文章编号: 1001- 2486(2009) 04- 0062- 06

一种基于 0- 1 整数规划的全局数据分布优化方法*

夏 军, 庞征斌, 张 峻, 李永进
(国防科技大学 计算机学院, 湖南 长沙 410073)

摘 要: 数据分布是影响并行程序在分布主存多处理机上执行性能的重要因素。针对分布主存多处理机中的数据分布问题, 提出了一种基于 0- 1 整数规划、利用数据变换技术进行有效数据分布的方法。该方法通过数据变换技术改变数据的存储布局, 以使得数据能被有效地分布, 并且该方法还利用数据分布图描述程序被并行的情况及其所含数组被访问的情况, 并将全局数据分布优化问题转换为求解数据分布图中最优路径的问题, 从而可用 0- 1 整数规划求解最优路径问题。该方法能对多个嵌套循环中具有仿射数组下标的任意维数组进行有效的数据分布, 并且也能使嵌套循环的并行度尽可能地大。另外, 该方法也考虑了偏移常量的对准问题, 从而能使数据通信量尽量地小。实验结果验证了该方法的有效性。

关键词: 分布主存多处理机; 数据变换; 数据分布; 数据存储布局; 0- 1 整数规划
中图分类号: TP311 文献标识码: A

A 0- 1 Integer Programming Based Approach for Global Data Distribution

XIA Jun, PANG Zheng-bin, ZHANG Jun, LI Yong-jin
(College of Computer, National Univ. of Defense Technology, Changsha 410073, China)

Abstract: Data distribution is one of the key factors that affect the performance of programs running on distributed memory multiprocessors. This paper presents a 0-1 integer programming based approach for effective data distribution using data transformation techniques. This approach uses data transformations to change memory layouts and hence makes effective data distribution possible. Moreover, it also uses data distribution graphs to describe how programs are parallelized and how arrays are accessed, and transforms global data distribution problems into the problems of finding optimal paths in data distribution graphs. Therefore, 0-1 integer programming can be used to solve optimal path problems. The approach can effectively distribute multidimensional arrays with affine subscripts accessed in multiple loop nests and can exploit the parallelisms of loop nests as much as possible. In addition, it can also solve offset alignment problems. Thus data communication overheads can be reduced as much as possible. The experimental results show that the approach presented in this paper is effective.

Key words: distributed memory multiprocessors; data transformations; data distribution; data memory layouts; 0- 1 integer programming

分布主存多处理机上并行程序的执行性能在很大程度上取决于数据分布的好坏。在分布主存多处理机中, 每个处理器能直接访问自己的本地存储器, 间接访问其他处理器的远地存储器。由于远地访存的开销一般要远大于本地访存的开销, 因此, 为了使并行程序能在分布主存多处理机中有效执行, 用户或并行编译器必须选择一个较好的数据分布以使得处理器所需访问的数据都尽量在本地内存中, 从而减少远地访存通信开销。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对数据进行有效分布的问题进行了研究[1- 4], 在他们的研究中, 有的对嵌套循环以及数组的访问形式做了一定的限制, 有的没有考虑尽可能多地开发并行度的问题, 有的虽然考虑了该问题, 但却没有给出数据如何分布的方法, 还有的没有考虑偏移常量对准问题, 从而不能使得数据通信量尽量地小。我们曾在文献[5- 6] 中提出了基于数据融合的数据分布方法, 该方法能对单个嵌套循环进行有效数据分布, 但不能对多个嵌套循环进行全局最优的数据分布。我们还在文献[7] 中提出了一种基于 0- 1 整数规划的全局局部性优化方

* 收稿日期: 2009- 01- 27
基金项目: 国家杰出青年科学基金资助项目(69825104)
作者简介: 夏军(1976—), 男, 副研究员, 博士。

法, 但该方法只适合优化 Cache 局部性, 而无法解决数据分布问题。

本文针对分布主存计算环境中的全局数据分布问题, 提出了一种基于 0– 1 整数规划、利用数据变换技术来对数据进行有效分布的方法, 在程序已进行了并行性分析的基础上考虑全局数据分布问题。由于已进行了并行性分析, 所以可知道哪些循环可被并行, 称由可被并行的循环构成的集合为并行循环集。为每个数组构造一个数据变换空间, 该空间中的每个元素都为该数组的一个数据存储布局, 并且在该数据存储布局下, 该数组可能被有效地分布。然后, 在并行循环集中选择尽可能多的循环作为最终的并行循环, 并在每个数组的数据变换空间中选择一个数据存储布局作为该数组的最终数据存储布局, 以使得程序中的数组在所确定的并行循环以及数据存储布局条件下能被有效地分布。在该方法中, 通过数据变换技术来改变数组的数据存储布局, 以使得数据能被有效地分布, 并且还利用数据分布图来描述程序被并行的情况及其所包含数组被访问的情况, 并将全局数据分布优化问题转换为求解数据分布图中最优路径的问题, 从而可以用 0– 1 整数规划来求解最优路径问题。在所确定的代价模型以及数据变换空间内, 该方法可以求得最优解。该方法能对多个嵌套循环中具有仿射数组下标的任意维数组进行有效的数据分布, 并且也能使得嵌套循环的并行度尽可能地大。该方法还考虑了偏移常量的对准问题, 从而能使数据通信量尽量地小。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。

1　基本术语

一个 n 重嵌套循环 L 的迭代空间可以被看作是一个在 n 维空间里的多面体, 其中的每个点可由一个 $n \times 1$ 的列向量表示, 即: $\boldsymbol{I}=\left(i_1, \cdots, i_n\right)^{\mathrm{T}}$, 其中 $i_1, \cdots, i_n$ 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, 将 $\boldsymbol{I}$ 称作迭代向量。同样, 程序中所定义的 m 维数组 X 也定义了一个在 m 维空间里的多面体, 即数据空间, 且多面体的每个点代表一个数组元素, 它可以由一个 $m \times 1$ 的列向量来表示。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。基于上述假设, 数组引用可以表示为 $\boldsymbol{AI}+\boldsymbol{o}$, $m \times n$ 维矩阵 $\boldsymbol{A}$ 被称作引用矩阵, 具有 m 个元素的列向量 $\boldsymbol{o}$ 被称作偏移向量。

2　数据存储布局

给定一个 n 重紧耦合嵌套循环及其中被访问的 m 维数组 B 的 q 次引用 $\boldsymbol{A}_1\boldsymbol{I}_1+\boldsymbol{o}_1, \cdots, \boldsymbol{A}_q\boldsymbol{I}_q+\boldsymbol{o}_q$, 并假设该嵌套循环中 i_k 是并行循环。如果 $\exists 1 \leqslant p \leqslant m$, 数组 B 这 q 次引用的第 p 维下标表达式都为 ai_k (a 为非零整数), 那么就能很容易地确定按第 p 维分布数组 B, 使这 q 次引用的访存都能局部化, 否则可以利用文献[8] 中提出的方法求解数组 B 的能使访存局部化的非奇异仿射数据变换, 以使得数组 B 在经过仿射变换后, 其所有 q 次引用在某维中的下标表达式都为 ai_k。这即表明, 利用文献[8] 介绍的方法, 能求得 $1 \times m$ 维非零行向量 θ 和有理数 τ, 使得 $\theta\left(\boldsymbol{A}_1\boldsymbol{I}_1+\boldsymbol{o}_1\right)+\tau=i_k, \cdots, \theta\left(\boldsymbol{A}_q\boldsymbol{I}_q+\boldsymbol{o}_q\right)+\tau=i_k$。称二元组(θ, τ) 为能使数组 B 的 q 次引用的访存局部化的数据存储布局。

3　数据变换空间

给定一组紧耦合嵌套循环 $L_1, \cdots, L_f$ 以及其中被访问的若干数组 $X_1, \cdots, X_l$, 按如下的方法来确定数组 $X_1, \cdots, X_l$ 的数据变换空间 $\Omega_1, \cdots, \Omega_l$: 初始 $\Omega_1, \cdots, \Omega_l$ 都为空, 由于嵌套循环已经过并行性分析, 所以知道哪些循环可以被并行。针对每一个嵌套循环中的每一个可被并行的循环, 利用第 2 节介绍的方法, 可以求解在该嵌套循环中被访问的每个数组的能使所有引用访存都局部化的数据存储布局(以下简称存储布局) , 若求解成功, 就将求得的存储布局添加到各个数组对应的数据变换空间中, 最终就求得了数组 $X_1, \cdots, X_l$ 的数据变换空间 $\Omega_1, \cdots, \Omega_l$。

4 数据分布图

4.1 数据分布图的构造

给定一组嵌套循环,用数据分布图来描述这组嵌套循环被并行的情况及其所含数组被访问的情况。一个数据布局图由多个嵌套循环图构成,每个嵌套循环图与一个嵌套循环相对应。一个嵌套循环图又由多个循环图构成,每个循环图与该嵌套循环图所对应的嵌套循环中的一个并行循环相对应。循环图由一个或多个结点列连接而成,其中,每个结点列与包含该循环图的嵌套循环图所对应的嵌套循环中的一个数组相对应。结点列中的每个结点都代表与该结点列相对应的数组的数据变换空间中的一个存储布局。

在一个循环图中,从左至右以任意顺序一个接一个地放置结点列。对于任何两个相邻的结点列 X 和 Y(即与数组 X 和数组 Y 相对应),结点列 X 中的每个结点都与结点列 Y 中的每个结点相连接,因此,结点列 X 和 Y 之间共有 Layout(X) × Layout(Y) 条边,其中 Layout(.) 返回给定数组的数据变换空间中存储布局的个数。循环图有一个起始结点(用循环图所对应的并行循环的循环索引变量来标示)和一个终止结点,第一个结点列中的每个结点都与起始结点相连接,而最后一个结点列中的每个结点都与终止结点相连接。

给定一个嵌套循环,当完成对该嵌套循环中某个并行循环的循环图构造后,将它复制给该嵌套循环的所有并行循环,并用这些循环图各自所对应的并行循环的循环索引变量标示自己的起始结点。将该嵌套循环的所有循环图的起始结点连接在一起,终止结点连接在一起,就得到了该嵌套循环所对应的嵌套循环图。称连接所有循环图起始结点的结点为嵌套循环图的起始结点,连接所有循环图终止结点的结点为嵌套循环图的终止结点。

给定一组嵌套循环,当完成对这组嵌套循环中每个嵌套循环的嵌套循环图构造后,将它们排成一行并首尾连接起来,这样就得到了这组嵌套循环的数据分布图。本文给出的数据分布图是受 Kandemir[9] 所使用的存储布局图(Memory Layout Graph)的启发而得到的,他利用存储布局图来解决 Cache 局部性优化问题。

定义从嵌套循环图的起始结点到其终止结点的路径为该嵌套循环图的路径。将数据分布图中第一个嵌套循环图的起始结点称为数据分布图的起始结点,将最后一个嵌套循环图的终止结点称为数据分布图的终止结点。定义从数据分布图的起始结点到其终止结点的路径为该数据分布图的路径。给定数据分布图的一条路径,若该路径经过某个嵌套循环图的某个循环图,那这就表示为了能进行有效的数据分布,该循环图所对应的循环应为该嵌套循环图所对应的嵌套循环的并行循环;若该路径经过某个结点列中的某个结点,那这就表示该结点所代表的待选储存布局应为该结点列所对应数组的储存布局。

4.2 结点代价

所给出的结点代价是结点列中结点的代价。用 $V_Q^{xl}[j]$ 表示嵌套循环图 x 的循环图 l 中的数组 Q 所对应的结点列中的第 j 个结点。定义 $Cost(V_Q^{xl}[j])$(即结点 $V_Q^{xl}[j]$ 的代价)为当将循环 l 作为嵌套循环 x 的并行循环,并且将 $V_Q^{xl}[j]$ 所代表的待选存储布局作为数组 Q 的存储布局时,位于嵌套循环 x 的数组 Q 所引起的远地访存失效数。还定义数据分布图的路径的代价为该路径所经过的所有结点列中结点的代价总和。可以用文献[10]中的方法来估算 $Cost(V_Q^{xl}[j])$。$Cost(V_Q^{xl}[j])$ 估算得越精确,最后得到的数据分布结果就越好。一旦结点代价的估算方法确定了,那么本文所述方法就与如何估算结点代价相独立。

5 全局数据分布问题求解

5.1 问题描述

给定一组嵌套循环,本文提出的全局数据分布优化方法的目标是尽量减少远地访存失效,从而减少因远地访存失效而引起的通信开销。为了达到该目标,在每个嵌套循环可并行的循环中选择一个并行循环,并在每个数组的数据变换空间中确定一个存储布局,以尽量减少这组嵌套循环中数组访问引起的

远地访存失效, 这相当于求解这组嵌套循环的数据分布图中代价最小的一条路径。利用 0– 1 整数规划求解数据分布图的最优路径。

5.2　整数变量与目标函数

用 $H_{PQ}^{xl}[j, k]$ 表示嵌套循环图 x 的循环图 l 中结点列 P 的第 j 个结点与位于其右边的相邻结点列 Q 的第 k 个结点之间的边, 其中 $j \in [1, \cdots, \text{Layout}(P)]$, $k \in [1, \cdots, \text{Layout}(Q)]$。还用 $H_{PQ}^{xl}[j, k]$ 表示与其所代表边相关联的 0– 1 整数变量。给定数据分布图的一条路径, 如果 $H_{PQ}^{xl}[j, k]$ 所代表的边属于该路径, 那么 $H_{PQ}^{xl}[j, k]$ 取值为 1; 否则取值为 0。所以, 最终每个 $H_{PQ}^{xl}[j, k]$ 变量的值表示它所代表的边是否属于数据分布图的最优路径。给定一个数据分布图的路径, 其路径的代价可表示为:

$$\sum_{x}\sum_{l}\sum_{Q}\sum_{k=1}^{\text{Layout}(Q)}\left[\left(\sum_{j=1}^{\text{Layout}(P)} H_{PQ}^{xl}[j, k]\right) Cost(V_Q^{xl}[k])\right] \tag{1}$$

其中, x 遍历数据分布图中所有的嵌套循环图, l 遍历给定嵌套循环图 x 中所有的循环图, Q 遍历给定嵌套循环图 x 中的给定循环图 l 的所有结点列, P 为与给定嵌套循环图 x 中的给定循环图 l 的给定结点列 Q 相邻的左方结点列或循环图 l 的起始结点。全局数据分布优化的目标就是求数据分布图中的一条路径, 以使式(1) 的值最小。

5.3　限制条件

在确定了目标函数后, 还需确定 0– 1 整数变量 $H_{PQ}^{xl}[j, k]$ 的限制条件, 以使被选中的边和结点能构成数据分布图的路径, 为此, $H_{PQ}^{xl}[j, k]$ 应满足如下 3 个条件:

(1) 嵌套循环图的路径条件

在嵌套循环图中被选中的边和结点应能构成该嵌套循环图的路径, 即应满足:

$$\forall k \in [1, \cdots, \text{Layout}(Q)] \quad \sum_{j=1}^{\text{Layout}(P)} H_{PQ}^{xl}[j, k] = \sum_{s=1}^{\text{Layout}(R)} H_{QR}^{xl}[k, s] \tag{2}$$

其中, P、Q、R 为嵌套循环图 x 的循环图 l 中的 3 个相邻结点列。

(2) 嵌套循环图的单条路径条件

在嵌套循环图中被选中的边和结点应只能构成该嵌套循环图的一条路径, 即应满足:

$$\sum_{j=1}^{\text{Layout}(P)}\sum_{k=1}^{\text{Layout}(Q)} H_{PQ}^{xl_1}[j, k] + \cdots + \sum_{j=1}^{\text{Layout}(P)}\sum_{k=1}^{\text{Layout}(Q)} H_{PQ}^{xl_n}[j, k] = 1 \tag{3}$$

其中, $l_1, \cdots, l_n$ 为嵌套循环图 x 中的所有循环图, P、Q 为嵌套循环图 x 的循环图 l_1、……、循环图 l_n 中任意两个相邻的结点列, 且 P、Q 在循环图 l_1、……、循环图 l_n 中是对应相同的。

(3) 静态数据分布条件

若在嵌套循环 x 中, 数组 Q 的某个存储布局被选中, 那么在其他所有访问数组 Q 的嵌套循环中, 数组 Q 的同一存储布局也应被选中。静态数据分布条件如下:

$$\begin{aligned}
&\forall k \in [1, \cdots, \text{Layout}(Q)]:\\
&\sum_{j=1}^{\text{Layout}(P_1)} H_{P_1Q}^{x_1 l_1^1}[j, k] + \sum_{j=1}^{\text{Layout}(P_1)} H_{P_1Q}^{x_1 l_2^1}[j, k] + \cdots\\
=\ &\sum_{j=1}^{\text{Layout}(P_2)} H_{P_2Q}^{x_2 l_1^2}[j, k] + \sum_{j=1}^{\text{Layout}(P_2)} H_{P_2Q}^{x_2 l_2^2}[j, k] + \cdots = \cdots\\
=\ &\sum_{j=1}^{\text{Layout}(P_v)} H_{P_vQ}^{x_v l_1^v}[j, k] + \sum_{j=1}^{\text{Layout}(P_v)} H_{P_vQ}^{x_v l_2^v}[j, k] + \cdots
\end{aligned} \tag{4}$$

其中, 数组 Q 只出现在嵌套循环 $x_1, \cdots, x_v$ 中, $P_1, \cdots, P_v$ 分别为在嵌套循环图 $x_1, \cdots\cdots$, 嵌套循环图 x_v 中与 Q 相邻的左方结点列。$l_1^s, l_2^s, \cdots$为嵌套循环图 x_s 的所有循环图, $1 \leqslant s \leqslant v$。

5.4 问题求解

在确定了0-1整数规划所需的目标函数和限制条件后,就能在限定的数据变换空间以及结点代价的估算模型中求得数据分布图的最优路径,这即是全局数据分布优化问题的最优解。根据最优路径,可以在每个嵌套循环可并行的循环中确定一个并行循环,并在每个数组的数据变换空间中确定一个存储布局,从而使得数据能被有效地分布,以尽量减少远地访存失效。根据所确定的存储布局,利用文献[8]介绍的方法,可以构造相应的仿射数据变换以完成对数组的数据变换。

由于嵌套循环可被并行的循环可能不止一个,为了尽可能多地开发嵌套循环的并行度,应使嵌套循环中可并行的循环尽可能多地成为最终的并行循环。为此,在已根据数据分布图的最优路径为各个嵌套循环确定了一个并行循环和为每个数组确定了一个存储布局的基础上,可再在各个嵌套循环剩余的可被并行的循环中选择一个并行循环,在各个数组数据变换空间的剩余待选存储布局中选择一个与已确定的存储布局相容的存储布局。

这一过程可以反复进行,以使每个嵌套循环中可并行的循环尽可能多地成为最终的并行循环,从而最大限度地开发嵌套循环的并行度。由于篇幅受限,具体做法将不再详细阐述。

6 实验结果

对以下6个程序进行了测试:syr2k是求解带状对称矩阵的Rank-2K更新(update)的计算程序,取自BLAS;stencil是计算5点stencil的计算程序;mxm,vpenta和btrix是取自Spec92/NASA基准程序测试包中的测试程序;transpose是计算矩阵转置的程序。实验用的是上述6个程序的Fortran版测试程序,且它们已经过并行性分析。每个测试程序都有2个不同的版本:第一个版本是原测试程序(用original表示),第二个版本是按本文的方法进行全局数据分布优化后的程序(用optimal表示)。在一个具有64个CPU的分布主存多处理机上对上述6个程序进行了测试,该机器有32个结点,每个结点有2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。图1给出了在不同版本下,测试程序相对于单机执行时的加速比。

从图1给出的实验结果可以看出,第二个版本的执行效果明显优于第一个版本。对于第一个版本,当处理器数目增加到一定程度时,其执行性能不再提高,而继续增加处理器数目时,其性能反而降低了。由于第一个版本未进行数据分布优化,因此其运行时会导致较多的数据通信。虽然随着处理器数的增加,程序并行执行的力度会加大,但数据通信量也会随之增加,如果并行执行带来的性能提升不足以抵消数据通信量增加所带来的性能损失时,其执行性能反而会降低。由于第二个版本按本文的方法进行了有效的数据分布优化,所以其运行时数据通信量较少,随着处理器数目的增加,并行执行带来的性能提升占据了主导地位,因此其执行效果较好。

7 结束语

针对分布主存多处理机中的数据分布问题,提出了一种基于0-1整数规划的全局数据分布方法。该方法通过数据变换技术改变数据存储布局,以使得数据能被有效地分布。该方法还利用数据分布图描述程序被并行的情况以及其所含数组被访问的情况,并将全局数据分布优化问题转换为求解数据分布图中最优路径的问题,从而可用0-1整数规划来求解最优路径问题。在本文所确定的代价模型以及数据变换空间内,该方法可以求得最优解。

参考文献:

[1] Chen T S, Chang C Y. Skewed Data Partition and Alignment Techniques for Compiling Programs on Distributed Memory Multicomputers[J]. The Journal of Supercomputing, 2002, 21(2): 191-211.

[2] Chang W L, Chu C P, Wu J H. Communication-free Alignment for Array References with Linear Subscripts in Three Loop Index Variables or Quadratic Subscripts[J]. The Journal of Supercomputer, 2001, 20(1): 67-83.

[3] Shih K P, Sheu J P, Huang C H. Statement-level Communication-free Partitioning Technique for Parallelizing Compilers[J]. The Journal of Super-

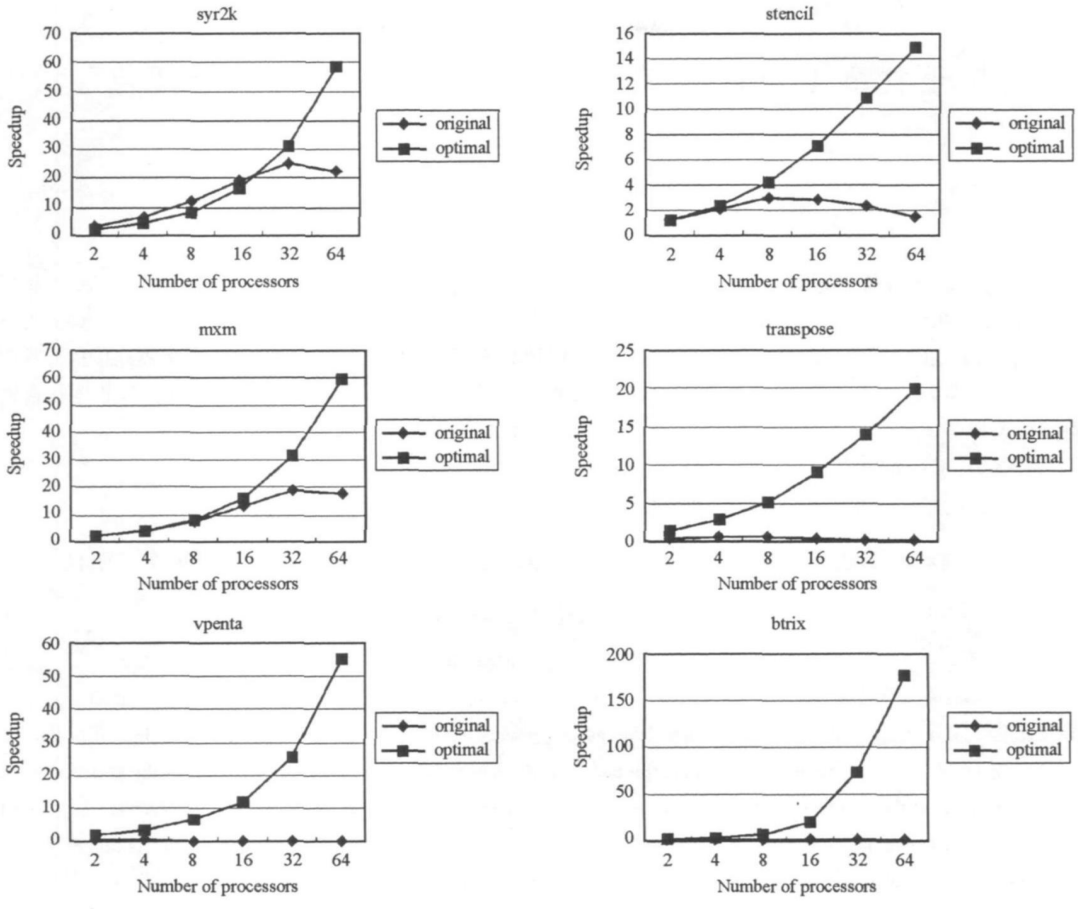

图 1　不同版本下测试程序的执行加速比

Fig. 1　Speedups of test programs under different versions

computing, 2000, 15(3): 243- 269.

[4] Lim A W. Improve Parallelism and Data Locality with Affine Partitioning[D]. Ph. D. Dissertation, Stanford University, Palo Alto, Cal, 2001.

[5] Xia J, Yang X J, Dai H D. Data Space Fusion Based Approach for Effective Alignment of Computation and Data[C]// Proc. of 5th International Workshop on Advanced Parallel Processing Technology, Xiamen, China, 2003: 215- 225.

[6] Xia J, Yang X J. A Data Transformations Based Approach for Optimizing Memory and Cache Locality on Distributed Memory Multiprocessors[C]// Proc. of 6th International Workshop on Advanced Parallel Processing Technology, Hongkong, China, 2005: 3- 12.

[7] Xia J, Luo L, Yang X J. A 0- 1 Integer Linear Programming Based Approach for Global Locality Optimizations[C]// Proc. of 11th Asia-pacific Conference on Advances in Computer Systems Architecture, Shanghai, China, 2006: 281- 294.

[8] 夏军, 杨学军. 一种面向分布主存多处理机的有效数据分布方法[J]. 计算机工程与科学, 2005, 27(10): 73- 76.

[9] Kandemir M, Banerjee P, Choudhary A. Static and Dynamic Locality Optimizations Using Integer Linear Programming[J]. IEEE Transactions on Parallel and Distributed Systems, 2001, 12(9): 922- 940.

[10] Sarkar V, Gao G, Han S. Locality Analysis for Distributed Shared-memory Multiprocessors[C]// Proc. of 9th Int'l Workshop Languages and Compilers for Parallel Computing (LCPC'96), 1996.