MPP 数据并行 FORTRAN 汇编代码模拟环境的设计与实现*

曾丽芳 杨灿群 胡子昂

(国防科技大学计算机系 长沙 410073)

摘 要 本文阐述了一种在单机上模拟 MPP 机执行并行汇编代码的方法。这种并行汇编代码由 MPP FORTRAN 编译器生成。由于在一个大型工程中，编译生成的目标代码运行环境不一定能及时具备，所以，研制汇编代码模拟环境 DPFAS 有助于及早验证编译目标代码的正确性和完备性。本文通过共享进程组在 SGI 工作站上实现多机环境模拟和远程地址访问模拟。此模拟环境的实现对 MPP 数据并行 FORTRAN 编译器的正确性验证和效率评估有一定的帮助。

关键词 模拟，符号再定位，共享进程组，MPP FORTRAN 编译器

分类号 TP313

The Design and Implementation of MPP DPFAS

Zeng Lifang Yang Canqun Hu Ziang

(Department of Computer, NUDT, Changsha, 410073)

Abstract In this paper, we introduce a method to emulate the execution of MPP parallel assemble code which is generated by MPP FORTRAN compiler. Such an emulator can help compiler workers test the compiler´s correctness and completeness in early time when runtime environment is not provided during huge engineering. We use shared process group on SGI workstation to emulate MPP environment and remote addressing. It´s helpful to correct and evaluate the MPP FORTRAN compiler.

Key words emulate, symbol relocation, shared process group, MPP FORTRAN compiler

MPP 数据并行 FORTRAN 语言是基于大规模并行处理机而开发的一种全局编程并行语言。在实现 MPP 数据并行 FORTRAN 编译器时，所遇到的主要问题在于：由数据分布而引起的诸如存储分配、数组下标地址计算及参数传递等。在编译实现过程中，我们的存储分配策略为：私有数据每个节点上复制一份；共享标量只有一份且存储于某一节点上；共享数组只有一份，分段存放于多个节点上，并要求同一共享数组在每个节点上的起始地址相同。共享数组下标地址计算包括两部分：计算数组元素所在的处理机和数组元素在处理机内的偏移[1]。

一个MPP 数据并行 FORTRAN 程序的编译过程为 (图 1)：

数据并行 Fortran 源程序 —预处理→ 编译可识别的表示形式 —编译→ MPP 汇编程序 —汇编→ MPP 目标代码 —装配→ MPP 可执行程序

图 1

Fig. 1

上述各步工作由不同人并行进行，且各环节相互依赖，例如，要验证编译的正确性，只有在汇编、装配、操作系统和硬件都已完成并保证正确性的前提下才能进行。为并行完成上述各步且保证正确性，我们在没有实际运行环境的条件下设计并实现了各级模拟环境。可执行程序模拟调试环境[2]模拟可执行程序的执行过程，它可将错误定位于操作系统或可执行代码。本文将介绍的DPFAS 模拟汇编代码的执行，它可将错误定位于编译或汇编装配，在汇编装配调试成功前，编译步可在此环境中模拟调试。下面分别阐述 DPFAS 的设计和实现。

* 国防预研基金资助项目
1998 年 10 月 5 日修订
第一作者：曾丽芳，女，1970 年生，讲师

1 功能界面

DPFAS 的用户界面是运行于X11 环境[3,4,5]下的基于 Motif 的应用程序。Motif 提供了丰富的构件(widget), 诸如菜单、按钮、文本区和对话框等, 每个构件的位置和外观都能灵活地设置。特别是构件可支持回调函数 (callback), 每个构件均可识别一事件集, 如键盘事件、鼠标事件、窗口显露事件等。构件一旦建立, 程序员可指定它应响应哪些事件并调用相应的处理程序。DPFAS 用户界面充分利用了 Motif 构件集及其设计风格, 界面美观实用, 操作方便。

DPFAS 用户界面主要由三部分组成: 一是菜单条, 在菜单条上提供了用于文件操作、编译链接、模拟运行、信息统计、选项设置和帮助信息等菜单以控制整个系统的工作。其中, 编译链接的主要任务是扫描并链接一个或多个汇编文件, 模拟运行通过设置断点可分为单步跟踪、多步运行及整个程序运行三种, 选项设置包括选择跟踪的处理机和设置各类访存统计选项等; 二是文本区, 用来显示、编辑汇编文件; 三是输出窗口, 用来输出编译链接汇编程序时详细的错误信息和模拟运行时用户程序的输出结果。另外, 还设计了若干对话框用于打开文件、显示提示和警告信息等, 其中有的对话框支持断点处寄存器值的显示。

2 总体结构

汇编代码模拟环境的整体流程如图 1。

图中: ¹ 读入一个或多个汇编文件, 经过一遍或多遍扫描而生成的符号表包括各种符号的各种属性, 如符号名、符号的大小、类型、共享特性、初值和地址等。

④ 存储分配为各种数据区分配空间。

㈣ 连接装配解决子程序调用问题, 生成完整的指令流序列。构成一个可执行程序的多个文件被扫描并被连接装配后, 生成的结构如图 2。

其中, 每一条指令的地址实际上指向一个结构, 此结构描述一条指令的属性。指令的属性主要包括: 操作码类型、操作数个数以及各个操作数。每个操作数指向一个符号表。

模拟环境的建立: 模拟环境在 SGI 单机工

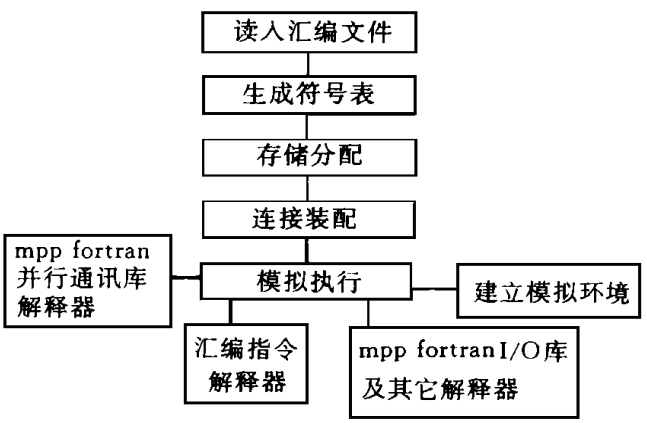

图 1 整体流程

Fig. 1 The whole flow chart

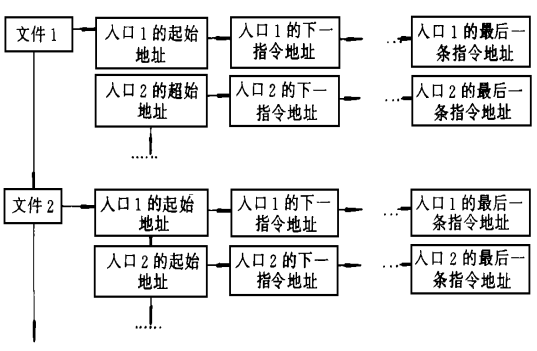

图 2 指令流

Fig. 2 The instruction flow

作站上实现, 此模拟平台和目标机处理机同属于 MIPS 芯片系列, 不同之处在于模拟平台为单机, 而目

标机为大规模并行多处理机。由于受单机内存的限制，我们只模拟四个处理机。我们利用 UNIX 共享进程组建立的模拟环境如下:

主进程读入用户程序 (汇编代码)，进行存储分配和连接装配，生成完整的指令流、符号名及其属性表。

主进程产生四个子进程，每个子进程共享主进程所定义的对象，包括指令流、符号名及其属性。每个子进程独立执行的主要功能为: ¹ 申请对应进程的私有栈并初始化; ④ 初始化一个结构以模拟私有栈链; ㈣ 申请一片内存数组以模拟通用寄存器、浮点寄存器和特殊寄存器; ¼ 提供一系列操作函数供解释器使用，如分配内存、从内存中取值等操作; ½ 调用指令解释器模拟指令流的执行。

指令解释器的主要工作为: ¹ 从过程入口地址链中找到主程序入口地址; ④ 从主程序入口地址指向的指令地址中取第一条指令作为当前指令; ㈣ 解释当前指令; ¼ 若㈣中的解释过程中没有发生转移，即不包含跳转指令、子程序调用或子程序返回指令，则继续取当前指令的下一条指令，并转㈣;

若㈣的解释过程中发生了转移，则保留当前指令，当前指令改变为转移后的地址所指指令，并转㈣。

若㈣的解释过程中发生了子程序返回，则释放子程序所占空间，当前指令改变为子程序调用前所保留的返回地址，并转㈣。

若㈣的解释过程中遇到了主程序结束，则结束指令模拟，并释放所有空间。

3 模拟过程中的难点

3.1 由于存储分配特点而引起的符号再定位及数组下标地址计算问题

引言中已说明在 MPP 系统中共享和私有数据的分配策略，但在单机模拟环境下，无法做到共享数组在四个处理机上具有相同的起始地址。因而，对所有数据，不管是共享还是私有，我们采用统一的存储分配法:

假设数组 A 在单个处理机上的大小为 size 个字节，则分配一片连续的大小为 4×size 个字节的空间，将此空间的起始地址 addr0 赋予符号 A。其中 addr0～addr0+ size- 1 的空间代表数据 A 在 0 号处理机上的存储空间; addr0+ size～addr0+ 2×size- 1 的空间代表数据 A 在 1 号处理机上的存储空间，addr0+ 2×size～addr0+ 3×size- 1 的空间代表数据 A 在 2 号处理机上的空间，addr0+ 3×size～addr0+ 4×size- 1 的空间代表数据 A 在 3 号处理机上的空间。

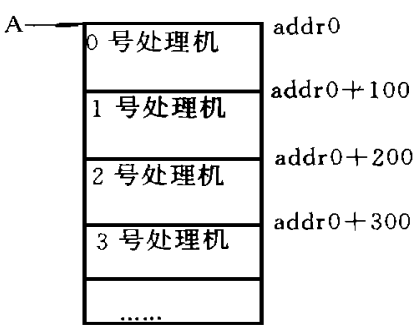

图 3 数组的空间分配

Fig. 3 The space allocation of dimension

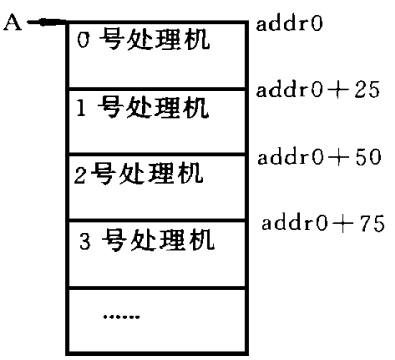

图 4 数组的空间分配

Fig. 4 The space allocaion of dimension

若 A 是大小为 100 的私有数组，则空间分配如图3。若 A 是大小为100 的共享数组，则空间分配如图 4。

综上所述，数据 A 所指的地址并非其在单个处理机上的实际地址，而只是一个相对的起始地址。所

以, 当遇到诸如 dla $2, A 和 sd $3, offset ($2) 的指令时, 就涉及到符号再定位问题。

dla 汇编指令表示将 A 的地址取入 2 号通用寄存器中, 由于符号 A 中所分配的地址的特殊性, 我们作如下处理:

¹ 从符号表中获悉 A 的共享特征、A 的大小 size 及 A 的起始地址 addr0。

④ 若 A 为共享数组, 则置标志位 flag 为 1, 然后将 flag、size 和 addr0 三个值作或操作, 将或操作的结果送 2 号寄存器, 即 $2 中的值形如图 5。

㈣ 若 A 为共享标量, 则置标志位 flag 为 2。

¼ 若 A 为私有变量, 则不置标志位 flag。

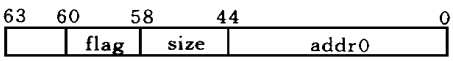

图 5 $2 的值

Fig. 5 The value of $2

sd 汇编指令表示将 $3 中的 64 位值存储到由 offset ($2) 算出的地址中。其中 offset 的值包含两部分: 处理机号及数组元素在处理机内的偏移。若数据为私有数组或共享标量, 则 offset 中的处理机号为 0。在计算数据的实际地址时, 从 $2 中地址值的 flag 标志位判断:

若 flag= 1, 则表示此数据为共享数组元素, offset 中必定包含有相应的处理机号 pn, 此时, 实际地址= pn×size+ addr0+ offset。

若 flag= 2, 表示此数据为共享标量, 则 offset 必定为 0, 实际地址= addr0。

若 flag= 0, 表示此数据为私有数据, offset 中的处理机号必定为 0, 实际地址= addr0+ 当前处理机号×size。

3.2 指令延迟槽处理

汇编代码中的分支跳转指令往往带有延迟槽, 实际执行时, 延迟槽指令在目标指令从内存中取出来时就开始执行, 即延迟槽指令中的操作数不依赖于目标指令。在模拟过程中, 我们这样处理延迟槽指令: 若遇到延迟槽指令总是有效的转移指令 (如JAL, J, JR), 则优先执行延迟槽中指令; 否则若遇到诸如 BLEZL、BGTAL 等分支指令, 则根据比较分支指令的比较结果决定是否作废延迟槽指令。例如, BGTZL $2, offset, 如果 $2 中的值大于零, 则先执行延迟槽中的指令, 然后再将控制转移; 否则, 跳过延迟槽中指令而执行下一条指令。

3.3 库调用的处理

MPP FORTRAN 中包含多类库调用, 总的处理思想如下: 不模拟库的动作, 而是将库子程序和模拟程序装配起来。在模拟器中设计一个结构, 每一个结构数据对应一个库函数, 结构中含一个指向该函数的指针。遇到库调用时, 从结构中找到相应的函数指针, 填好相应的入口参数, 然后转库子程序执行。库子程序运行结束后, 再将其返回结果送到相应的结果寄存器。各类库如果要进行修改, 则可以单独修改库程序, 而无需修改模拟过程。

上述处理应排除 exit、stop 和 abort 等关于程序正常或异常终止的库调用。遇到上述调用时, 不真正运行库调用, 而只结束指令解释器的解释, 不退出模拟环境, 并给用户一个信息, 告知MPP FORTRAN 程序的模拟执行已终止。

4 访存统计

在模拟过程中, 可以统计出某一共享变量在每个处理机上的局部、远程读写情况, 使得共享变量的访存情况一目了然, 从而有助于开发人员对共享数据的分布及程序的并行性效率进行评估和优化。

统计信息的界面表示如图 6。其中, ¹ 、④和㈣表示一组用户可以选择的信息, 由界面的其它部分显示。

¹ 表示在哪一层进行统计, 统计层可以分为: 不产生统计信息; 在程序结束时产生; 在最外层master 区的开始和结束处产生; 在 doshared barrier 处产生。

④ 表示统计变量在哪个作用域中的访存信息, 可分为: 整个程序的内存访问统计; 某个过程的内存访问统计。

㈣ 表示选择需统计的变量名。

statistics at：①

scope＝② var＝③

	局部访问		远程访问		迭代次数
	存	取	存	取	
PE0					
PE1					
PE2					
PE3					

图6 统计信息

Fig. 6 The statistics information

参考文献

1 赵克佳，沈志宇. 数据并行程序设计语言中分布数组的地址计算. 国防科技大学学报，1995，(4)

2 瞿国平等. 指令模拟的应用和特点. 计算机工程与科学，1996，(1)

3 Young A D. The X window system: Programming and Applications with Xt, OSF/Motif Edition. Prentice-Hall, 1990

4 Prentice-Hall. OSF/Motif Programmer's Guide, 1990

5 Prentice-Hall. OSF/Motif Programmer's Reference, 1990

(上接第70页)

2 谢寿生，徐永进. 微波遥感技术与应用. 北京：电子工业出版社，1987

3 K. Eldhuset. An Automatic Ship and Ship Wake Detection System for Space-borne SAR Images in Coastal Region. IEEE Transactio on Geos. and Res. 1996, 34 (4)

4 Michael D. Henschel, Richard B. Olsen, Pat Hoyt and Paris W. Vachon. The Ocean Monitoring Workstation: Experience Gained With RADSAT.

5 Rey M. T., T. Drosopoulos, and Petrovic D.. A search Procedure for Ships in RADARSAT Imagery. DREO Report No. 1305, 1996

6 Jiang, Q, S. Wang, D. Ziou, G. B. Benie, and A. El Zaart. Ship Detection in RADARSAT SAR Imagery. to appear in Proceedings of IEEE SMC'98 (San Diego)

7 Y. Delignon, R. Garello and Hillion. Statistical Modeling of Ocean SAR Imagery, IEE Proc. Radar, Sonar Naving 1997, 144 (6)

8 H. Andas and K. Eldhuset. Estimation of Central Moments in SAR Images for Homogeneity Testing. Tech. Rep. FFI/NOTAT-88/9003 1988 (in Norwegian)

9 F. T. Ulaby, F. Kouyate, B. Brisco, T. H. L. and Williams. Textural Information in SAR Images. IEEE Transaction on Geos. and Res. 1986, Vol. GE-24: 235～245