LS SIMD C 编译器的数据通信优化算法*)

A Data Communication Optimization Algorithm for LS SIMD C Compiler

王 晖 何华灿 陈 丹 胡 麒 张宝稳

(西北工业大学计算机系 西安710072)

Abstract LS SIMD is an embedded memory-shared massively parallel machine. In this paper, we present a deep study of data communication optimizations techniques of LS SIMD parallel compiler. First, some conceptions of data layout and data communication are given, and the data communication principle of LS SIMD is analyzed. Second, we propose an optimization algorithm of data communication, and discuss some aspects in detail, such as the representation of register state space, decision and generation of inner data communication and batch data communication in PE array.

Keywords Parallel compiler, Automatic data layout, Data communication, Optimization

1 引言

当前理想的程序自动并行化系统的实现存在许多难于解决的问题,因此较为流行的并行计算方法是利用并行语言编写并行程序,编译器对并行程序进行编译生成相应的节点程序执行。并行语言按并行执行的粒度分为基于任务的并行语言(主要面向一般应用领域的计算)和数据并行语言(主要应用于科学数值计算),典型的数据并行语言如 HPF。

对于数据并行语言而言,程序执行的并行性已由程序设计人员根据程序中的数据相关性给出。因此,如何确定数据的分布、优化数据的通信是影响并行程序执行效率的重要问题。数据分布大致可以分为两个阶段:首先对源程序中数据的相关性分析得到数据在抽象处理机上的分布,然后将抽象处理机上的数据分布映射到物理处理机上。数据分布的确定通常有以下几种实现方式:一种是由程序员给出抽象数据分布,编译器根据给出的抽象数据分布将数据映射到物理处理机上,如 HPF 提示的数据分布[1],另一种是由编译系统实现自动数据分布[2,3]。还有一些并行编译系统在编译过程中产生动态的数据分布提示信息,在程序运行时系统根据这些数据分布提示信息动态地改变数据分布[4]。

LS SIMD 是西安微电子技术研究所于1999年研制成功的嵌入式共享存储器型大规模并行处理机。LS SIMD 基本系统由32×32的处理单元阵列、共享存储器和主控器组成。处理单元阵列采用环网结构实现互连,每一处理单元(PE)不带有局部存储器,只包含8个通用寄存器[5]。我们针对 LS SIMD 设计实现了一个数据并行 C 语言编译器。该编译器能够自动对程序中并行结构的数据引用进行分析,充分考虑数据通信的优化,从而实现自动数据分布的功能[6]。本文就其中数据通信优化的实现方法进行详细的论述。

2 数据通信的基本概念及 LS SIMD 通信机制

2.1 基本概念

为了便于对通信优化的描述,首先给出相关的一些基本概念。

共享存储器对应的存储空间简称共享存储空间,它是一维线性连续存储空间;寄存器空间指 PE 阵列中的寄存器构成的三维存储空间,空间中的每一点用一个三元组(x,y,r)表示,分别对应 PE 所在的横向坐标、纵向坐标与当前的寄存器号,具有相同寄存器号的点构成一个寄存器平面;迭代空间指并行循环结构中由循环向量 $I=(i_1,i_2,\cdots,i_n)^T$ 构成的 N 维线性空间;一个 M 维数组对应一个 M 维的数组空间,数组在共享存储空间中按行顺序存储;M 维数组引用的下标对应数组空间的一个 M 维整数向量。在循环迭代中,每一数组下标对应循环向量 I 的一个函数。对于一个确定的数组引用,如果每一下标均为 I 的仿射函数,则该数组引用的下标可以用一个从迭代空间到数组空间的仿射变换表示。相应下标表示为 $S=G\times I+G_0$,G、G_0 分别为相对于 I 为不变的 N×M 矩阵和 M 维的行向

*)课题研究得到国防九五预研项目的支持。

量。

为了简化讨论，本文假定并行循环结构中循环变量的下界为0，步长为1，循环变量上界均为相对循环迭代的不变量。

2.2 LS SIMD 的数据通信机制

LS SIMD 的数据通信方式包括以下5类：

(1)共享存储器与单个 PE 的通信，其形式为 a↔(x,y,r)。a 为共享存储空间中的简单变量。

(2)共享存储器到 PE 阵列的广播通信。形式为 a→(*,*,r)。* 表示该维的所有元素。

(3)共享存储器与 PE 阵列的成组通信。在一个并行结构中，LS SIMD 的控制器中用三个数据地址寄存器(Ar0-Ar2)存储成组数据访问的相关信息：Ar0用于存放每一次存取的起始数据地址，为数组 A 的起始地址与数组内偏移地址之和，记为 $A+\Delta_0$；Ar1用于设置数据访问的列间步长 Δ_1，Ar2用于设置行间步长 Δ_2。其形式为：

$(A,\Delta_0,\Delta_1,0)\leftrightarrow(*,y,r)$(对于 N 维列向量)或$(A,\Delta_0,\Delta_1,\Delta_2)\leftrightarrow(*,*,r)$(对于32×32矩阵)。

(4)PE 之间的邻近通信。邻近通信方式实现 PE 与上、下、左、右等相邻 PE 的数据通信。形式为：(x,y,r)↔(x+1,y,r)或(x,y,r)↔(x,y+1,r)。

(5)PE 阵列内的广播通信指定阵列中的列或行的数据广播到整个 PE 阵列。形式为：(*,y,r)↔(x,y,r)或(x,*,r)↔(x,y,r)。

3 数据通信优化

并行编译器的数据通信优化是在中间代码生成后进行的。数据通信优化是根据并行计算的要求，选择适当的数据通信方式，实现数据在寄存器空间的优化分布，使数据通信的时间在整个数据分布中尽可能小。

3.1 寄存器空间的状态描述

在并行计算中，寄存器多是成组参与数据分布和引用的。因此，我们将同一寄存器平面上的行或列或整个寄存器平面作为基本分配单位，使用一个线性表来描述目标代码生成过程中的寄存器空间状态。表1举例说明了线性表中每个节点可能描述的不同类型信息。

寄存器空间因计算或数据通信而导致状态的更新，状态表中的信息具有冗余性，所以当计算或数据通信时，需要修改状态表中所有相关的节点数据，及时将更新的数据写回共享存储空间。

3.2 PE 阵列内部通信的判定

我们分别用 S_r、OP_r 表示寄存器状态空间中所有元素集和 PE 阵列内部通信的基本操作集，相应的内部通信用一个二元组(s,op)来表示，s 表示空间中相应的一个元素集、一个列、一个行或一个寄存器平面，$s\subseteq S_r$，op 为对应的基本操作，$op\in OP_r$。如果待实现的数据分布能够通过 PE 阵列内部通信实现，则必存在一个有序内部通信集$\{(s_i,op_i):1\leqslant i\leqslant t\}$，t 为通信步数。利用 PE 阵列内部通信实现数据分布的过程就是这个有序内部通信集存在的判定及构造的过程。在实现中首先查找状态表，判定节点对应的共享存储空间的信息与待实现的分布之间的联系，以确定是否能够通过相应节点的广播通信(对于行或列)或邻近通信实现，然后进一步判定相应节点的操作步骤，从而得出相应的判定结果及可能的通信过程。

表1 不同类型的寄存器空间状态信息(PE 阵列规模为 N×N)

类型	寄存器空间坐标	对应共享存储空间信息	状态	在寄存器空间中的相对偏移	说明
行	(x,*,r)	$(A,\Delta_0,\Delta_1,0)$	L-装入	$(\Delta_x,0)$ $\lvert\Delta_x\rvert\leqslant N/2$	对于数组 A，从相对偏移 Δ_0起始，列间距为 Δ_1的 N 个元素，在 PE 阵列中向右偏移 Δ_x
列	(*,y,r)	$(B,\Delta_0,\Delta_1,0)$	U-更新	$(0,\Delta_y)$ $\lvert\Delta_y\rvert\leqslant N/2$	对于数组 B，从相对偏移 Δ_0起始，列间距为 Δ_1的 N 个元素，在 PE 阵列中向下偏移 Δ_y
N×N 矩阵	(*,*,r)	$(C,\Delta_0,\Delta_1,\Delta_2)$	L-装入	(Δ_x,Δ_y) $\lvert\Delta_x\rvert\leqslant N/2$ $\lvert\Delta_y\rvert\leqslant N/2$	对于数组 C，从相对偏移 Δ_0起始，列间距为 Δ_1、行间距为 Δ_2的 N×N 个元素，在 PE 阵列中向右偏移 Δ_x、向下偏移 Δ_y

3.3 成组通信的判定

在数据分布过程中，如果 PE 阵列的内部数据通信不能满足数据分布的要求，则需要将数据从共享存储器中加载。此外，输出的计算结果也需要存储到共享存储器。PE 阵列与共享存储器进行数据交换时，我们总是希望能够使用成组通信以减少总的通信时间。下面给出成组通信在并行循环结构中的判定定理及相应通信参数的确定方法。

对于给定的并行循环结构 PS(I,U)中数组 A 的一个引用，具有如下的形式：$A(f_1,f_2,\cdots,f_M)$，其中第 k 维下标 $f_k(i_1,i_2,\cdots,i_n)$为循环向量$(i_1,i_2,\cdots,i_n)^T$确定的一个数组下标函数，我们主要讨论一维和二维数组的成组通信。当数组引用的维数 M>2时，先通过仿射变换将数组映射到一维或二维数组空间上，然后考虑

基于一维或二维数组空间中成组通信实现。

3.3.1 一维数组的成组通信的判定 对于并行循环迭代中形如A(f(a))的一维数组引用，显然若相邻迭代满足：

$$f(i_{1,k},i_{2,k},\cdots,i_{n,k})-f(i_{1,k-1},i_{2,k-1},\cdots,i_{n,k-1})=\Delta \quad (1)$$

其中，Δ 为一常数，k 满足：$1<k\leqslant\prod_{i=1}^{n}(u_i+1)$，则该数组引用能够实现成组通信。

确定符合式1的函数 f 是比较困难的。但在许多科学计算中，一维数数组引用的下标是迭代向量的仿射函数。相应第 k 次迭代引用函数 f_k 可以表示为如下的仿射函数：

$$f(k)=f(i_{1,k},i_{2,k},\cdots,i_{n,k})=G\times I_k+g_0$$

其中 $G=(g_1,g_2,\cdots,g_n)$ 为一常向量，g_0 为一常数。$I_k=(i_1,i_2,\cdots,i_n)^T$。则：

$$f(k)-f(k-1)=G(I_k-I_{k-1})$$

从而我们可以得到如下的判定定理。

定理1 对于给定并行循环结构 PS(I,U)中的形如 $A(G\times I_k+g_0)$一维数组引用，如果对于任意 g_l，$1\leqslant l<N$，满足：$g_l=g_m+\sum_{j=l+1}^{n}g_j\cdot u_i$，则该一维数组引用能够实现成组通信。

证明：(略)

所以该数组引用的任意相邻迭代等间距，该数组引用可以实现成组通信。定理得证。

推论1 对于满足定理1条件的数组引用，其实现成组通信的数据访问起始地址与数组 A 的起始地址之差(数组内部偏移)为 g_0，引用间距为 g_n。

推论2 在单循环的并行循环迭代中，如果存在一个一维线性数组引用，其下标是循环变量的仿射函数，则该数组引用可以实现成组通信，相应数据访问的数组内部偏移为 g_0，引用间距为 g_1。

3.3.2 二维数组引用的成组通信判定 一般情况下在一个并行循环中，二维数组的引用需要同时使用行间距和列间距来判定是否满足成组通信的条件，以确定是否能够实现成组通信。

假设对于并行循环迭代 I 中的二维数组引用的下标向量为迭代向量的仿射变换，其形式为 $A(G\times I^T+G_0)$。G、G_0分别为2×N 常系数矩阵和2维常向量，$G_0=(g_{0,1},g_{0,2})$。数组下标向量为迭代向量的仿射变换。我们给出二维数组引用成组通信判定定理。

定理2 对于并行循环迭代 I 中的二维数组引用 $A(G\times I+G_0)$，$H=(h_1,h_2)$为对应二维数组的上界。如果存在 m 满足：

(1)$G=\begin{bmatrix} g_{1,1} & \cdots & g_{1,m} & \cdots & 0 \\ 0 & \cdots & 0 & g_{2,m+1} & \cdots & g_{2,N} \end{bmatrix}$

(2)对于任意 p，q，$1\leqslant p<m$，$m+1\leqslant q<N$

$$g_{1,p}=g_{1,m}+\sum_{j=p+1}^{m}g_{1,j}\cdot u_{1,j} \quad (2)$$

$$g_{2,q}=g_{2,m}+\sum_{j=q+1}^{n}g_{2,j}\cdot u_{2,j} \quad (3)$$

则该数组引用可以实现成组通信。

证明：(略)

推论3 满足定理2的数组引用实现成组通信的数组内部偏移为 $g_{0,1}h_2+g_{0,2}$，列间距 $\Delta_1=g_{2,N}$，行间距 $\Delta_2=(g_{1,m}-1)h_2+(h_2-G_2(0,\cdots,0,u_{m+1},\cdots,u_n)^T)+g_{0,2}=g_{1,m}h_2-G_2(0,\cdots,0,u_{m+1},\cdots u_n)^T)+g_{0,2}$。

3.4 数据通信优化算法的生成

根据前面讨论的结果，数据通信优化算法概括如下：

(1)计算待进行分布(期待分布)的数据在共享存储空间的位置。如果待分布的数据为数组引用，则计算出相应的数组内部偏移 Δ_0、列间距 Δ_1和行间距 Δ_2及在寄存器空间中的数据对准信息(Δ'_x,Δ'_y)。

(2)查找寄存器空间状态表，根据每一节点对应共享存储空间的信息判断待进行分布的数据是否已存在于寄存器空间中，如已存在，计算该节点的数据分布与期待分布的距离$(\Delta'_x-\Delta_x,\Delta'_y-\Delta_y)$。计算相应的移动通信的代价。

(3)如果期待分布的数据具有如下的形式，则考虑使用广播通信实现数据分布。

如果 $\Delta_1(N-1)=-\Delta_2$，查找寄存器空间状态表中对应共享存储空间的信息为$(A,\Delta_0,\Delta_2,0)$且类型为行的节点。由系统指定一个寄存器平面，将该行广播到该寄存器平面上。计算相应的广播通信代价。

如果 $\Delta_1=0$，查找寄存器空间状态表中对应共享存储空间的信息为$(A,\Delta_0,\Delta_1,0)$且类型为列的节点。由系统指定一个寄存器平面，将该列广播到该寄存器平面上。计算相应的广播通信代价。

(4)判断是否能够通过成组通信实现数据分布。

如果期待分布的数组为一维数组或二维数组，则根据定理1、定理2判断能否实现成组通信，如能实现，根据推论1，推论3计算相应的成组通信参数，由系统指定一个寄存器平面，确定在该寄存器平面上实现期待分布的成组通信步骤及通信代价。

(5)由系统指定一个寄存器平面，确定从共享存储空间到该寄存器平面的单个 PE 通信实现期待分布的通信步骤及相应代价。

(6)从(2)-(5)中选择具有最小通信代价的通信步骤实现相应的期待分布。

(7)转1，直到当前计算所需数据分布全部实现。

(下转第115页)

4.5 以服务为单位的实时监测

本系统实现了以服务为单位的实时监测。当一个服务运行时，服务的承载方和备份方都有相应的监测手段，对服务状态进行监测。

服务的承载方要对服务使用的硬件资源，目前包括本服务使用的网络接口和 SCSI 接口的状态进行监测。由于本系统采用了多进程、多线程的软件结构，对上述硬件资源的监测可以由独立的线程来完成。当服务启动时，系统同时创建两个线程，分别对服务使用的网络接口和 SCSI 接口的状态进行监测。

同时服务的监视方也要对服务的运行状态进行监视，监视程序由服务提供，每个服务都应提供相应的启动、停止、重新启动和监视等操作的脚本文件，备份服务器上创建一个独立的进程，执行服务的监视脚本。

当上述任何一个线程或进程发现服务及其使用的资源出错时，将进行自动的服务迁移，在承载服务器上停止该服务，迁移到备份服务器上。

4.6 服务事件处理线程的设计

本系统采用了统一的事件处理机制对进程与进程之间、线程与线程之间、不同节点之间的通讯进行处理，事件是上述通讯传递的信息的总称。事件可能是由系统产生的，也可能是用户发起的，还可能是对等服务器向本地主机发送的。事件有两种类型，一种是全局事件，对所有服务有效，如监听到对等机失败、诊听线的所有本地 NIC 崩溃、诊听线的所有远程 NIC 崩溃、停止所有的服务、查询所有服务的状态等事件；另一种是局部事件，对某个或某几个服务有效，如提交一个服务、停止一个服务、删除一个服务、在本地启动一个服务、在本地关闭一个服务、查询服务的状态、查询服务的配置信息、将服务从一个主机迁移到另一个主机等事件。本系统设计了一个全局事件队列，事件全部发送到该队列中，并且按插入队列的次序，顺序处理。

由于各个服务之间是互相独立的，只与各个服务相关的事件完全可以并发处理。因此，为了提高并行处理能力，使事件得到及时处理，采用多线程技术。为每个服务设计了一个局部事件队列，并创建了一个线程来处理服务自己的局部事件。

这样，全局事件队列中的事件有以下几种处理方式：1）直接处理；2）复制成多个事件，发送到相关的服务的事件处理队列中；3）直接转发给对应服务的事件处理队列；4）生成新的事件，向对等服务器发送。

服务事件处理线程等待在服务自己的局部事件队列上，当有事件到来时，对事件进行处理，完成后，继续等待在局部事件队列上。

由于有多个线程要同时访问全局事件队列和每个服务的局部事件队列，因此必须分别对上述所有队列互斥访问，并确保不会产生死锁。

结束语 2000年11月25日，由总装司令部组织，北大杨芙清院士任鉴定委员会主任委员，对高可用性系统软件 HHA 进行了鉴定，鉴定委员会一致认为："该成果是具有完全自主版权的软件产品，对军事信息系统具有实际应用价值，对我国软件产业的发展有积极作用。该产品具有创新性，在双服务器结构、高可用性技术上达到了目前国内领先水平、国际先进水平。"服务级的管理是该系统中最主要的一个功能和特点。

参 考 文 献

1 Castagnera K, Cheng D, Fatochi R, et al. Clustered Workstations and Their Potential Role as High Speed Computer Processors: [NAS Computational Services Technical Report RNS-94-003]. April 1994

2 Sterling T, Becker D, Savarese D, et al. BEOWULF: A Parallel Workstation for Scientific Computation. In: Proc of the 1995 Int Conf. on Parallel Processing (ICPP), August 1999

（上接第118页）

在步骤(2)-(4)中，系统是按如下方法确定寄存器平面的：如果存在一个寄存器平面从未参与数据分布，则指定该寄存器平面，否则按照最远引用先分配的原则指定寄存器平面，如果该寄存器平面包含先前计算更新的数据，则应将数据存储到共享存储空间中。

结论 本文结合 LS SIMD 并行 C 编译器的具体设计，在分析了 LS SIMD 数据通信机制的基础上，对数据通信优化的关键技术进行了深入研究并提出了解决方法，主要包括以下内容：

1）给出了与数据通信优化相关的一些基本概念，并描述了这些概念间的相互联系，针对并行计算的一般特点，对问题进行了相应的简化操作。

2）针对自动数据分布给出了相应的数据通信优化算法，并对其中的寄存器空间的状态表示、各类数据通信的选择及生成方法进行了详细的描述。

参 考 文 献

1 High Performance Fortran Forum. High Performance Fortran Language Specification. Version 2.0. 1997

2 Kennedy K. Automatic Data Layout for High Performance Fortran. In: Proc. of Supercomputing, 1995

3 Kandemir M. A Linear Algebra Framework for Automatic Determination of Optimal Data Layouts. IEEE Trans. Parallel and Distributed Systems, 1999, 10(2)

4 Bal H E. Object Distribution in Orca using Compile and Run-Time Techniques. In: Proc. Conf. on Object-Oriented Programming Systems, Languages and Applications, 1993

5 沈绪榜 MPP 嵌入式计算机设计. 清华大学出版社, 1999

6 Wang Hui. Implementation of Data Parallel C Compiler of LS SIMD. In: Proc. of 7th Joint Int Computer Conf. 2000