 parallelizing system. Journal of Software, 1999,10(12):1259~1267 (in Chinese).

[7] Clement, M. J., Quinn, M. J. Automated performance prediction for scalable parallel computing. Parallel Computing, 1997,23(10):1405~1420.

[8] Blume, W., Eigenmann, R. Symbolic range propagation. Technical Report, 1429, Center for Supercomputing Research and Development, University of Illinois at Urbana-Champaign, 1995.

[9] Yang, Bo, Chen, Wen-guang, Zheng, Wei-min, *et al*. Fast and effective performance prediction for parallelizing compiler. In: Zhou, Xing-ming, Xu, Ming, Lou, Si-wei, eds. Proceedings of the 3rd Workshop on Advanced Parallel Processing Technologies. Changsha: Publishing House of Electronics Industry, 1999. 194~198.

[10] Blume, W., Eigenmann, R. The range test: a dependence test for symbolic, non-linear expressions. In: Proceedings of the Supercomputing'94. Los Alamitos, CA: IEEE Computer Society Press, 1994. 528~537.

[11] Burke, Micheal, Torczon, Linda. Interprocedural optimization: eliminating unnecessary recompilation. ACM Transactions on Programming Languages and Systems, 1993,15(3):367~399.

附中文参考文献:

[3] 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.

[6] 陈文光,杨博,王紫瑶,等.一个交互式的 Fortran77 并行化系统.软件学报,1999,10(12):1259~1267.

Several Critical Techniques in Constructing Interactive Environment of Parallelizing Compiler*

YANG Bo, WANG Ding-xing, ZHENG Wei-min

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

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

http://hpc.cs.tsinghua.edu.cn

Abstract: Interactive parallelizing system can provide user-friendly interactive functions and introduce user's knowledge to assist in parallelizing. It is a promising way to solve the problem of limited capability of automatic parallelizing. In this paper, an interactive environment of parallelizing compiler TIPSIE (interactive environment of Tsinghua interactive parallelizing system) is provided. Several critical techniques in constructing TIPSIE are also discussed, such as performance prediction, incremental compilation and data dependence query. The experimental results show that these techniques can get good result in improving the capability and efficiency of parallelizing compiler.

Key words: parallelizing compiler; interactive system; incremental compilation; data dependence query; performance prediction

* Received November 25, 1999; accepted February 23, 2000

Supported by the National High Technology Development Program of China under Grant No. 863-306-ZD-02; the National Natural Science Foundation of China under Grant No. 69933020