第7卷第3期
1996年3月

软 件 学 报
JOURNAL OF SOFTWARE

Vol. 7, No. 3
Mar. 1996

程序自动并行化系统*

朱传琪 臧斌宇 陈 彤

(复旦大学并行处理研究所 上海 200433)

摘要 有效的自动并行化系统能帮助用户充分地利用并行计算机资源.本文介绍了自动并行化的历史及现状,并着重介绍了作者所开发的自动并行化系统AFT(automatic Fortran transformer)及其特色和所用的关键技术.测试结果表明AFT系统在功能上已超越了传统的自动并行化系统.

关键词 向量化,并行化,数组私有化,相关性测试,过程间分析.

90年代,随着并行计算机的大量涌现,如何开发并行程序以充分有效地利用并行计算机资源已成为非常重要但又十分困难的问题.解决这一问题的有效途径之一是自动并行化,即由系统软件将串行程序自动变换为可执行的并行程序.自动并行化有以下几方面的好处.首先大量现存的串行计算机上运行的程序只有借助于自动并行化才能在并行机上有效运行.其次,编制新的程序时,借助于自动并行化工具,用户只需集中精力考虑算法的并行性而将一些繁琐的任务交给系统软件去完成,这样既能减轻用户的负担又可提高程序的可靠性和可移植性.第3,自动并行化是研制并行编程环境或开发新的并行语言的基础,普遍认为并行编程环境或新的并行语言是解决并行程序开发问题的最终途径.

自动并行化的重要性已被广泛接受,但自动并行化的技术还未成熟.本文将介绍自动并行化的历史、现状以及我们开发的自动并行化系统AFT(automatic Fortran transformer).AFT是针对Fortran语言的自动并行化系统,AFT采用了许多先进的技术,测试结果表明AFT系统在功能上已超越了传统的自动并行化系统.

1 背 景

自动并行化的前身是自动向量化.自动向量化的研究可追溯到70年代中期超级计算机的出现.在自动向量化的研究中David Kuck教授提出了相关性理论,通过相关性分析可以确定程序的向量化部分.对于阻碍向量化的相关性可通过各种程序变换尽可能消去,以便向量化成功.相关性分析和程序变换构成了自动向量化的基础.进入80年代后自动向量化系

* 本研究得到国家自然科学基金、国家863高科技项目与攀登计划基金资助.作者朱传琪,1943年生,教授,博士导师,主要研究领域为并行处理,高性能计算.臧斌宇,1965年生,讲师,主要研究领域为并行处理,高性能计算.陈彤,1968年生,讲师,主要研究领域为并行处理,高性能计算.

本文通讯联系人:朱传琪,上海 200433,复旦大学并行处理研究所

本文1995-08-30收到

统大量出现,例如 Illinois 大学的 Paraphrase,Rice 大学的 PFC,独立软件制造商的 KAP 和 VAST 以及机器制造商 Cray 和 Convex 等系统的向量化编译器,这些系统的出现标志着自动向量化技术的成熟.

80 年代中期,并行计算机的重要性日益明显,人们开始将注意力转移到自动并行化的研究上.自动并行化基础也是相关性分析,因而最初的工作是试图将自动向量化系统简单地扩充成为自动并行化系统,例如 KAP 和 VAST 都具有了并行化功能.但是这种扩充并不成功.90 年代初进行的一系列测试表明,代表当时最高水平的自动并行化系统对于超级计算机上运行的许多实际应用程序即所谓的 dusty deck 程序收效甚微.[1~3]这些测试结果使人们清醒地认识到了自动并行化的困难性,自动并行化与自动向量化有着本质的区别.并行化需要大粒度并行性而向量化只需要小粒度并行性.分析大粒度并行性时往往会遇到 CALL 语句和符号量等,传统的相关性分析方法能力有限不能分析这些情况,从而导致并行化失败.大粒度循环的复杂性又使绝大多数的程序变换毫无作用.面对这种情况,有人提出了悲观的论点:认为除非对 dusty deck 程序进行手工改写,否则自动并行化不可能有好的效果.[4,5]尽管如此,自动并行化的重要性仍然是一致公认的,因为它是其它方法的基础.即使自动并行化不能解决 dusty deck 程序改写中的所有问题,但是它的能力越强、手工改写的负担就越轻.

为了提高自动并行化的效果,近年来人们在基础理论和实用方法 2 方面开展了广泛的研究.在这些研究的基础上,出现了一些新的系统.如 Maryland 的 Tiny,Stanford 的 SUIF[6]和 Illinois 的 Polaris.[7]

部分的基础理论研究以线性方程、线性不等式、线性变换、整形规划等数学方法为基础,着重研究精确的相关性分析、存储相关消除和并行变换,提出了 Omega test[8], Unimodualr[9],数组扩张和私有化的一些方法.[10~12]还有一些研究工作集中在过程间的分析方法,试图消除过程对自动并行化的不良影响,这部分工作以 Rice 大学为代表.基础理论的研究结果一般都比较完美,但实际效果有时并不理想.例如:SUIF 系统实现了精确的相关性测试,但由于这一算法对循环界限要求太高,它的相关性分析能力整体上和 KAP 相当,有时还低于 KAP.SUIF 系统还实现了 Unimodular,但没有实现 KAP 中的 Loop Distribution,其实际并行变换能力低于 KAP.[6]而 Rice 大学关于过程间分析的一系列算法则对一些程序可取得效果.[5]

实用方法研究的代表是 Illinois 大学的 CSRD 的工作,他们主要是通过人工对实际应用程序的分析和改写,为自动并行化提供新方法.在分析了一些 dusty deck 程序后指出数组私有化(Array Privatization)、规约识别(Reduction Recognition)、复杂形式的递归标量(Inductive Variables)识别、符号数据相关性分析(Symbolic Data Dependence Analysis)和过程间分析是非常重要的技术.[2,13]这些新的方法,对实用程序非常有效且具有相当大的代表性,是新的系统必须考虑的问题.但实现这些技术有很大难度,有人甚至怀疑这些技术是否可能在真正的自动并行化系统中实现.[5]CSRD 开发了 Polaris 系统,初步说明了这些技术是可实现的.但系统本身还不太稳定,许多程序无法处理,尚未发表较完整的测试数据.这又从另一方面说明实现这些技术确实相当困难.

2 AFT系统技术介绍

本节介绍AFT系统中所采用的技术和实现.着重描述一些重要的、有特色的技术.

2.1 数组私有化(Array Privatization)

数组私有化主要试图识别出DO循环中用作临时变量(无跨循环流相关)的数组,并将其变形为循环实例的私有变量,从而消除该数组由存储单元引起的跨循环的非本质相关(反相关和输出相关),为并行化创造条件.对于在多层嵌套循环中临时数组的处理,就如最内循环中对临时标量的处理那样重要,数组私有化是大粒度并行化的关键手段之一,也是并行化与向量化的一个显著区别.

近年对如何实现数组私有化提出了不少方法.[11,12,14,15]考虑到实际上的效果和效率,我们设计了一种以相关性指导数据流分析的私有化方法.[16]为识别私有数组,通常方法直接求证所有的读引用只被本次循环实例中的写所覆盖,而我们还通过分析不同循环实例中写的关系,来识别私有数组.可处理一些其它方法难以克服的复杂读的情况.另外,对于跨循环求暴露读时,我们以相关性为指导,效率与精确度较高.在AFT系统中私有化技术与过程间分析、符号分析技术紧密结合,从而得到强有力的大范围的数组私有化能力.

2.2 相关性测试

相关性测试是自动并行化的基础.并行性识别与多种并行变换都需要用到相关性.相关性测试方法多年来已有许多研究,大部分方法都要求循环界限和数组下标是循环变量的线性函数(Affine Function).在这种假定下,这些方法已是精确高效的.但在实际程序中,循环界限或数组下标中往往会出现符号,有些符号是循环不变量,另一些则是递归(Inductive)变量.对于嵌套三角形循环,递归变量的通项是非线性多项式.为此相关性测试需要突破线性函数的限制,要有处理符号不变量和非线性多项式的能力.在AFT系统中,我们以传统的相关距离测试、GCD测试和Banerjee测试为基础,适当增加处理符号不变量与非线性函数的能力.AFT系统的相关性测试方法的效果涵盖了Polaris所采用的著名的实用测试方法Range Test.[17]

2.3 并行变换

AFT采用基于循环分布的并行变换[18],在根据相关图的强连通分支进行循环分布后,试图对所有循环进行并行化或向量化,然后进行循环合并.[19]另外,系统中还采用了一些有效的并行变换技术,如循环交换、复杂的归约(Reduction)和波前计算(Wave front)的识别.其中复杂的归约是指多个语句对一定数据区域的归约运算,在实际程序中尤为重要.归约运算满足交换律和结合律,如+,-,×,MIN,MAX等运算,在保证互斥后,可以并行执行.为高效执行,还需要通过变形,如变量私有化,来减少互斥开销.

2.4 过程间分析

高级语言强调程序的模块性,过程的广泛采用,成为并行化不可回避的问题.不仅要并行化含有过程调用的循环,还需要跨越过程获取精确的信息,以帮助完成过程内的并行化.

AFT的设计遵循所有分析和变形都必须考虑对过程的处理的原则,广泛采用各种过程间分析技术,包括过程间的数据流分析、向前替代、相关性分析、过程繁衍、过程嵌入等,以下简单介绍各个技术.

过程间数据流分析　过程间数据流分析求出进程调用语句相应的暴露集合,确定写集合、可能写集合和活跃变量集合.集合中,数组由正规段(Regular Section)表示.过程间数据流分析为过程间向前替代、数组私有化、死码删除等提供了基础.同时所得信息可用于相关性分析,与传统的调用副作用(Side Effect)分析相比,更精确,可支持私有化,调用语句自身的相关性会略有差异,但不影响并行化.

过程间向前替代　系统中,根据文献[20]中的算法框架,实现过程间的向前替代.采用的 JUMP 函数为入口虚参的表达式.过程间向前替代本身以及对其影响的操作,如 IF 消除、递归变量识别,都要求迭代至稳定.

过程繁衍(Cloning)　当过程调用的环境(Context)不同时,为不同的环境产生不同的过程版本,称为过程繁衍.该技术开销比过程嵌入小,而比简单的过程间信息传递更精确.文献[21]对过程繁衍有全面的介绍.但传统技术仅处理实参为常数值的情况.AFT 提出了更为一般性的过程繁衍技术,它可分析出实参之间的线性关系,将这些等式约束传递到多版本中,从而增强系统跨过程的符号分析能力,有助于相关性分析、变量私有化等.

过程嵌入(Inline)　过程嵌入是用过程体变形后替代调用语句,从而使整个程序只有一个主程序.AFT 中嵌入仅由需求驱动,尽可能保持程序的模块化.当要进行跨过程变形时,如循环交换、变量私有化、跨过程归约等变形,才进行嵌入.

2.5　存储优化

当前并行计算机多采用多级存储结构来提高存储访问速度,但存储操作仍然比运算慢,特别是对主存的访问,因此存储的开销是影响执行时间的重要因素,在程序并行化时必须予以考虑,否则可能并行化是得不偿失.另外,当采用 cache 技术时,并行化还可能导致 cache 冲突,使系统性能严重下降.

AFT 系统中,在进行并行化的同时,还考虑存储优化的问题,力图避免并行化带来 cache 冲突,同时尽量提高 cache 的利用率.所用技术主要有循环分割、循环交换、变量引用分析,并结合循环静态调度技术,产生高效的并行程序.

2.6　系统集成

AFT 拥有许多功能,不但这些功能本身就很复杂,而且还需要相互配合协调才能够完成并行化,系统集成对于完全自动的并行化工具是一个关键的方面.AFT 系统内部功能分为分析与变形 2 部分.分析以过程部分析为核心,在全局范围内获取、传递信息.变形对选定的循环进行相关性分析和并行变换.这样,分析在相对稳定的环境中进行,而复杂的多个关联数据结构的维护局限在较小范围内,有利于系统的实现.

3　试验结果

在实现了上述技术后,我们对 AFT 进行了测试,检验系统是否达到了设计的目的,实际并行化能力是否有显著的提高.

主要测试 Pefect Benchmark,该测试集是美国多所大学和公司为测试超级计算机性能而设计的,是科学计算应用程序类测试包,由 13 个程序组成,共约 60 000 行 FORTRAN 源程序.这些程序是从多个的应用领域中的源程序整理得到,充分体现了应用程序的特点,因此被广泛采用.传统的并行化工具能对小的核心(Kernal)类程序,如矩阵乘法、Linpack 等

取得成功,而对于应用测试包效果不佳.如对于 Perfect Benchmark 13个程序中一般仅2个有明显加速比.我们选择 Perfect Benchmark 作为主要测试包,一则因为它对应用程序具有代表性,有一定的难度,可反映系统并行化的能力.二来因为对它已有大量的研究,便于分析和比较.我们还收集测试了另外一些实用程序,检验所用技术的通用性.

测试平台是 SGI 公司的 Challenge 4L 系统.该系统是广泛使用的典型 SMP 系统,有4个 CPU(可扩充),通过总线共享主存,操作系统为 UNIX.系统中,对标准 FORTRAN 进行了扩充,增加了描述并行循环的指导语句,编译器可根据语句产生并行执行代码,从而利用多机资源并行执行.AFT 接受标准 FORTRAN 源程序,自动产生加入了并行循环的指导语句的程序,通过编译后,能并行执行.并行执行的时间与原始源文件串行执行时间的比较,可反映并行化工具的并行化效果.

为方便评价结果,我们选择 Challenge 配备的自动并行化工具 PFA(power FORTRAN accelerator)作参照,PFA 实际是 KAP 的 SGI 版本(1994－03 的版本),是传统并行编译器的典型代表.PFA 的外部接口与 AFT 相同,可自动地在 FORTRAN 源程序中正确加入并行循环的指导语句.二者的输出在同样的系统软件和硬件环境下运行,因此结果可充分反映两者并行化能力的差异,检验 AFT 是否相对于传统并行编译工具有显著的改进.

所有测试程序中都插入了时间检测函数(ETIME),有关执行时间的数据都是相应程序在机器上实际执行所获得的.需强调的是,AFT 的运行是完全自动的,除了给出源程序,用户不提供任何帮助,PFA 采用了最高级优化选项.

表1给出了矩阵乘法和 Linpack 的测试结果.这类程序是一般编译器可做好的,AFT 取得好于 PFA 的结果,说明 AFT 产生的程序执行效率较高.表2给出了 Perfect Benchmark 的测试结果.这类程序大部分是传统编译器做得不好的,而 AFT 对于6个程序取得明显加速比,这说明 AFT 的自动并行化能力比传统的系统有显著的提高,证明了 AFT 所采用的技术是有效的、可实现的.

通过对其它题目的测试,AFT 也取得了较好的结果,说明这些新技术具有一定的普遍意义.

从这些测试结果中,可看到 AFT 作为一个原型系统,不仅具有较强的功能,而且比较的稳定和可靠.而且也为今后的研究及产品化打下了良好的基础.

表1

矩阵乘法

Size	PFA Speedup	AFT Speedup
64×64	6.86	8.00
256×256	7.08	7.50

Linpack:Gaussian elimination

Size	PFA Speedup	AFT Speedup
100 single	* *	1.73
100 double	0.94	2.43
1000 single	* *	5.10
1000 double	2.09	4.39

表 2

Perfect Benchmark

	Serial	PFA runtime	AFT runtime	PFA Speedup	AFT Speedup
adm(ap)	27.46	79.37	29.35	0.35	0.93
arc2d(sr)	257.65	66.51	60.98	3.87	4.22
bdna(na)	86.58	80.30	24.86	1.07	3.48
dyfesm(sd)	16.50	82.51 *	16.15	0.39 *	1.02
flo52(tf)	28.67	15.87	12.68	1.81	2.26
mdg(lw)	257.17	254.46 *	66.47	1.01 *	3.87
mg3d(sm)	714.54	708.78	726.49	1.01	0.98
ocean(oc)	112.67	113.18	95.57	1.00	1.18
qcd(lg)	15.26	14.57	13.67	1.05	1.12
spec77(ws)	166.76	178.79	97.14	0.93	1.72
spice(cs)	6.41	* *	6.16	* *	1.04
track(mt)	8.67	10.23	8.50	0.85	1.02
trfd(ti)	27.09	74.99	17.06	0.36	1.59

* PFA 最高级优化不能处理,采用 PFA 缺省优化级别,* * 在缺省优化级别上,PFA 仍不能处理.

4 结　论

AFT 系统通过借鉴国际上的研究成果和自身的研究与开发,在自动并行化能力上取得了较大的提高.这一成功表明新一代的并行化技术对于实用程序的有效性,也证明了这些技术可在自动并行化工具中实现.我们同时应看到,Perfect Benchmark 中仍有不少程序未取得令人满意的结果,另外还需要用更多的测试程序包与实用程序对 AFT 进行测试,对并行化方法作更深入广泛的研究,进一步提高系统的能力.

致谢　参加 AFT 开发与测试工作的还有尹激雷、林源、王琦、李冰、张福波、唐卫宇、丁永华、施武、周诚彪、顾曙焱和国防科技大学的沈志宇、廖湘科.

参考文献

1　Cheng D Y, Pase D M. An evaluation of automatic and interactive parallel programming tools. Proc. of Supercomputing'91, 1991. 412~442.

2　Blum W, Eigenmann R. Performance analysis of parallelizing compilers on the perfect benchmarks programs. IEEE Trans. Parallel Distributed Syst., Nov. 1992,**3(6)**:643~656.

3　Singh J P, Hennessy J L. An empirical investigation of the effectiveness and limitations of automatic parallelization. Proc. Int. Symp. Shared Memory Multiprocessing, Tokyo, Apr. 1991.

4　Kennedy K. Compiler technology for machine independent parallel programming. Int. Journal of Parallel Programming, 1994,**22(1)**.

5　McKinliey K. Evaluation automatic parallelization for efficient execution on shared memory multiprocessors. ICS'94, 1994. 54~63.

6　Stanford Compiler Group. The SUIF parallelizing compiler guide. Technique Report 1994, 1994.

7　Blume B, Eigenmann R, Faigin K *et al*. The next generation in parallelizing compiler. Proc. of 7th Workshop on LCPC, Aug. 1994.

8　Pugh W. A practical algorithm for exact array dependence analysis. CACM, Aug. 1992,**35(8)**:102~114.

9 Banerjee U. Unimodular transformations of double loops. Advances in Languages and Compilers for Parallel Procession Cambridge, MA: MIT Press, 1991. 192～219.

10 Feautrier P. Data flow analysis of array and scalar reference. Int. Journal of Parallel Programming, 1991, **20**(1): 23～53.

11 Pugh W, Wonnacott D. Eliminating false data dependence using the omega test. Proc. of the SIGPLAN 1992 Programming Language Design and Implementation, 1992. 140～151.

12 Maydan D E, Amarasinghe S P, Lam M S. Array data—flow analysis and its use in array privatization. Proc. of 20th Annual ACM SIGACT—SIGPLAN Symp. on Principles of Programming Language, Jan. 1993.

13 Eigenmann R, Hoeflinger J, Li Z. Experience in the automatic parallelization of four perfect benchmark programs. Proc. of 4th Workshop on LCPC, Aug. 1991. 65～83.

14 Li Z. Array privatization for parallel execution of loops. Proc. of Int. Conference on Supercomputing, 1992. 313～322.

15 Tu P, Padua D. Automatic array privatization. Proc. of 6th Workshop on LCPC, Aug. 1993. 500～521.

16 Chen T, Zang B, Zhu C—Q. A new method for array privatization. Proc. of HPCC'94, Aug. 1994. 43～50.

17 Blume W, Eigenmann R. The range test: a dependence test for symbolic, non—linear expressions. Technical Report 1345, Univ. of Illinois at Urbana—Champaign, CSRD, April 1994.

18 Allen J R, Kennedy K. Automatic transformation of Fortran programs to vector form. ACM Trans. Program. Languages Syst., Oct. 1987, **9**(4): 419～542.

19 臧斌宇,朱传琪. 最小循环分布和最大循环合并. 中国青年计算机研究新进展, 1993. 76～80.

20 Callahan, Cooper K, Kenned K. Interprocedural constant propagation. Proc. of the ACM SIGPLAN'86 Sym. on Compiler Construction, SIGPLAN Not. 21, 7, July 1986.

21 Cooper K, Hall M, Kennedy K. A methodology for procedure cloning. Comput. Lang., 1993, **19**(2): 105～117.

AN AUTOMATIC PARALLELIZER

Zhu Chuanqi Zang Binyu Chen Tong

(Institute of Parallel Processing Fudan University Shanghai 200433)

Abstract An effective automatic parallelizer is critical for users to exploit the resources of parallel computer. This paper introduces AFT(automatic Fortran transformer), an automatic parallelizer which the authors developed on the background of the situation of the past and present of such kind systems. The characteristics of AFT and the advanced techniques it adopted are described in detail. Test result shows AFT is notably more capable of the parallelization than the conventional systems.

Key words Vectorization, parallelization, array privatization, dependence test, interprocedural analysis.