基于数据空间融合的全局计算与数据划分方法*

夏 军+, 杨学军

(国防科学技术大学 计算机学院,湖南 长沙 410073)

A Data Space Fusion Based Approach for Global Computation and Data Decompositions

XIA Jun+, YANG Xue-Jun

(School of Computer, National University of Defense Technology, Changsha 410073, China)

+ Corresponding author: Phn: +86-731-6695352, E-mail: ddk@nudt.edu.cn, http://www.nudt.edu.cn

Received 2003-04-21; Accepted 2003-12-08

Xia J, Yang XJ. A data space fusion based approach for global computation and data decompositions. *Journal of Software*, 2004,15(9):1311~1327.

http://www.jos.org.cn/1000-9825/15/1311.htm

Abstract: Computation and data decompositions are key factors of affecting the performance of parallel programs running on distributed memory multicomputers. This paper presents a theoretical framework of data space fusion and an effective global computation and data decomposition approach based on it, which can be used to solve computation and data decomposition problems on distributed memory multicomputers. The approach can exploit the parallelism of computation space as high as possible, use the technique of data space fusion to optimize data distribution, and search the optimizing global computation and data decompositions. The approach can also be integrated with data replication and offset alignment naturally, and therefore can make the communication overhead as low as possible. Experimental results show that the approach presented in the paper is effective.

Key words: distributed memory multicomputer; parallel compiler; computation decomposition; data decomposition; data fusion

摘 要: 计算与数据划分问题是影响并行程序在分布主存多处理机中执行性能的重要因素,也是并行编译优化的重点.针对该问题,提出了一套关于数据空间融合的理论框架,并基于该框架给出了一种有效的全局计算与数据划分方法,用于分布主存计算环境中的计算与数据划分问题的求解.该方法能够尽量开发计算空间的并行度,利用数据融合技术优化数据分布,并能搜寻优化的全局计算与数据划分.该方法还能很自然地与数据复制以及偏移常量的对准结合在一起,从而使得数据通信量尽可能地小.实验结果表明了所提出方法的有效性.

关键词: 分布主存多处理机;并行编译器;计算划分;数据划分;数据融合

中图法分类号: TP311 **文献标识码**: A

* Supported by the National Natural Science Foundation of China under Grant Nos.69825104, 69933030 (国家自然科学基金)

作者简介: **夏军**(1976－),男,重庆人,博士,助理研究员,主要研究领域为并行编译优化;**杨学军**(1963－),男,博士,教授,博士生导师,主要研究领域为并行计算机系统结构,并行操作系统,并行编译.

分布主存多处理机是用于高端科学计算最流行的并行计算机之一,并被越来越多地用于解决大规模的科学计算问题.分布主存多处理机的可扩展性好,但由于它没有单一的全局地址空间而使得编程较困难.在分布主存的计算环境中,本地访存的速度一般要远远快于远地访存的速度,因此为了使程序能在分布主存多处理机中高效执行,用户或并行编译器必须有效地分布代码和数据,以使得处理器所需访问的数据都尽量在本地内存中,从而减少数据迁移开销.近十几年来,许多研究人员对如何在分布主存的计算环境中进行计算划分与数据划分的问题进行了研究[1~7].Chen 和 Chang[1]针对传统维对准的不足,提出了一种扭曲对准的数组对准策略,从而使得与维对准相比较,他们的方法能对更复杂的情形进行处理.但他们的方法是在迭代空间的划分已确定的基础上进行的,并且也没有考虑数据复制与偏移常量对准问题,从而其方法不能使得数据通信量尽量小.Chang 和 Chu[2]提出了两种针对带 3 个循环索引变量的线性下标和二次方下标($ai^2+bi+cj+d$)的数组对准函数,他们没有考虑数据复制以及系统的全局计算与数据划分方法,并且其方法只适用于解决带 3 个循环索引变量的线性下标和二次方下标的数组对准问题.Shih 和 Sheu[3]提出了一种语句级的基于超平面的无通信计算和数据划分方法.他们将嵌套循环中的每个语句看做是一个调度单位,即每个语句都有一个独立的迭代空间,并分别考虑它们的划分.由于其方法是基于超平面的,所以最终求得的计算划分与数据划分的并行度有限.另外,虽然将每个语句看做是一个独立的调度单位可以在一定程度上拓宽他们的方法的适应面,但同时也增加了代码变换的难度.Lim[4]提出了一种基于仿射变换的最大化并行度的计算划分方法.他将单个语句看做是一个调度单位,并从程序全局分多个层次(即无同步,$O(1)$同步和 $O(n)$同步)来考虑对迭代空间进行划分.他只考虑了对迭代空间进行划分而未考虑对数据空间进行划分,所以其方法一般只适用于共享主存或分布共享主存多处理机系统.Chen 和 Sheu[5]通过分析嵌套循环的所有数组访问来求得数据相关距离向量[8],并通过它来划分迭代空间和组织被同一个迭代划分所访问的数据元素.虽然他们给出了无复制和带复制的无通信计算与数据划分方法,但其方法仅适用于单个嵌套循环且还要求同一数组的所有引用的数据访问矩阵[9]必须相同.另外,该方法也没有系统地给出数据是如何分布的.Ramanujiam 和 Sadayappan[6]提出了一种基于超平面无通信的计算和数据划分方法,他们首先考虑数据空间的超平面划分,然后通过数据划分来求出计算划分.但他们只给出了单个嵌套循环的计算和数据划分方法,且所考虑的数组只是二维数组.另外,由于他们的方法是基于超平面的,所以最后求得的计算和数据划分的并行度受到限制.Huang 和 Sadayappan[7]将文献[6]的方法推广,考虑了任意维数组和多个嵌套循环的计算与数据划分方法.同样,由于他们的方法也是基于超平面的,所以最后求得的计算和数据划分的并行度也有限.另外,他们没有考虑数据复制问题,并且还假设计算空间和数据空间无限大,从而强化了无通信的计算和数据划分的限制条件,因而限制了其方法的适用范围.

本文针对分布主存计算环境中计算与数据划分问题,提出了一套关于数据空间融合的理论框架,并基于该框架给出了一种有效的全局计算与数据划分方法.该方法首先确定包含嵌套循环所有数据相关距离向量的向量空间,并通过它来划分嵌套循环的迭代空间,然后再对每个数组的每次不同引用分别求出迭代划分块所访问的数据空间,并将同一数组所有不同引用被同一个迭代划分块所访问的数据空间融合成一个统一的数据空间,然后再据此对该数组的数据空间进行划分,从而使得计算与数据能够有效地对准.针对数据分布问题,我们提出了一套关于数据空间融合的理论框架,以使得我们能够根据迭代空间的划分来有效地组织数据,优化数据分布.针对多个嵌套循环的全局计算与数据划分问题,我们基于数据空间融合的理论框架,给出了一种搜寻优化的全局计算与数据划分的贪婪启发式搜索算法.我们所提出的方法考虑了多个嵌套循环中具有一般线性下标的任意维数组的计算和数据划分问题,并且我们的方法能够尽量开发计算划分和数据划分的并行度,该方法还能很自然地与数据复制以及偏移常量的对准结合在一起,从而使得数据通信量尽可能地小.最后,我们通过对一组基准测试程序的测试验证了本文所提出的方法的有效性.

本文第 1 节描述本文要使用的一些基本术语与假设.第 2 节讨论单个嵌套循环的计算与数据划分方法.第 3 节以单个嵌套循环的计算与数据划分方法为基础,讨论多个嵌套循环的全局计算与数据划分方法.第 4 节给出实验结果.第 5 节总结全文.

1 基本术语与假设

给定一个 n 重嵌套循环 L 中的 m 维数组 X,我们用 $\bar{I}=(i_1,...,i_n)^T$ 来表示迭代向量(其中 $i_1,...,i_n$ 从左至右分别代表最外层循环索引变量直至最内层循环索引变量),用 L^n 来表示 L 的迭代空间.我们假设循环上下界和数组的下标表达式为包含它们的循环索引变量和常量的仿射函数,这样,数组访问可以表示为 $A\bar{I}+\bar{o}$,$m\times n$ 维矩阵 A 被称作访问矩阵,具有 m 个元素的向量 $\bar{o}$ 被称作偏移向量[9].

我们假设嵌套循环为紧耦合嵌套循环,而对于某些松耦合嵌套循环,我们可以利用代码下沉(code sinking)、循环分离和循环合并等变换技术[8,10]来将其转换成紧耦合嵌套循环.我们还假设嵌套循环只含数组变量,对于标量,我们可以通过标量扩张或标量私有化对其进行预处理.另外,我们也对嵌套循环进行了归一化(normalized)[8]处理,以使每层循环的索引变量都以 1 为步长递增变化.

为了讨论方便,我们用 $span\{\bar{b}_1,...,\bar{b}_l\}$ 表示由向量 $\bar{b}_1,...,\bar{b}_l$ 扩展所构成的空间,用 $span\{\Omega\}$ 表示由集合 Ω 中的向量扩展所构成的空间,用 Q 表示有理数域,用 Q^n 表示由全体 n 维有理数向量所构成的空间,用 $\dim(\Psi)$ 表示向量空间 Ψ 的维数,用 $\dim(\bar{b}_1,...,\bar{b}_l)$ 表示向量组 $\bar{b}_1,...,\bar{b}_l$ 的秩.我们还令 $\bar{b}+\Psi=\left\{\bar{q}\middle|\bar{q}=\bar{b}+\bar{p},\bar{p}\in\Psi\right\}$,其中 $\bar{b}$ 为 $n\times 1$ 维的列向量,而 Ψ 为由 $n\times 1$ 维的列向量所组成的向量空间.本文所考虑的向量空间都是数域 Q 上的向量空间.

2 单个嵌套循环的计算与数据划分

2.1 划分计算空间

给定一个 n 重嵌套循环 L 以及其中被访问的若干数组 $X_1,...,X_p$ ($p\geq 1$),本节我们将给出对迭代空间 L^n 进行划分的方法.

2.1.1 求解数据相关距离向量空间

令 Ψ^L 为包含 n 重嵌套循环 L 所有数据相关距离向量的向量空间,为了求出 Ψ^L,我们可以首先对每个数组分别进行相关性分析,求出与它相关的数据相关距离向量,令包含与数组 X_j 相关的所有数据相关距离向量的向量空间为 $\Psi^L_{X_j}$ ($1\leq j\leq p$),那么 $\Psi^L=span\{\Psi^L_{X_1}\cup...\cup\Psi^L_{X_p}\}$.通过对嵌套循环进行精确的数据相关性分析[8,10]和消除冗余计算[5]的操作,我们就能得到维数尽可能小的 Ψ^L,从而最大限度地开发嵌套循环的并行度.

2.1.2 线性计算划分

在求得了包含 n 重嵌套循环 L 所有数据相关距离向量的向量空间 Ψ^L 后,我们就能利用 Ψ^L 对嵌套循环 L 的迭代空间进行划分.若 $\dim(\Psi^L)=n$,那么嵌套循环的整个迭代空间只能被划分成一个可独立并行执行的迭代划分块,即该迭代空间本身,因此在本文后面的讨论中,我们都缺省假设 $\dim(\Psi^L<n)$,即只讨论嵌套循环的迭代空间能被划分成多个独立的可被并行执行的迭代划分块的情形.

定义 1. 给定一个 n 重嵌套循环 L,设包含嵌套循环 L 所有数据相关距离向量的向量空间为 Ψ^L,Ψ^L 的基底为 $\bar{\beta}_1,...,\bar{\beta}_u$ ($u\geq 0$,如果 $u=0$,则表示 Ψ^L 为 0 空间),且 $\bar{\alpha}_1,...,\bar{\alpha}_v,\bar{\beta}_1,...,\bar{\beta}_u$ 为 Q^n 的一个基底.令 $B^L(k_1,...,k_v)=k_1\bar{\alpha}_1+...+k_v\bar{\alpha}_v+\Psi^L$ ($v\geq 1$),那么我们称 $\{B^L(k_1,...,k_v)\,|\,k_1,....,k_v\in Q\}$ 是嵌套循环 L 的迭代空间的一个线性计算划分,$B^L(k_1,...,k_v)$ 为计算划分函数,$k_1,...,k_v$ 为计算划分函数的自变量.

我们也称 $B^L(k_1,...,k_v)$ 为迭代划分块.一般地,容易证明在包含嵌套循环所有数据相关距离向量的向量空间保持不变的情形下,满足定义 1 条件的所有线性计算划分对迭代空间的划分结果都是相同的.

定义 2. 给定一个 n 重嵌套循环 L 迭代空间的线性计算划分 $\{B^L(k_1,...,k_v)=k_1\bar{\alpha}_1+...+k_v\bar{\alpha}_v+\Psi^L\,|\,k_1,...,$

$k_v \in Q\}$,那么我们称该线性计算划分的并行度为 v (或 $\dim(\Psi^{L^{\perp}})$, $n-\dim(\Psi^L)$).

与 Lim[4]在计算划分方面的工作相比,我们用向量的形式来表示计算划分,并以此为基础为后面求解数据划分做准备,最后还根据求得的数据划分来进一步对计算划分进行调整.虽然 Lim[4]在计算划分方面的工作具有更大的普遍性,但由于他只考虑了对迭代空间进行划分而未考虑对数据空间进行划分,所以其方法一般不适用于分布主存多处理机系统,而我们的方法由于同时考虑了对迭代空间和数据空间进行划分,所以我们的方法适用于分布主存多处理机系统.

2.2 数据空间的划分与融合

2.2.1 线性数据划分

定义 3. 给定一个 m 维的数组 X,一组 $m\times 1$ 维的向量 $\bar{\delta},\bar{\gamma}_1,...,\bar{\gamma}_v$,以及由 $m\times 1$ 维的向量组成的向量空间 Ω_X.令 $D_X(k_1,...,k_v)=\bar{\delta}+k_1\bar{\gamma}_1+...+k_v\bar{\gamma}_v+\Omega_X$ ($v\geq 1$),那么我们称 $\{D_X(k_1,...,k_v)|k_1,...,k_v\in Q\}$ 是数组 X 的数据空间的一个线性数据划分, $D_X(k_1,...,k_v)$ 为数据划分函数, $k_1,...,k_v$ 为数据划分函数的自变量.设 $\dim(\Omega_X)=u$,且 $\bar{\eta}_1,...,\bar{\eta}_u$ 为 Ω_X 的一个基底, $r=\dim(\bar{\gamma}_1,...,\bar{\gamma}_v,\bar{\eta}_1,...,\bar{\eta}_u)$,我们还称线性数据划分 $\{D_X(k_1,...,k_v)|k_1,...,k_v\in Q\}$ 的并行度为 $r-u$.

我们也称 $D_X(k_1,...,k_v)$ 为数据划分块.在求得了给定迭代空间的线性计算划分后,我们就能求出迭代空间中各个迭代划分块对于每个数组的每次不同引用所访问的数据空间.给定一个 n 重嵌套循环 L 以及其中被访问的数组 X 的某次引用 $A\bar{I}+\bar{o}$,设 $\left\{B^L(k_1,...,k_v)=k_1\bar{\alpha}_1+...+k_v\bar{\alpha}_v+\Psi^L\middle|k_1,...,k_v\in Q\right\}$ 是 L^n 的线性计算划分, $\bar{\beta}_1,...,\bar{\beta}_u$ 为 Ψ^L 的基底.对于数组 X 的引用 $A\bar{I}+\bar{o}$,迭代划分块 $B^L(k_1,...,k_v)$ 所访问的数组空间为 $AB^L(k_1,...,k_v)+\bar{o}=\bar{o}+k_1A\bar{\alpha}_1+...+k_vA\bar{\alpha}_v+span\left\{A\bar{\beta}_1,...,A\bar{\beta}_u\right\}$,若我们令 $D_X(k_1,...,k_v)=\bar{o}+AB^L(k_1,...,k_v)$,那么根据定义 3 可知, $\{D_X(k_1,...,k_v)|k_1,...,k_v\in Q\}$ 是数组 X 的数据空间的一个线性数据划分.综上所述,给定一个嵌套循环的一个线性计算划分以及在该嵌套循环中被访问的某个数组的某次引用,那么通过该线性计算划分,我们就能求出该数组的数据空间的一个线性数据划分.

2.2.2 线性数据划分的标准形式

定义 4. 给定一个线性数据划分 $\left\{D(k_1,...,k_v)=\bar{\delta}+k_1\bar{\gamma}_1+...+k_v\bar{\gamma}_v+\Omega\middle|k_1,...,k_v\in Q\right\}$ 和集合 $\{D'((F\bar{K})^T)=\bar{\delta}'+HF\bar{K}+\Omega'\,|\,\bar{K}\in Q^v\}$,其中 Ω 和 Ω' 是由 $m\times 1$ 维的向量构成的向量空间, $H=(\bar{\varphi}_1,...,\bar{\varphi}_l)$, F 为 $l\times v$ 维的矩阵, $\bar{K}=(k_1,...,k_v)^T$, $\bar{\delta},\bar{\delta}',\bar{\gamma}_1,...,\bar{\gamma}_v,\bar{\varphi}_1,...,\bar{\varphi}_l$ 为 $m\times 1$ 维的向量.

(1) 若 $\forall 1\leq j\leq v,\bar{\gamma}_j\in\Omega$,那么我们称 $\left\{D'((F\bar{K})^T)\middle|\bar{K}\in Q^v\right\}$ 为线性数据划分 $\{D(k_1,...,k_v)|k_1,...,k_v\in Q\}$ 的标准形式,其中 $\bar{\delta}'=\bar{\delta}$, $\Omega'=\Omega$, H 为 $m\times 1$ 维的零向量, F 为 $1\times v$ 维的零向量.这时显然有 $\forall k_1,...,k_v\in Q$, $D(k_1,...,k_v)=D'((F\bar{K})^T)$.

(2) 若 $\exists 1\leq j\leq v,\bar{\gamma}_j\notin\Omega$,设 $\bar{\eta}_1,...,\bar{\eta}_u$ 为 Ω' 的基底,如果 F 行满秩, $\bar{\varphi}_1,...,\bar{\varphi}_l,\bar{\eta}_1,...,\bar{\eta}_u$ 为线性无关组,并且 $\forall k_1,...,k_v\in Q$,有 $D(k_1,...,k_v)=D'((F\bar{K})^T)$,那么我们称 $\left\{D'((F\bar{K})^T)\middle|\bar{K}\in Q^v\right\}$ 为线性数据划分 $\{D(k_1,\ldots,k_v)|k_1,\ldots,k_v\in Q\}$ 的标准形式.

从定义 4 我们可看出,若 $\forall 1\leq j\leq v,\bar{\gamma}_j\in\Omega$,那么我们能直接求出线性数据划分唯一的一个标准形式,否则我们可以利用算法 1 来求解线性数据划分的标准形式.

算法 1. 求解线性数据划分的标准形式.

输入:线性数据划分 $\left\{D(k_1,...,k_v)=\bar{\delta}+k_1\bar{\gamma}_1+...+k_v\bar{\gamma}_v+\Omega\middle|k_1,...,k_v\in Q\right\}$,其中 $\exists 1\le j\le v,\bar{\gamma}_j\notin\Omega$.

输出:该线性数据划分的标准形式 $\left\{D'((F\bar{K})^T)=\bar{\delta}'+HF\bar{K}+\Omega'\middle|\bar{K}\in Q^v\right\}$.

(1) 设 $\bar{\eta}_1,...,\bar{\eta}_u$ 为 Ω 的一个基底,令 $r=\dim(\bar{\gamma}_1,...,\bar{\gamma}_v,\bar{\eta}_1,...,\bar{\eta}_u)$,求包含 $\bar{\eta}_1,...,\bar{\eta}_u$ 且能线性表示 $\bar{\gamma}_1,...,\bar{\gamma}_v$ 的个数最少的线性无关向量组(例如,向量组 $\bar{\gamma}_1,...,\bar{\gamma}_v,\bar{\eta}_1,...,\bar{\eta}_u$ 的一个包含 $\bar{\eta}_1,...,\bar{\eta}_u$ 的极大线性无关向量组),显然该线性无关向量组的个数为 r,设其为 $\bar{\varphi}_1,...,\bar{\varphi}_{r-u},\bar{\eta}_1,...,\bar{\eta}_u$(由于包含 $\bar{\eta}_1,...,\bar{\eta}_u$ 且能线性表示 $\bar{\gamma}_1,...,\bar{\gamma}_v$ 的个数最少的线性无关向量组不是唯一的,所以最后求得的线性数据划分的标准形式也可能不是唯一的);

(2) 设 $\bar{\gamma}_j=a_{j1}\bar{\varphi}_1+...+a_{j(r-u)}\bar{\varphi}_{r-u}+c_{j1}\bar{\eta}_1+...+c_{ju}\bar{\eta}_u$,$1\le j\le v$,将它们代入 $D(k_1,...,k_v)$ 中可得:

$$D(k_1,...,k_v)=\bar{\delta}+(a_{11}k_1+...+a_{v1}k_v)\bar{\varphi}_1+...+(a_{1(r-u)}k_1+...+a_{v(r-u)}k_v)\bar{\varphi}_{r-u}+span\left\{\bar{\eta}_1,...,\bar{\eta}_u,\bar{0}\right\}$$

(3) $\bar{\delta}'=\bar{\delta}$, $F=\begin{pmatrix}a_{11}&...&a_{v1}\\ \vdots&\vdots&\vdots\\ a_{1(r-u)}&...&a_{v(r-u)}\end{pmatrix}$, $\bar{K}=(k_1,...,k_v)^T$, $H=(\bar{\varphi}_1,...,\bar{\varphi}_{r-u})$, $\Omega'=\Omega$;

RETURN ($\{D'((F\bar{K})^T)=\bar{\delta}'+HF\bar{K}+\Omega'\mid\bar{K}\in Q^v\}$)

经过对算法 1 进行分析可知,其时间复杂度为 $O(muv+mvr)$,其中 m 为 Ω 中列向量的行数.易证算法 1 所返回的结果确实为线性数据划分的标准形式,且线性数据划分的任何一个标准形式中矩阵 F 的行秩都等于该线性数据划分的并行度.

2.2.3 数据复制

给定一个嵌套循环 L 迭代空间的线性计算划分以及在 L 中被访问的数组 X 的某次引用,那么对于此次引用,通过该线性计算划分,我们可求得数组 X 的数据空间的一个线性数据划分,并可进一步求得该线性数据划分的标准形式,设其为 $\left\{D'_x((F\bar{K})^T)\middle|\bar{K}\in Q^v\right\}$ ($\bar{K}=(k_1,...,k_v)^T$).矩阵 F 列向量的个数为 v,设其行秩为 f,如果 $f<v$,那么方程 $F\bar{d}=\bar{0}$ 有非零解,且解空间的维数为 $v-f$,所以对于数组 X 的此次引用,这时有多个迭代划分块访问同一个数据划分块的情况,为了使关于此次引用的访存都被本地化,我们需要将数据划分块进行复制.由于该线性计算划分的并行度为 v,为了充分并行,我们假设处理器空间的维数为 v,又由于该线性数据划分的并行度为 f,所以我们需要将数据划分块向其他 $v-f$ 维进行复制,我们可以看出所需复制的维数与 $F\bar{d}=\bar{0}$ 解空间的维数是相同的.

定义 5. 给定一个嵌套循环迭代空间的线性计算划分以及通过该线性计算划分所求得的其中被访问的某个数组的线性数据划分,设该线性计算划分和线性数据划分的并行度分别为 v 和 f ($f\le v$),那么我们称该线性数据划分相对于该线性计算划分的复制度为 $v-f$.

我们总是希望线性数据划分相对于线性计算划分的复制度尽量的小,因为这能使运行时所需复制的数据量尽量的少,并且如果被复制的数组有被写访问的,那么一般编译器还将对被复制的数据进行一致性维护[8],因此减小复制度,也会减少运行时的数据一致性维护开销.

2.2.4 数据空间的融合

给定一个 n 重嵌套循环 L 以及其中被访问的某个数组 X,令 $\{B^L(k_1,...,k_v)=k_1\bar{\alpha}_1+...+k_v\bar{\alpha}_v+\Psi^L\mid k_1,...,k_v\in Q\}$ 为 L^n 的线性计算划分.设数组 X 有 q 次不同的引用,它们分别为 $A_1\bar{I}+\bar{o}_1,...,A_q\bar{I}+\bar{o}_q$.对于引用 $A_j\bar{I}+\bar{o}_j$,令 $\left\{D_{X^j}(k_1,...,k_v)=\bar{\delta}_j+k_1\bar{\gamma}_{j1}+...+k_v\bar{\gamma}_{jv}+span\left\{\bar{\eta}_{j1},...,\bar{\eta}_{ju},\bar{0}\right\}\middle|k_1,...,k_v\in Q\right\}$ 为通过 L^n 的线性计算划分

所求得的数组 X 的线性数据划分,$1 \le j \le q$.如果 $q>1$,那么数组 X 有多个不同的引用,对于每次不同的引用,我们都可以求出其对应的线性数据划分,但由于同一数组的数据空间的数据划分是唯一的(因为本文所考虑的是静态数据划分,而未考虑数据重分布的情况),所以我们需要将这多个线性数据划分融合为一个统一的线性数据划分.设融合后统一的线性数据划分为 $\{D_X(k_1,...,k_v)|k_1,...,k_v \in Q\}$,那么 $\{D_X(k_1,...,k_v)|k_1,...,k_v \in Q\}$ 应满足: $\forall k_1,...,k_v \in Q, 1 \le j \le q$,都有 $D_{X^j}(k_1,...,k_v) \subseteq D_X(k_1,...,k_v)$.该条件即是要求同一个迭代划分块所访问的同一数组的所有数据空间都被融合成一个统一的数据空间.满足上述条件的线性数据划分可能会有许多,我们希望求出其中并行度最大的线性数据划分,因为在线性计算划分的并行度一定的情况下,这能使线性数据划分相对于线性计算划分的复制度最小.算法 2 给出了求解满足条件且具有最大并行度的融合后统一的线性数据划分的算法.

算法 2. 求解多个线性数据划分融合后统一的线性数据划分.

输入:数组 X 的 q 个线性数据划分 $\{D_{X^j}(k_1,...,k_v)|k_1,...,k_v \in Q\}, 1 \le j \le q$.

输出:数组 X 的 q 个线性数据划分融合后统一的线性数据划分 $\{D_X(k_1,...,k_v)|k_1,...,k_v \in Q\}$ $\theta = \{\bar{0}\}$.

DO $j=1,q$
  DO $f=1,u$
    $\theta = \theta \cup \{\bar{\eta}_{jf}\}$
  ENDDO
ENDDO
DO $j=2,q$
  DO $p=1,v$
    $\theta = \theta \cup \{\bar{\gamma}_{Jp} - \bar{\gamma}_{1p}\}$
  ENDDO
ENDDO
DO $j=2,q$
    $\theta = \theta \cup \{\bar{\delta}_J - \bar{\delta}_1\}$
ENDDO
$\Omega_X = span\{\theta\}$;
$D_X(k_1,...,k_v) = \bar{\delta}_1 + k_1\bar{\gamma}_{11} + ... + k_v\bar{\gamma}_{1v} + \Omega_X$;
Return($\{D_X(k_1,...,k_v) \mid k_1,...,k_v \in Q\}$)

经过对算法 2 进行分析可知其时间复杂度为 $O(mqu+mqv)$,其中 m 为 Ω_X 中列向量的行数.我们能够证明算法 2 所求得的线性数据划分满足前面我们对融合后统一的线性数据划分所要求的条件且具有最大的并行度,但限于篇幅,我们将省略该证明.

2.2.5 计算与数据到虚拟处理器空间的映射

给定一个 n 重嵌套循环 L 迭代空间的线性计算划分 $\{B^L(k_1,...,k_v)|k_1,...,k_v \in Q\}$,以及在 L 中被访问的 l 个数组 $X_1,...,X_l$ 经过算法 2 融合后统一的线性数据划分的标准形式 $\{D_{X_1}((F_1\bar{K})^T)|\bar{K} \in Q^v\},...,\{D_{X_l}((F_l\bar{K})^T)|\bar{K} \in Q^v\}$,其中 $\bar{K}=(k_1,...,k_v)^T$,F_j 是 $r_j \times v$ 维的矩阵,$1 \le j \le l$.假设虚拟处理器空间为 Q^v,其中 v 为虚拟处理器空间的维数.下面我们定义两个映射: $IP: \Delta \to Q^v$ 为迭代划分块到虚拟处理器空间坐标的映射,Δ为由迭代划分块组成的集合; $DP_{X_j}: \Theta \to Q^v$ 为数组 X_j 的数据划分块到虚拟处理器空间坐标的映射,Θ为由数据划分块组成的集合.如果 F_j 为零矩阵,那么数组 X_j 的线性数据划分的并行度为 0,为了使对该数组的所有访问都本地化,数组 X_j 应被复制到每个处理器上,由于它不会影响迭代划分块和其他数组的数据划分块到虚拟处

理器空间映射的确定,所以在下面的讨论中,我们都假设 $F_j(1\le j\le l)$是行满秩的矩阵.下面我们将分两种情形来确定 IP 和 $DP_{X_1},...,DP_{X_l}$.

(1) $\forall 1\le j\le l$,有 $r_j=v$,这时 $F_1,...,F_l$ 为非奇异方阵.

我们令 $IP(B^L(k_1,...,k_v))=\bar{K}$, $DP_{X_j}(D_{X_j}((F_j\bar{K})^T))=\bar{K}$ $(1\le j\le l)$.显然上述 IP 和 $DP_{X_1},...,DP_{X_l}$ 能使对数组 $X_1,...,X_l$ 的所有引用的访问都本地化,并且上述映射使得各数组的数据划分块不需要复制,因此其复制度为 0.

(2) $\exists 1\le j\le l, r_j<v$,这时,$F_1,...,F_l$ 不全为方阵.

不妨设 $F_1,...,F_q$ 为方阵$(0\le q<l)$, $F_{q+1},...,F_l$ 的行数少于列数.由于 $F_{q+1},...,F_l$ 的行数少于列数,所以存在多个迭代划分块访问相同的数据划分块的情况,所以为了使对数组 $X_{q+1},...,X_l$ 的访问本地化,数据划分块必须复制.由于一般编译器(例如 HPF)要求数据的复制是按某一维或某几维进行复制,并且可能有多个数组的数据划分块需要进行复制,所以我们所确定的迭代划分块到虚拟处理器空间坐标的映射必须能使所有这些数组的数据划分块都能按维进行复制.假设 $F_{q+1},...,F_l$ 的前 p 行对应相同,但第 $p+1$ 行不全相同.如果 $p>0$,那么设 T 为由 F_{q+1} 的前 p 行按原来的顺序所构成的矩阵,并给 T 的底部添加上 $v-p$ 行以使 T 变为非奇异方阵,令其为 T'.

我们令 $IP(B^L(k_1,...,k_v))=T'\bar{K}$.对 $\forall 1\le j\le q$,我们令 $DP_{X_j}(D_{X_j}((F_j\bar{K})^T))=T'\bar{K}$.对 $\forall q+1\le j\le l$,我们令 DP_{X_j} $(D_{X_j}((F_j\bar{K})^T))=\bar{K}'$,其中 $\bar{K}'$ 是 $v\times 1$ 维的列向量,且由它的前 p 个元素按原来的顺序所构成的向量为 $T\bar{K}$,而后 $v-p$ 个元素为任意有理数,从该映射我们可以看出,数据划分块要在最后 $v-p$ 维中进行复制.易知,上述 IP 和 $DP_{X_1},...,DP_{X_l}$ 能使对数组 $X_1,...,X_l$ 的所有引用的访问都本地化,且使得数组 $X_1,...,X_q$ 的复制度为 0 而数组 $X_{q+1},...,X_l$ 的复制度为 $v-p$.为了使 $X_{q+1},...,X_l$ 的复制度尽可能地小, p 应尽可能地大.由于 $X_{q+1},...,X_l$ 的线性数据划分的标准形式可能不是唯一的,因此我们可以选择其中某组标准形式以使得 p 最大.限于篇幅,求解使 p 最大的一组标准形式的算法将不在这里给出.

在确定了 IP 和 $DP_{X_1},...,DP_{X_l}$ 后,我们还需进一步求出 IP 和 $DP_{X_1},...,DP_{X_l}$ 的具体表达形式,由于篇幅所限,我们将省略 IP 和 $DP_{X_1},...,DP_{X_l}$ 的具体表达形式的求解方法.

3 全局计算与数据划分

3.1 多个嵌套循环的线性数据划分融合

给定多个嵌套循环及其所访问的数组,我们可以做出代表嵌套循环及其所访问数组间关系的双向关联图 $G_s=\langle V_c,V_d,E\rangle$ [11],其中,V_c 是由嵌套循环所构成的顶点集合,V_d 是由数组所构成的顶点集合.E 是无向边的集合,如果某个嵌套循环访问了某个数组,那么就有一条连接该嵌套循环和该数组的无向边.根据图的连通性,我们可以将 G_s 划分为若干个子图,即任意一个子图内部是连通的,但各子图间没有边相连.这样,我们就可以根据各子图所包含的嵌套循环将原来的多个嵌套循环划分成若干组,每一组嵌套循环都由其中某个子图所包含的嵌套循环所组成.由于各组嵌套循环所访问的数组互不相同,所以我们可以完全独立地分别考虑每一组中嵌套循环的计算划分与数据划分方法.假设其中某组嵌套循环为 $L_1,...,L_f$,不失一般性,下面我们就考虑该组嵌套循环的计算与数据划分方法.

假设嵌套循环 $L_1,...,L_f$ 整个所访问的数组为 $X_1,...,X_l$;对 $\forall 1\le j\le f$,嵌套循环 L_j 整个所访问的数组为 $X_{h_1^j},...,X_{h_{q_j}^j}$, 其中 $1\le q_j,h_1^j,...,h_{q_j}^j\le l$, 且 $\forall 1\le u,w\le q_j,u\ne w$,有 $h_u^j\ne h_w^j$;对 $\forall 1\le j\le l$,数组 X_j 仅在嵌套循环 $L_{p_1^j},...,L_{p_{t_j}^j}$ 中被访问,其中 $1\le t_j,p_1^j,...,p_{t_j}^j\le f$,且 $\forall 1\le u,w\le t_j,u\ne w$,有 $p_u^j\ne p_w^j$.令 $\{B^{L_j}(k_1^j,...,k_{v_j}^j)=k_1^j\bar{\alpha}_1^j+...+k_{v_j}^j\bar{\alpha}_{v_j}^j+\Psi^{L_j}\mid k_1^j,...,k_{v_j}^j\in Q\}$ 为嵌套循环 L_j 迭代空间的线性计算划分, $\{D_{X_j}^{L_{p_1^j}}(k_1^{p_1^j},...,k_{v_{p_1^j}}^{p_1^j})=\bar{\delta}_j^{p_1^j}+k_1^{p_1^j}\bar{\gamma}_{j1}^{p_1^j}+...+$

$k_{v_{p_1^j}^j}^{p_1^j}\ \bar{\gamma}_{jv_{p_1^j}^j}^{p_1^j}+\Omega_j^{p_1^j}\mid k_1^{p_1^j},...,k_{v_{p_1^j}^j}^{p_1^j}\in Q\},\ldots,\left\{D_{X_j}^{L_{p_{t_j}^j}}(k_1^{p_{t_j}^j},...,k_{v_{p_{t_j}^j}^j}^{p_{t_j}^j})=\bar{\delta}_j^{p_{t_j}^j}+k_1^{p_{t_j}^j}\bar{\gamma}_{j1}^{p_{t_j}^j}+...+k_{v_{p_{t_j}^j}^j}^{p_{t_j}^j}\bar{\gamma}_{jv_{p_{t_j}^j}^j}^{p_{t_j}^j}+\Omega_j^{p_{t_j}^j}\middle|k_1^{p_{t_j}^j},...,k_{v_{p_{t_j}^j}^j}^{p_{t_j}^j}\in Q\right\}$ 分别为数组 X_j 在嵌套循环 $L_{p_1^j},...,L_{p_{t_j}^j}$ 中融合后统一的线性数据划分.由于数组 X_j 的线性数据划分是唯一的,所以我们需要将上述 X_j 在嵌套循环 $L_{p_1^j},...,L_{p_{t_j}^j}$ 中的线性数据划分再次融合成一个统一的线性数据划分.算法 2 所给出的线性数据划分的融合方法要求所有线性数据划分的数据划分函数中自变量是对应相同的,但 $\left\{D_{X_j}^{L_{p_1^j}}(k_1^{p_1^j},...,k_{v_{p_1^j}^j}^{p_1^j})\middle|k_1^{p_1^j},...,k_{v_{p_1^j}^j}^{p_1^j}\in Q\right\},\ldots,\left\{D_{X_j}^{L_{p_{t_j}^j}}(k_1^{p_{t_j}^j},...,k_{v_{p_{t_j}^j}^j}^{p_{t_j}^j})\middle|k_1^{p_{t_j}^j},...,k_{v_{p_{t_j}^j}^j}^{p_{t_j}^j}\in Q\right\}$ 中各自变量相互独立且个数可能不同,因此为了应用算法 2,我们需要将属于不同嵌套循环的线性数据划分的数据划分函数中的自变量对准.我们可以通过将不同嵌套循环迭代空间线性计算划分的计算划分函数中的自变量对准来达到使不同嵌套循环线性数据划分的数据划分函数中的自变量对准的目的.

令 $v=\min\{v_1,...,v_f\}$, $k_1,...,k_v$ 为对准后的自变量,对 $\forall 1\le j\le f$,我们为 L_j 定义自变量对准映射 $\sigma^j:O\to\Gamma^j$,其中 $O=\{1,...,v\}$, $\Gamma^j=\{1,...,v_j\}$,并且 σ^j 是单射.我们令 $M^j=\left\{\bar{\alpha}_1^j,...,\bar{\alpha}_{v_j}^j\right\}-\left\{\bar{\alpha}_{\sigma^j(1)}^j,...,\bar{\alpha}_{\sigma^j(v)}^j\right\}$, $\Psi^{L_j}=$, $span\{M^j\cup\Psi^{L_j}\}$ 易证 $\left\{B_{\sigma_j}^{L_j}(k_1,...,k_v)=k_1\bar{\alpha}_{\sigma^j(1)}^j+...+k_v\bar{\alpha}_{\sigma^j(v)}^j+\psi^{L_j}\middle|k_1,...,k_v\in Q\right\}$ 是嵌套循环 L_j 迭代空间的一个线性计算划分.假设 X_u 是在嵌套循环 L_j 中被访问的数组,令 $N_u^j=\left\{\bar{\gamma}_{u1}^j,...,\bar{\gamma}_{uv_j}^j\right\}-\left\{\bar{\gamma}_{u\sigma^j(1)}^j,...,\bar{\gamma}_{u\sigma^j(v)}^j\right\}$, $\Phi_u^j=span\{N_u^j\cup\Omega_u^j\}$,那么易证对应于嵌套循环 L_j 的线性计算划分 $\{B_{\sigma_j}^{L_j}(k_1,...,k_v)\mid k_1,...,k_v\in Q\}$, $\left\{D_{X_u\sigma_j}^{L_j}(k_1,...,k_v)=\bar{\delta}_u^j+k_1\bar{\gamma}_{u\sigma^j(1)}^j+...+k_v\bar{\gamma}_{u\sigma^j(v)}^j+\Phi_u^j\middle|k_1,...,k_v\in Q\right\}$ 是数组 X_u 在嵌套循环 L_j 中融合后统一的线性数据划分.由于 X_u 出现在嵌套循环 $L_{p_1^u},...,L_{p_{t_u}^u}$ 中,所以数组 X_u 在嵌套循环 $L_{p_1^u},...,L_{p_{t_u}^u}$ 中经融合后统一的线性数据划分分别为 $\left\{D_{X_u\sigma_{p_1^u}}^{L_{p_1^u}}(k_1,...,k_v)\middle|k_1,...,k_v\in Q\right\},...,\left\{D_{X_u\sigma_{p_{t_u}^u}}^{L_{p_{t_u}^u}}(k_1,...,k_v)\middle|k_1,...,k_v\in Q\right\}$,由于这时各线性数据划分的数据划分函数中自变量都对应相同,所以我们可以应用算法 2 将数组 X_u 在嵌套循环 $L_{p_1^u},...,L_{p_{t_u}^u}$ 中的线性数据划分再次融合成一个统一的线性数据划分.综上所述,在为嵌套循环 $L_1,...,L_f$ 确定了自变量对准映射 $\sigma_1,...,\sigma_f$ 后,我们就能分别求出数组 $X_1,...,X_l$ 在多个嵌套循环中再次融合后统一的线性数据划分,然后再假设每个嵌套循环都包含 $X_1,...,X_l$,并用第 2.2.5 节所介绍的方法确定嵌套循环迭代空间的迭代划分块到虚拟处理器空间的映射以及数组 $X_1,...,X_l$ 的数据划分块到虚拟处理器空间的映射.

3.2 确定全局计算与数据划分

选择不同的自变量对准映射,最后所得到的数组的复制度可能不同,我们总希望复制度越小越好.一般地,确定多个嵌套循环的计算与数据划分是 NP 完全问题[12,13],我们可以按以下两种方法来确定多个嵌套循环的线性计算划分和线性数据划分:

第 1 种方法是全搜索法.即遍历所有可能的自变量对准映射的组合,并选择使得所有数组的复制度之和最小的一组自变量对准映射.这种方法能求得最优解,但可能会因为自变量对准映射的组合量过大而耗时较多.

第 2 种方法是贪婪启发式搜索法.给定一个 m 维数组 X 的一个线性数据划分 $\{D_X^L(k_1,...,k_v)=\bar{\delta}+k_1\bar{\gamma}_1+...+k_v\bar{\gamma}_v+\Omega_X\mid k_1,...,k_v\in Q\}$,一般来说,如果 $\dim(\Omega_X)$ 的值越小,那么该线性数据划分的并行度就可能越大,从而数组 X 的复制度就可能越小.因此,在对多个嵌套循环中的数组的线性数据划分进行融合时,我们尽可能使得在融合的各个步骤中, $\dim(\Omega_X)$ 尽量地小,这样数组的线性数据划分的并行度就可能尽可能地大,从而

有可能使得最终融合后数组的线性数据划分的并行度最大,也即是使得数组的复制度最小,所以我们按照上述原则进行启发式搜索以确定自变量对准映射.为了讨论方便,我们定义两个集合的连接操作 $\triangleright\triangleleft$,设 M_1 为由 $1\times m_1$ 维向量组成的集合,M_2 为由 $1\times m_2$ 维向量组成的集合,$M_1\triangleright\triangleleft M_2$ 是由 $1\times(m_1+m_2-1)$ 维向量组成的集合,且 $M_1\triangleright\triangleleft M_2$ 按如下的规则构成:任取 $(u_1,...,u_{m_1-1},u_{m_1})\in M_1$,$(w_1,w_2,...,w_{m_2})\in M_2$,若 $u_{m_1}=w_1$,那么就添加 $(u_1,...,u_{m_1-1},u_{m_1},w_2,...,w_{m_2})$ 到 $M_1\triangleright\triangleleft M_2$ 中,我们称 $M_1\triangleright\triangleleft M_2$ 是由 M_1 连接 M_2 所构成的集合,例如 $\{(1,2),(3,4)\}\triangleright\triangleleft\{(2,3),(4,1)\}=\{(1,3),(3,1)\}$,$\{(1,1)\}\triangleright\triangleleft\{(2,2)\}=\varnothing$.特别地,我们令 $\varnothing\triangleright\triangleleft M_2=M_2$,$M_1\triangleright\triangleleft\varnothing=M_1$.求解多个嵌套循环的自变量对准映射的贪婪启发式搜索算法见算法 3.

算法 3. 求解多个嵌套循环的自变量对准映射的贪婪启发式搜索算法.

假设嵌套循环 $L_1,...,L_f$ 整个所访问的数组为 $X_1,...,X_l$,而对 $\forall 1\le j\le f$,嵌套循环 L_j 所访问的数组为 $X_{h_1^j},...,X_{h_{q_j}^j}$,其中 $1\le q_j,h_1^j,...,h_{q_j}^j\le l$,且 $\forall 1\le u,w\le q_j,u\ne w$,有 $h_u^j\ne h_w^j$;对 $\forall 1\le j\le l$,数组 X_j 出现在嵌套循环 $L_{p_1^j},...,L_{p_{t_j}^j}$ 中,其中 $1\le t_j,p_1^j,...,p_{t_j}^j\le f$,且 $\forall 1\le u,w\le t_j,u\ne w$,有 $p_u^j\ne p_w^j$.

输入:嵌套循环 L_j 迭代空间的线性计算划分 $\left\{B^{L_j}(k_1^j,...,k_{v_j}^j)=k_1^j\,\bar{\alpha_1^j}+...+k_{v_j}^j\,\bar{\alpha_{v_j}^j}+\Psi^{L_j}\,\middle|\,k_1^j,...,k_{v_j}^j\in Q\right\}$,嵌套循环 L_j 中数组 $X_{h_u^j}$ 融合后统一的线性数据划分 $\{D_{X_{h_u^j}}^{L_j}(k_1^j,...,k_{v_j}^j)=\bar{\delta_{h_u^j}^j}+k_1^j\,\bar{\gamma_{h_u^j1}^j}+...+k_{v_j}^j\,\bar{\gamma_{h_u^jv_j}^j}+\Omega_{h_u^j}^j\mid k_1^j,...,k_{v_j}^j\in Q\}$,$1\le j\le f$,$1\le u\le q_j$.

输出:嵌套循环 $L_1,...,L_f$ 的自变量对准映射 $\sigma_1,...,\sigma_f$.

$$\sigma_1=...=\sigma_f=\varnothing$$

/*第 1 部分.确定每个嵌套循环的待对准映射变量集合*/

$$v=\min\{v_1,...,v_f\}$$

DO $j=1,f$

$\Gamma^j=\{1,...,v_j\}$

DO $z=1,v_j-v$

$\Gamma=\Gamma^j$;min*sum*=+∞

DO WHILE $\Gamma\ne\varnothing$

任取 $w\in\Gamma$;*sum*=0

DO $u=1,q_j$

IF $\bar{\gamma_{h_u^jw}^j}\notin\Omega_{h_u^j}^j$ THEN *sum*=*sum*+1

ENDDO

IF *sum*<min*sum* THEN min*sum*=*sum*;*index*=*w*

$\Gamma=\Gamma-\{w\}$

ENDDO

$\Gamma^j=\Gamma^j-\{index\}$

DO $u=1,q_j$

$$\Omega_{h_u^j}^j=span\left\{\Omega_{h_u^j}^j\cup\left\{\bar{\gamma_{h_u^j(index)}^j}\right\}\right\}$$

ENDDO

ENDDO

ENDDO

$\Omega_1=...=\Omega_l=\varnothing$

DO $j=1,l$

$\Omega_j = span\left\{\Omega_j^{p_1^j} \cup ... \cup \Omega_j^{p_{t_j}^j}\right\}$

IF $t_j \geq 2$ THEN $\Omega_j = span\left\{\Omega_j \cup (\delta_j^{\bar{p}_2^j} - \delta_j^{\bar{p}_1^j}) \cup ... \cup (\delta_j^{\bar{p}_{t_j}^j} - \delta_j^{\bar{p}_1^j})\right\}$

ENDDO

/*第 2 部分.求解嵌套循环的自变量对准映射*/

求 $L_1,...,L_f$ 中,访问相同数组个数最多的两个嵌套循环(可根据双向关联图求得),设为 L_x, L_y

$M = \{L_x\}$; $N = \{L_1,...,L_f\} - M$; $\sigma = \sigma^{temp} = \varnothing$; $vect = (L_x)$

DO $z = 1, f-1$

不妨设 L_x, L_y 所访问的相同数组为 $X_1,...,X_s$

$\Gamma_1 = \Gamma^x$; $\Gamma_2 = \Gamma^y$

DO WHILE $\Gamma_1 \neq \varnothing$

任取 $w \in \Gamma_1$;min*sum*=+∞; $\Gamma_{temp} = \Gamma_2$

DO WHILE $\Gamma_2 \neq \varnothing$

任取 $u \in \Gamma_2$;*sum*=0

DO $e = 1, s$

IF $\gamma_{ew}^{\bar{x}} - \gamma_{eu}^{\bar{y}} \notin \Omega_e$ THEN *sum*=*sum*+1

ENDDO

IF *sum*<min*sum* THEN min*sum*=*sum*;*index*=*u*

$\Gamma_2 = \Gamma_2 - \{u\}$

ENDDO

$\sigma^{temp} = \sigma^{temp} \cup \{(w, index)\}$

DO $e = 1, s$

$\Omega_e = span\left\{\Omega_e \cup \left\{\gamma_{ew}^{\bar{x}} - \gamma_{e(index)}^{\bar{y}}\right\}\right\}$

ENDDO

$\Gamma_1 = \Gamma_1 - \{w\}$; $\Gamma_2 = \Gamma_{temp} - \{index\}$

ENDDO

设 L_x 是 *vect* 的第 *g* 个元素,若 L_x 不是 *vect* 的最后一个元素,则将 L_x 与最后一个元素互换位置,同时将 σ中所有向量的第 *g* 个元素与最后一个元素互换位置;

$\sigma = \sigma \triangleright\triangleleft \sigma^{temp}$;将 L_y 添加到 *vect* 的末尾; $M = M \cup \{L_y\}$; $N = N - \{L_y\}$;

若 $N \neq \varnothing$,任取 $L_{y'} \in N$,求 *M* 中与 L_y 所访问的相同数组个数最多的嵌套循环,设为 $L_{x'}$(可根据双向关联图求得),令 $L_x = L_{x'}$, $L_y = L_{y'}$.

ENDDO

$\sigma^{temp} = \sigma$

DO $z' = 1, v$

任取 $y \in \sigma^{temp}$

DO $z = 1, f$

设 L_z 为 *vect* 的第 *x* 个元素

$\sigma_z = \sigma_z \cup \{(z', y') | y' 为 y 的第 x 个元素\}$

ENDDO

$\sigma^{temp} = \sigma^{temp} - \{y\}$

ENDDO

RETURN($\sigma_1,...,\sigma_f$)

经过对算法 3 进行分析可知其时间复杂度为 $O(f\upsilon^2qm(m+\upsilon)+lmt(m+\upsilon)+f^2q^2)$,其中 $\upsilon=\max\{v_1,...,v_f\}$, $q=\max\{q_1,...,q_f\}$, $t=\max\{t_1,...,t_l\}$, m 为 $\Omega_1,...,\Omega_l$ 中列向量行数的最大值.从算法 3 可以看出,虽然求解自变量对准映射的每个步骤都在前一个步骤所求得的结果基础上,朝使线性数据划分的并行度尽可能大的方向前进,但最终的线性数据划分的并行度却不一定是最大的,所以算法 3 不一定能求出最优解,但与全搜索方法相比,其耗时较少.

3.3 举 例

考虑对图 1 中嵌套循环 L_1, L_2 进行计算与数据划分,图 2 给出了代表嵌套循环 L_1, L_2 及其所访问数组间关系的双向关联图.由于该图是连通的,所以我们应该同时考虑嵌套循环 L_1, L_2 的计算与数据划分.

```
REAL X(N,N,N,N),Y(N,N,N,N),Z(N,N,N)
DO i1=1,10
 DO i2=1,10
   DO i3=1,10
     DO i4=1,10
       X(i1+1,i2+1,i3+1,i4)=Y(i1+2i4,i2+1,i1+i3+1,i2+i3+i4+1)+Z(i3,i1+i4,i2+2i4)
       X(2i1+i2+i3−1,i1+2i2+i3−1,i1+i2+2i3−1,i4)=X(i1+1,i2+1,i3+1,i4)+
                                                 Y(i1+i2+i3+i4,i1+i2+i4,2i1+i2+2i3,i1+2i2+2i3+i4)
     ENDDO
   ENDDO
 ENDDO
ENDDO
```

(a) Loop nest L_1

(a) 嵌套循环 L_1

```
DO i1=1,10
 DO i2=1,4
   DO i3=1,10
     DO i4=1,10
       Y(i1+2i3+i4+1,i3+i4−1,i1+i3+i4,i1−i2+2i3+2i4)=X(2i1+i2+i3+i4+2,2i1+i3+2i4+2,2i1+i3+i4+2,i3)
     ENDDO
   ENDDO
 ENDDO
ENDDO
```

(b) Loop nest L_2

(b) 嵌套循环 L_2

Fig.1 Code fragment example

图 1 代码段示例

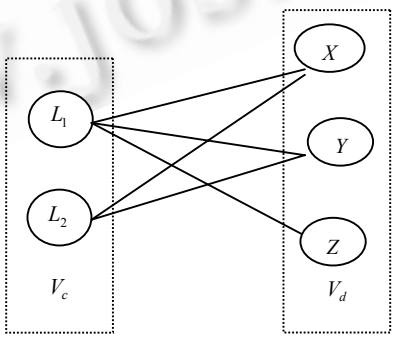


Fig.2 The bipartite interference graph of the loop nests L_1 and L_2 with accessed arrays in Fig.1

图 2 图 1 中嵌套循环 L_1, L_2 及其所访问数组间关系的双向关联图

通过对嵌套循环 L_1 进行相关性分析,我们可以求得包含嵌套循环 L_1 所有数据相关距离向量的向量空间

$\Psi^{L_1}=span\{(1\ \ 1\ \ 1\ \ 0)^T\}$,根据定义 1 可知,$\left\{B^{L_1}(k_1^1,k_2^1,k_3^1)=k_1^1\begin{pmatrix}1\\0\\0\\0\end{pmatrix}+k_2^1\begin{pmatrix}0\\1\\0\\0\end{pmatrix}+k_3^1\begin{pmatrix}0\\0\\0\\1\end{pmatrix}+span\left\{\begin{pmatrix}1\\1\\1\\0\end{pmatrix}\right\}\middle|k_1^1,k_2^1,k_3^1\in Q\right\}$是嵌套循环 L_1 的迭代空间的一个线性计算划分,并且根据定义 2 可知其并行度为 3.然后根据第 2.2.1 节所介绍的方法,利用该线性计算划分和算法 2 可求得:在嵌套循环 L_1 中被访问的数组 X,Y 和 Z 经融合后统一的线性数据划分分别为 $\left\{D_X^{L_1}(k_1^1,k_2^1,k_3^1)=\begin{pmatrix}1\\1\\1\\0\end{pmatrix}+k_1^1\begin{pmatrix}1\\0\\0\\0\end{pmatrix}+k_2^1\begin{pmatrix}0\\1\\0\\0\end{pmatrix}+k_3^1\begin{pmatrix}0\\0\\0\\1\end{pmatrix}+span\left\{\begin{pmatrix}1\\1\\1\\0\end{pmatrix}\right\}\middle|k_1^1,k_2^1,k_3^1\in Q\right\}$,$\left\{D_Y^{L_1}(k_1^1,k_2^1,k_3^1)=\begin{pmatrix}0\\1\\1\\1\end{pmatrix}+k_1^1\begin{pmatrix}1\\0\\1\\0\end{pmatrix}+k_2^1\begin{pmatrix}0\\1\\0\\1\end{pmatrix}+k_3^1\begin{pmatrix}2\\0\\0\\1\end{pmatrix}+span\left\{\begin{pmatrix}1\\1\\2\\2\end{pmatrix},\begin{pmatrix}3\\2\\5\\5\end{pmatrix}\right\}\middle|k_1^1,k_2^1,k_3^1\in Q\right\}$和 $\left\{D_Z^{L_1}(k_1^1,k_2^1,k_3^1)=k_1^1\begin{pmatrix}0\\1\\0\end{pmatrix}+k_2^1\begin{pmatrix}0\\0\\1\end{pmatrix}+k_3^1\begin{pmatrix}0\\1\\2\end{pmatrix}+span\left\{\begin{pmatrix}1\\1\\1\end{pmatrix}\right\}\middle|k_1^1,k_2^1,k_3^1\in Q\right\}$.

同理,我们可以求得嵌套循环 L_2 迭代空间的线性计算划分为 $\left\{B^{L_2}(k_1^2,k_2^2,k_3^2,k_4^2)=k_1^2\begin{pmatrix}1\\0\\0\\0\end{pmatrix}+k_2^2\begin{pmatrix}0\\1\\0\\0\end{pmatrix}+k_3^2\begin{pmatrix}0\\0\\1\\0\end{pmatrix}+k_4^2\begin{pmatrix}0\\0\\0\\1\end{pmatrix}+span\left\{\begin{pmatrix}0\\0\\0\\0\end{pmatrix}\right\}\middle|k_1^2,k_2^2,k_3^2,k_4^2\in Q\right\}$,在嵌套循环 L_2 中被访问的数组 X 和 Y 经融合后统一的线性数据划分分别为 $\left\{D_X^{L_2}(k_1^2,k_2^2,k_3^2,k_4^2)=\begin{pmatrix}2\\2\\2\\0\end{pmatrix}+k_1^2\begin{pmatrix}2\\2\\2\\0\end{pmatrix}+k_2^2\begin{pmatrix}1\\0\\0\\0\end{pmatrix}+k_3^2\begin{pmatrix}1\\1\\1\\1\end{pmatrix}+k_4^2\begin{pmatrix}1\\2\\1\\0\end{pmatrix}+span\left\{\begin{pmatrix}0\\0\\0\\0\end{pmatrix}\right\}\middle|k_1^2,k_2^2,k_3^2,k_4^2\in Q\right\}$和 $\left\{D_Y^{L_2}(k_1^2,k_2^2,k_3^2,k_4^2)=\begin{pmatrix}1\\-1\\0\\0\end{pmatrix}+k_1^2\begin{pmatrix}1\\0\\1\\1\end{pmatrix}+k_2^2\begin{pmatrix}0\\0\\0\\-1\end{pmatrix}+k_3^2\begin{pmatrix}2\\1\\1\\2\end{pmatrix}+k_4^2\begin{pmatrix}1\\1\\1\\2\end{pmatrix}+span\left\{\begin{pmatrix}0\\0\\0\\0\end{pmatrix}\right\}\middle|k_1^2,k_2^2,k_3^2,k_4^2\in Q\right\}$.由于数组 X 和 Y 同时在嵌套循环 L_1,L_2 中出现,所以我们需要再次对它们进行融合,这需要首先确定对准映射.

根据算法 3,我们可求得嵌套循环 L_1 的对准映射 σ_1 为 $\{(1,1),(2,2),(3,3)\}$,嵌套循环 L_2 的对准映射 σ_2 为 $\{(1,2),(2,4),(3,3)\}$.设 k_1,k_2,k_3 为对准后的自变量,然后我们就可以利用算法 2 和算法 1 求出再次融合后数组 X 和 Y 统一的线性数据划分的标准形式分别为 $\left\{D_x'((F_X\ \bar{K})^T)=\begin{pmatrix}1\\1\\1\\0\end{pmatrix}+\begin{pmatrix}1&0&0\\0&1&0\\0&0&0\\0&0&1\end{pmatrix}\begin{pmatrix}1&0&0\\0&1&0\\0&0&1\end{pmatrix}\begin{pmatrix}k_1\\k_2\\k_3\end{pmatrix}+span\left\{\begin{pmatrix}1\\1\\1\\0\end{pmatrix}\right\}\middle|\bar{K}\in Q^3\right\}$

和 $\left\{D'_y((F_Y\ \bar{K})^T)=\begin{pmatrix}0\\1\\1\\1\end{pmatrix}+\begin{pmatrix}1&0\\0&1\\1&0\\0&1\end{pmatrix}\begin{pmatrix}1&0&1\\0&1&2\end{pmatrix}\begin{pmatrix}k_1\\k_2\\k_3\end{pmatrix}+span\left\{\begin{pmatrix}1\\1\\2\\2\end{pmatrix},\begin{pmatrix}3\\2\\5\\5\end{pmatrix}\right\}\middle|\ \bar{K}\in Q^3\right\}$.数组 Z 不需要融合,其标准形式为 $\left\{D'_z((F_Z\ \bar{K})^T)=\begin{pmatrix}0&0\\0&1\\1&2\end{pmatrix}\begin{pmatrix}-2&1&0\\1&0&1\end{pmatrix}\begin{pmatrix}k_1\\k_2\\k_3\end{pmatrix}+span\left\{\begin{pmatrix}1\\1\\1\end{pmatrix}\right\}\middle|\ \bar{K}\in Q^3\right\}$.

进一步地,我们需要求出迭代划分块与数据划分块到虚拟处理器的映射,按第 2.2.5 节所介绍的方法,我们可求出 $IP^{L_1}(B^{L_1}_{\sigma_1}(k_1,k_2,k_3))=T'\bar{K}$, $IP^{L_2}(B^{L_2}_{\sigma_2}(k_1,k_2,k_3))=T'\bar{K}$, $DP_X(D'_X((F_X\ \bar{K})^T))=T'\bar{K}$, $DP_Y(D'_Y((F_Y\ \bar{K})^T))=\bar{K}'$, $DP_Z\ (D'_Z((F_Z\ \bar{K})^T))=\bar{K}'$,其中 $\bar{K}=(k_1,k_2,k_3)^T$, $T'=\begin{pmatrix}1&0&1\\0&1&2\\0&0&1\end{pmatrix}$, $T=\begin{pmatrix}1&0&1\\0&1&2\end{pmatrix}$, $\bar{K}'$ 是 3×1 维的列向量,且由它的前两个元素所构成的向量为 $T\bar{K}$,而最后一个元素为任意有理数.从中我们可以看出,数组 X 的复制度为 0,数组 Y 和 Z 的复制度都为 1,所以它们的复制度之和为 2.利用全搜索法可求得数组 X,Y 和 Z 的复制度之和的最小值为 2,即与贪婪启发式搜索法所得的结果是一致的.最后可求得

$$IP^{L_1}(x)=\begin{pmatrix}1&0&-1&1\\0&1&-1&2\\0&0&0&1\end{pmatrix}x+\begin{pmatrix}N-1\\N-1\\-1\end{pmatrix},$$

$$IP^{L_2}(x)=\begin{pmatrix}0&1&1&0\\0&0&2&1\\0&0&1&0\end{pmatrix}x+\begin{pmatrix}N-1\\N-1\\-1\end{pmatrix},$$

$$DP_X(x)=\begin{pmatrix}1&0&-1&1\\0&1&-1&2\\0&0&0&1\end{pmatrix}x+\begin{pmatrix}N-1\\N-1\\-1\end{pmatrix},$$

$$DP_Y(x)=\begin{pmatrix}1&1&0&-1\\1&1&-1&0\end{pmatrix}x+\begin{pmatrix}N-1\\N-1\end{pmatrix},$$

$$DP_Z(x)=\begin{pmatrix}-1&1&0\\-1&0&1\end{pmatrix}x+\begin{pmatrix}N-1\\N-1\end{pmatrix},$$

其中 $DP_Y(x)$ 和 $DP_Z(x)$ 分别是数组 Y 和数组 Z 的数据划分块到虚拟处理器空间前两维坐标的映射表达式.

在确定了 IP 和 $DP_{X_1},...,DP_{X_l}$ 的表达式之后,我们可以利用循环变换与数据变换技术将它们在具体的编程语言(例如 HPF 或 MPI)中实现,以完成对代码的优化.图 3 给出了对图 1 中的例子进行变换后的 HPF 代码示例图.

从图 3 中我们可以看出,变换后的代码使得计算所需的数据都在本地,因此其计算与数据被有效地对准了.由于嵌套循环的并行度和数组的复制度是相互关联的,所以我们可以通过牺牲嵌套循环的并行度来达到使数组的复制度减少或使其为 0 的目的.由于数组 Y 在第 2 个嵌套循环中被写访问,并且数组 Y 的复制度为 1,所以运行时,会有维护数组 Y 的一致性通信开销.为了使运行时完全无通信开销,我们可以只并行这两个嵌套循环的最外面两个循环,并且只分布模板 T 的前两维,这样虽然牺牲了嵌套循环的并行度,但却能使数组 Y 和 Z 的复制度为 0,从而使得运行时完全无通信.另外,为了能进一步优化程序性能,我们还可以在不影响已取得的优化效果的情况下利用数据变换对数组进行本地空间局部性优化,即当数组被分布到各个结点后,能被本地处理器按列进行访问(HPF 编译器缺省的数据存储方式).对于图 3 中的代码,我们可以对所有的数组进行置换数据变换来完成对数组 X 和 Z 以及第 2 个嵌套循环中数组 Y 的本地空间局部性优化.

```
REAL X(1:3N−2,2:4N−2,0:N−1,1:N),Y(1:3N−2,1:3N−2,1:N,1:N),Z(0:2N−2,0:2N−2,1:N)
!HPF$ TEMPLATE T(0:3N−2,0:4N−2,0:N−1)
!HPF$ ALIGN X(i1,i2,i3,i4) WITH T(i1,i2,i3)
!HPF$ ALIGN Y(i1,i2,i3,i4) WITH T(i1,i2, * )
!HPF$ ALIGN Z(i1,i2,i3) WITH T(i1,i2, * )
!HPF$ DISTRIBUTE T(CYCLIC,CYCLIC,CYCLIC)
!HPF$ INDEPENDENT
      DO i1=N−9,N+18
!HPF$ INDEPENDENT
        DO i2=max(2i1−2N−16,i1−N−7,−7)+N−1,min(2i1−2N+20,i1−N+20,29)+N−1
!HPF$ INDEPENDENT
          DO i3=max((i2−N−8)/2,i1−N−8,i2−i1−9,1)−1,min((i2−N+10)/2,i1−N+10,i2−i1+9,10)−1
            DO i4=max(−i2+2i3+N+2,−i1+i3+N+1,1),min(−i1+i3+N+10,−i2+2i3+N+11,10)
              X(i1,i2,i3,i4+1)=Y(i1,i2,i2−2i3+i4−N,i2−i3+2i4−N+1)+Z(i1,i2,i2+i4−N+1)
              X(i1,i2,i3,i1+i2−3i3+4i4−2N−2)=X(i1,i2,i3,i4+1)+
                                              Y(i1,i2,i1+i2−2i3+2i4−2N,i1+2i2−4i3+5i4−3N−1)
            ENDDO
          ENDDO
        ENDDO
      ENDDO
!HPF$ INDEPENDENT
      DO i1=N+1,N+13
!HPF$ INDEPENDENT
        DO i2=max(2i1−2N−5,3)+N−1,min(2i1−2N+10,30)+N−1
!HPF$ INDEPENDENT
          DO i3=max((i2−N−9)/2,i1−N−3,1)−1,min((i2−N)/2,i1−N,10)−1
            DO i4=1,10
              Y(i1,i2,i2−i3−N−1,−i1+2i2−i3+i4−N)=X(i1,i2,i3,i2−i3+2i4−N+2)
            ENDDO
          ENDDO
        ENDDO
      ENDDO
```

Fig.3 The transformed HPF code fragment example

图 3 变换后的 HPF 代码示例图

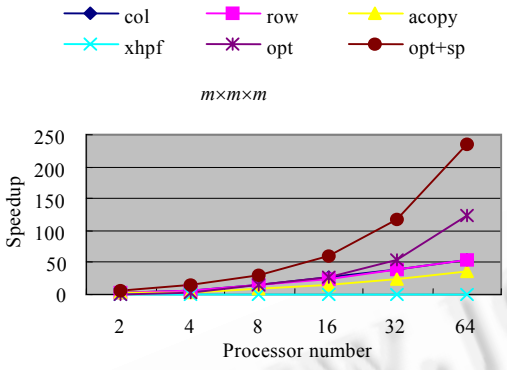
4 实验结果

我们对以下 4 个程序进行了测试:$m\times m\times m$ 是 3 个矩阵乘的计算程序,取自文献[14];stencil 是计算 5 点 stencil 的计算程序;vpenta 是取自 Spec92/NASA 基准程序测试包中的测试程序;hydrodynamics 是取自 Livermore 的核心基准测试程序.在上述测试程序中,$m\times m\times m$,vpenta 和 hydrodynamics 含有多个嵌套循环,因此它们的实验涉及全局计算与数据划分问题.我们的实验采用的是上述 4 个程序的 xHPF 版测试程序,并且每个测试程序都有 6 个不同的版本:只进行并行性分析,所有数组按列进行分布(用 col 表示);只进行并行性分析,所有数组按行进行分布(用 row 表示);只进行并行性分析,所有数组进行全复制(用 acopy 表示);xHPF 编译器用自带的优化策略(优化开关-auto=1)对程序进行优化后的版本(用 xhpf 表示);按本文介绍的方法(不包括本地空间局部性优化)所求得的优化版本(用 opt 表示);本文介绍的方法再加上本地空间局部性优化所得到的版本(用 opt+sp 表示).

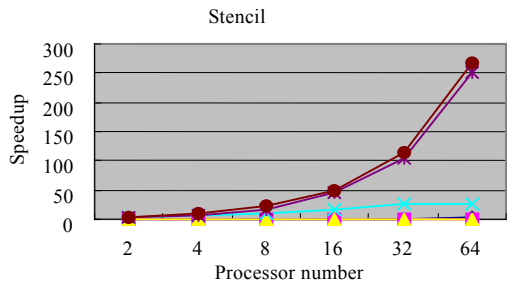
我们在一个具有 64 个 CPU 的分布主存多处理机上对上述 4 个程序进行了测试,该机器有 32 个结点,每个结点有两个 CPU,结点内采用共享存储结构而结点间采用分布存储结构.我们用 xHPF 对测试程序进行编译.图 4 给出了在不同的计算与数据分布方式下,测试程序的执行加速比.

通过对实验结果以及测试源程序进行分析,我们发现:由于不同数组的访问模式可能不同,所以对所有的数组都采用相同的数据分布方式(如按列分布或按行分布)可能只能满足部分数组本地化的要求,而对其余数组的访问则需要通信,因此不一定能获得较好的性能.对于全复制的数据分布方式,由于所有的数组都被盲目地复制到所有的结点上,如果被写访问的数组较多,而对这些数组的读引用又不多,那么虽然所有这些引用所需的数据能被本地化,但由于运行时被写访问数组的数据一致性维护开销可能会很大,这样反而会使性能变坏.xHPF 编译器的优化策略只能对嵌套循环进行简单的并行性分析以及对数组进行简单的数据分布模式分析,并且其对多个嵌套循环的优化是从第 1 个嵌套循环开始进行测试分析,并在此基础上依序对后续的嵌套循环进行分析,而不能基于全局对多个嵌套循环进行全局数据分布方式的分析,所以 xHPF 编译器的优化能力有限.我们的方

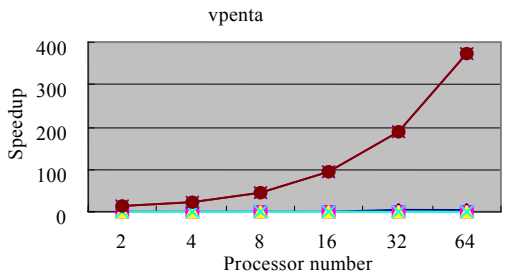
法能基于全局对较复杂的并行情形以及较复杂的数据分布模式进行分析,同时也考虑了数组复制问题,且能使数组的复制度尽量小,所以与前述的方法相比,我们的方法能取得较好的效果.实验结果也说明了这一点,在所有4个测试程序中,我们的方法都取得了比其他方法要好的效果,并且对于大部分测试程序(除了hydrodynamics之外),随着处理器数目的递增,其优化效果的差距越来越明显.*m*×*m*×*m* 和 stencil 这两个测试程序的实验结果表明,在对数据进行优化分布后,再进一步进行本地空间局部性优化将会取得更好的效果,所以对于分布主存多处理机系统,除了优化数据分布,使计算与数据有效对准外,单机的本地局部性优化也很重要.



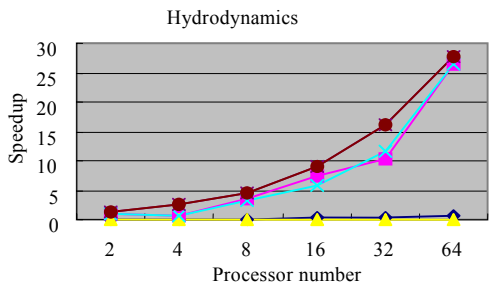
dv \ pn	2	4	8	16	32	64
col	2.223	6.046 8	14.969	25.79	39.567	55.503
row	2.830 7	5.921 4	13.594	24.823	37.708	55.503
acopy	2.067 7	3.933 9	7.570 1	13.879	23.644	35.072
xhpf	0.290 5	0.276 2	0.210 2	0.139 6	0.087 4	0.053 9
opt	1.307 4	4.243	13.787	27.956	53.31	123.45
opt+sp	7.415 4	15.196	30.507	60.526	117.61	235.22



col	0.063	0.117	0.249	0.467	0.833	1.852
row	0.047	0.090	0.190	0.345	0.684	1.429
acopy	0.105	0.091	0.063	0.050	0.048	0.044
xhpf	2.270	4.993	9.910	16.09	24.73	24.51
opt	2.488	6.233	16.15	46.04	102.9	252.5
opt+sp	3.572	8.268	23.68	49.31	115.8	268.8



col	0.116	0.339	0.827	1.366	2.520	4.549
row	0.276	0.542	0.835	1.024	1.337	1.387
acopy	0.218	0.398	0.189	0.153	0.149	0.128
xhpf	0.097	0.010	0.099	0.097	0.092	0.081
opt	11.30	21.94	46.64	93.20	186.7	372.3
opt+sp	11.30	21.94	46.64	93.20	186.7	372.3



col	0.017	0.048	0.116	0.170	0.349	0.686
row	1.095	0.771	3.699	7.353	10.17	26.47
acopy	0.141	0.080	0.041	0.033	0.029	0.022
xhpf	1.026	0.587	3.245	5.759	11.72	26.47
opt	1.192	2.700	4.518	8.976	16.07	27.66
opt+sp	1.192	2.700	4.518	8.976	16.07	27.66

Fig.4 The speedups under different computation and data decompositions

图4 不同计算与数据分布方式的执行加速比

图4的左边为加速比的图示,右边的表给出了与左图对应的加速比的具体数据,其中 *pn* 表示处理器数目,dv

表示不同的计算与数据分布方式($m\times m\times m$ 使用 1024×1024 的数组;stencil 使用 1024×1024 的数组;vpenta 使用 1840×1840×3 的三维数组以及 1840×1840 的二维数组;hydrodynamics 使用 2048×2048 的数组).

另外,$m\times m\times m$,stencil 和 vpenta 的实验结果中出现了超线性加速比,这主要是由如下两个原因引起的:第 1 个原因是因为加速比的计算是以原测试程序在单个处理器上的执行时间为基础的,由于原测试程序没有进行数据分布优化,运行时需要进行数据通信,所以 xHPF 编译器将会产生相应的数据通信代码,虽然在单个处理器上执行并不需要进行真正的数据通信,但这部分代码仍要执行.当原测试程序经过数据分布优化后,其运行时不需要进行数据通信,所以 xHPF 编译器将不会产生数据通信代码,因此优化后的测试程序将不需要执行数据通信代码而有可能产生了超线性加速比;第 2 个原因是测试程序的规模可能不够大,处理器个数越多,分布到各个处理结点上的数据就越少,这可能会减少 cache 的容量失效和冲突失效,从而产生超线性加速比.

5 结束语

计算与数据划分问题是影响并行程序在分布主存多处理机中执行性能的重要因素,也是并行编译优化研究的重点,针对该问题,本文提出了一种基于数据空间融合的全局计算与数据划分方法.该方法将计算空间划分成数量尽可能多的数据无关集合,以保证能最大限度地开发程序的并行性,并根据对计算空间的划分来组织数据,利用数据融合技术将对同一数据的多次引用融合成统一的集合,并据此对数据空间进行划分,以使得计算与数据能够有效对准.实验结果表明,该方法是可行的,并且取得了较好的效果.

在分布主存计算环境中,数据划分的选择是一个重要的研究问题,本文的方法是以最小化复制度为原则来选择数据划分,在下一步工作中,我们将分析和讨论以该原则来决定数据划分的适用范围.在第 4 节的实验中,我们是按照本文所提出的优化方法手工地对测试程序进行变换的.在下一步工作中,我们准备将本文所提出的优化方法在 SUIF(Stanford University Intermediate Format)并行编译器中实现,以使得新生成的 SUIF 并行编译器能使用我们的方法来对分布主存多处理机中的程序进行优化.

References:

[1] Chen TS, Chang CY. Skewed data partition and alignment techniques for compiling programs on distributed memory multicomputers. The Journal of Supercomputing, 2002,21(2):191~211.

[2] Chang WL, Chu CP, Wu JH. Communication-Free alignment for array references with linear subscripts in three loop index variables or quadratic subscripts. The Journal of Supercomputer, 2001,20(1):67~83.

[3] Shih KP, Sheu JP, Huang CH. Statement-Level communication-free partitioning technique for parallelizing compilers. The Journal of Supercomputing, 2000,15(3):243~269.

[4] Lim AW. Improve parallelism and data locality with affine partitioning [Ph.D. Thesis]. Palo Alto: Stanford University, 2001.

[5] Chen TS, Sheu JP. Communication-Free data allocation techniques for parallelizing compilers on multicomputers. IEEE Trans. on Parallel and Distributed Systems, 1994,5(9):924~938.

[6] Ramanujam J, Sadayappan P. Compile-Time techniques for data distribution in distributed memory machines. IEEE Trans. on Parallel and Distributed systems, 1991,2(4):472~482.

[7] Huang CH, Sadayappan P. Communication-Free hyperplane partitioning of nested loops. Journal of Parallel and Distributed Computing, 1993,19(2):90~102.

[8] Wolf M. High Performance Compilers for Parallel Computing. Redwood: Addison-Wesley Publishing Company, 1996. 137~510.

[9] Wolf M, Lam M. A data locality optimizing algorithm. In: Mauney J, ed. Proc. of the SIGPLAN'91 Conf. on Programming Language Design and Implementation. New York: ACM Press, 1991. 30~44.

[10] Shen ZY, Hu ZA, Liao XK, Wu HP, Zhao KJ, Lu YT. Methods of Parallel Compilation. Beijing: National Defence Industry Press, 2000. 31~127 (in Chinese).

[11] Anderson JM. Automatic computation and data decomposition for multiprocessors. Technique Report, CSL-TR-97-719, Palo Alto: Stanford University, 1997. 1~192.

[12] Ayguade E, Garcia J, Kremer U. Tools and techniques for automatic data layout: A case study. Parallel Computing, 1998,24(3-4):557~578.

[13] Li J, Chen M. The data alignment phase in compiling programs for distributed-memory machines. Journal of Parallel and Distributed Computing, 1991,13(2):213~221.

[14] Cierniak M, Wei L. Unifying data and control transformations for distributed shared-memory machines. In: Sestoft P, ed. Proc. of the ACM SIGPLAN'95 Conf. on Programming Language Design and Implementation. New York: ACM Press, 1995. 205~217.

附中文参考文献:

[10] 沈志宇,胡子昂,廖湘科,吴海平,赵克佳,卢宇彤.并行编译方法.北京:国防工业出版社,2000.31~127.