维普资讯 http://www.cqvip.com

计算机科学2002Vol.29№.11

分布内存系统中节点间软流水优化技术*)

Exploiting Inter-node Pipelining Parallelism in Distributed Memory Systems

陈 莉 张兆庆 冯晓兵

(中国科学院计算技术研究所体系结构研究室 北京100080)

Abstract Maximize parallelism and minimize communication overheads are important issues for distributed memory systems. Communication and data redistribution cannot be avoided even when considering global optimization of data distribution and computation decomposition. A new approach based on loop fusion is presented exploiting pipelining parallelism, thus communication overhead can be hidden and data redistribution can be avoided. This technique exploits pipelining from complex loop structures, which distinguishes itself from traditional pipelining techniques. Experiments show that the technique is superior to other optimizations.

Keywords Parallel compiling, Data redistribution, Software pipeline, Parallel loop, Loop fusion

1. 引言

目前分布式处理系统的并行编译技术还处于研究阶段，对于实际应用程序还难以获得高性能。在程序中充分发掘并行性和极小化通信的开销是需要协同考虑的两个方面。考虑全局数据分布和计算分割时，在应用程序中仍然可能存在这样一些情况：在程序中按全局优化数据分布策略，某数组采用一种数据分布方式会取得好的效果，但对某些循环而言，该数组必须采用另一种分布模式，才能并行。这种情况下处理办法一般有两种：其一是在循环体中引入同步和/或通信；其二是采用数据重分布(remap)。

在循环体中引入同步和通信。当外层循环的迭代次数较小时，通信的代价是可以接受的；反之，当外层循环的迭代次数很大，内层通信(含同步)无法实现通信外提时，通信的开销甚至大于数据的重分布。

第二种策略是采用数据重分布。数据重分布在存储、通信上的开销非常大。每次数据重分布，通信总量相当于被重分布的数组的总规模，通信的次数至少是 n * (n-1)，其中 n 是处理机的个数，且需要解决网络竞争。如果重分布点出现在一个循环中，通信开销呈数量级增长。

本文提出一种对可分割的循环进行合并，用以获取流水并行的方法。对数据分布与可选的循环分割不一致的循环套，通过对内层并列的可分割循环进行合并，得到可分割的流水循环。紧嵌的流水循环，有机会通过循环交换降低通信的次数，更好地隐藏通信的开销。这一优化技术已经在中科院计算所先进编译技术小组为并行机曙光3000开发的并行优化编译系统 AutoPar3.0中实现。

1.1 相关工作

数据分布与可选的循环分割不一致时，流水是解决此矛盾的一种优化手段。流水的可分块性，易于实现通信的向量化，较好地实现通信的隐藏；Fortran D 优化编译器[1]通过循环交换、分块(strip-mining)调整流水的粒度，降低通信开销。PARADIGM 利用粗粒度流水[2]解决 DOACROSS 循环的并行化问题。粒度的选取对于流水性能有重要的影响，文[2～4]讨论了分布内存系统上对流水优化的程序如何恰当地选择流水粒度。

传统的流水优化只考虑 DOACROSS 循环。而我们关注的是复杂循环结构，其循环体是多个并列的并行循环，并行循环的分割不一致会在循环体中引入大量通信。

循环合并是一种重要的循环重构技术，有利于提高数据局部性、增大并行循环的粒度从而减少同步、降低循环开销。串行循环合并的数据依赖条件[5]是循环间没有后向(违反字典序)的依赖关系，这种依赖称为阻碍合并的依赖。

文[6]提出通过代码复制、数据冗余解决循环对准冲突，获得循环套无同步并行执行的技术。这种解决方案将导致循环体以指数级增大。

文[7,8]以图论为基础，把最优循环合并转化为无环划分的相关问题，研究极大化并行、提高数据局部性和寄存器需求的最优循环合并。但他们不处理阻碍合并的依赖；针对共享内存的机器，文[9,10]讨论通过并行循环套的合并获取 cache 局部性、极小化同步数的方法；文[9]的核心是对循环间后向的依赖采用对准消除，前向的依赖采用 peeling 消除。合并后成为一个主循环和若干 peeled 循环，主循环构成一个更大的并行区间。但主循环和若干 peeled 循环之间存在通信，而且由于 peeled 循环中计算分割与数据分布的不一致，peeled 循环后将引入通信。对于我们关注的复杂循环结构和分布内存系统，它会带来更多的通信。

本文给出可分割循环合并的条件；讨论可分割循环的对准；提出发掘流水并行的技术实现。通过对 NAS 标准测试程序包中几个程序的几种并行方案的分析比较，说明了发掘流水并行对提高并行性和降低通信开销的作用。

2 可分割循环及其合并

本文的工作基于以下几点假设：首先，应用程序中的数组都是规则引用，也即数组引用的下标表达式是循环索引量的线性形式；第二，循环是规则的，也即任何数据依赖的距离向量的各个分量都是已知常量；第三，循环是半正规化的，即循环索引量的步长非负；第四，我们只考虑一维处理器序列的映

*)本工作得到国家高性能计算基金、国家自然科学基金项目(69933020)的资助。陈 莉 博士研究生，研究方向为并行编译。张兆庆 研究员，研究领域包括并行编译技术、并行程序设计环境、自动并行化与并行可视化工具。冯晓兵 副研究员，研究领域包括并行编译、动态优化技术。

维普资讯 http://www.cqvip.com

射和数组的一维块分布。

我们所谓的循环套，不一定是最外层的循环套，也可能是具有共同外层循环的并列循环套。循环套中紧嵌循环的编号从最外层依次为1，2，…。称两个循环套的相容层次是k，如果两循环套从1到k层的各对应循环的索引变量具有相同的初值、终值和步长。否则，称两循环套的循环头不相容，或称相容层次是0。

定义1[11] 循环套L称为可流水的，如果存在分割映射Φ(把迭代集映射到多维处理器网格)，对L的任意一个数据依赖关系 $S(\vec{i})\delta S'(\vec{i'})$，有 $\Phi_{s'}(\vec{i'})-\Phi_s(\vec{i})\geqslant\vec{0}$。且存在一个数据依赖关系 $S_1(\vec{i})\delta S_2(\vec{i'})$，有 $\Phi_{s2}(\vec{i'})-\Phi_{s1}(\vec{i})>\vec{0}$。其中S、S'、$S_1$、$S_2$是循环套中的语句。

流水的本质是，任意数据依赖关系在处理器网格的每一维上总是单向流动，因而通信也是单向的。对于一维处理器网格，显然有以下性质：

性质1 循环套L的紧嵌深度为n，且d(d≤n)层循环不是并行循环，对L的任意数据依赖关系 $S(\vec{i})\delta_\theta S'(\vec{i'})$，$\theta$是对应的依赖距离向量，$\theta_d\geqslant 0$，则循环套L是可流水的。

证明是显然的。我们只需对d层循环进行块分割映射Φ，容易证明Φ满足定义1。

可流水循环的定义并没有考虑循环分割与数据分布的一致性问题，而这对分布内存的系统来说是至关重要的。

定义2 紧嵌深度为n的循环套L，设其紧嵌的各循环为$L_i(1\leqslant i\leqslant n)$，L可能有多个并行循环。我们称并行循环$L_k$是可分割的，如果在已知的数据分布模式下，分割循环$L_k$引入的通信最少。

定义3 两个循环套N_1、N_2的相容层次是c，可分割循环都在p层，其中$p\leqslant c+1$，则称L和Q是分割一致的，max(p，c)是N_1、N_2的合并层次。

合并层次亦即合并后循环套的紧嵌深度。只对相容的循环层进行合并的要求过于苛刻，实际中许多循环只相差了很少的迭代数。定义3中$p\leqslant c+1$，表明我们只要求p-1层的相容性。如果循环套N_1、N_2的可分割循环都在$p=c+1$层，即max(p，c)层，且循环间没有阻碍合并的依赖，则循环体可在$c+1$层合并，对$c+1$层循环的迭代集取两者的并，并为循环体引入guard。可证，分割一致的关系具有自反、对称、传递的性质。

循环间依赖关系图是我们考察循环合并合法性的基础。分割一致的循环套N_1、N_2在c层相容，其可分割循环在p层。把两个循环套看成具有共同的max(p，c)层外层循环，进行循环间的依赖测试，得到循环间依赖关系图$G=(V,E,W)$，其中顶点集V表示各循环中的语句，边集E表示语句间的数据依赖关系，W是E到依赖距离向量的映射。由于本文考虑的是规则的循环，依赖距离向量的各个分量均是常数。

```
do i=1, 100
do j=1, 100
        a(i,j)=...
do i=1,99
do j=2,99
    ...=a(i+1,j+1)+a(i,j-1)
```

图1 虚线是后向的依赖

循环套N_1、N_2之间的数据依赖会出现三种情况：1)无依赖；2)依赖的携带循环是包含N_1、N_2的外层循环；3)是N_1、N_2之间的循环无关依赖。

定义4 分割一致的循环套N_1、N_2，可分割循环在p层，其循环间依赖关系图是$G=(U,E,W)$。对$e\in E$，我们称e是阻碍可分割循环合并的依赖，当且仅当(1)或(2)成立：

(1)e是后向的数据依赖关系，即：对$j=\min\{r\mid w(e)_r\neq 0\}$，有$w(e)_j<0$。

(2)e是前向的数据依赖关系，且$w(e)_p<0$。

定义4中的两类阻碍可分割循环合并的依赖，分别被称为(1)类依赖和(2)类依赖。

性质2 程序$P=\{N_1、N_2\}$，循环套N_1、N_2是分割一致的，可分割循环在p层，合并层次是c，如果循环间不存在阻碍可分割循环合并的数据依赖，设N_1、N_2在c层合并得到循环套N，则N的p层循环是可分割循环或是可流水的循环。

证明：设P的循环间依赖关系图是$G=(U,E,W)$。对于$\forall e\in E$，由于循环间没有阻碍可分割循环合并的依赖，e是前向的数据依赖，可知$w(e)_p\geqslant 0$。分两种情况考虑：1)如果存在$e'\in E$，使得$w(e')_p>0$，则N的p层循环是可流水的。2)如果$\forall e\in E, w(e)_p=0$，则N的p层循环是并行循环，只需证N的可分割层仍在p层。反证法：假设N的可分割层在p'层，$p\neq p'$，$p'\leqslant c$，则N_1、N_2在p'层都是并行循环。因为对循环套N分割p层比合并前没有引入新的通信，所以至少存在i，$1\leqslant i\leqslant 2$，使循环套N_i分割p'层循环会比分割p层引入更少的通信。这与前提矛盾。证毕。

3 可分割循环的对准

阻碍可分割循环合并的数据依赖，可以通过对准来消除。对准变换用对准向量来描述。向量运算+、-、max定义为对向量的对应分量作max、+、-运算。

定义5 L是一个紧嵌深度为n的循环套，Align(L)是一个n维向量，循环套L以Align(L)进行对准变换，也即循环套L的迭代空间发生如下变换：

$\vec{i}+\text{Align}(L)=\vec{i'}$，其中$\vec{i}$是原来的迭代向量，$\vec{i'}$是变换后的迭代向量。

我们称Align(L)是L的对准向量。

性质3 l_1、l_2是两个分割一致的循环套，其可分割循环在p层。数据依赖关系λ的依赖源在循环套l_i中，依赖目的在循环套l_j中$(1\leqslant i,j\leqslant 2)$，λ对应的距离向量是v；循环套$l_1$、$l_2$分别以对准向量$\text{Align}(l_1)$、$\text{Align}(l_2)$作对准变换，则变换后λ对应的依赖距离向量变成$\text{Align}(l_j)-\text{Align}(l_i)+v$。

证明：只需要考虑$i\neq j$的情形。对性质3中的数据依赖λ，给出其依赖系统，该整数系统的解是依赖距离向量v。根据对准变换对各循环索引量进行变量替换，则与循环对准后的依赖系统等价。把对准后的循环索引量按替换式代入，则易推出循环对准后λ对应的距离向量是$\text{Align}(l_j)-\text{Align}(l_i)+v$。证毕。

程序$P=\{l_1,l_2,\cdots,l_n\}$，循环套$l_1,l_2,\cdots,l_n$是分割一致的。浓缩P的循环间依赖关系图G得到循环合并图$H=(V,F,\text{Align})$。其中V中每个顶点表示一个循环套；F是边集，$(l_i,l_j)\in F$，如果存在从循环套l_i到l_j的数据依赖；Align是V到对准向量的映射。

算法1 求对准向量
输入：程序$P=\{l_1,l_2,\cdots,l_n\}$，循环套$l_1,l_2,\cdots,l_n$是分割一致的，可分割循环在p层；

维普资讯 http://www.cqvip.com

输出:1)循环对准变换后,循环间的依赖关系图 $G'=(U,E,W')$;2)循环合并图 $H=(V,F,Align)$

①构造 P 的循环间依赖关系图 $G=(U,E,W)$

②构造循环合并图 $H=(V,F,Align)$,初始化:$\forall l_i\in V, Align(l_i)=\vec{0}$.
为 F 引入相对对准映射 dist:$\forall (l_i,l_j)\in F, dist(l_i,l_j)=\vec{0}$.

③for($e\in E$,设 e 的依赖源、依赖目的分别在循环套 l_i、l_j 中,l_i、l_j 的合并层次是 m)
3.1)if$(w(e)_p<0)$ $dist(l_i,l_j)_p=\max(dist(l_i,l_j)_p,-w(e)_p)$;
3.2)if (e 是阻碍合并的(1)类依赖关系,up= min{r| $w(e)_r>0$或 r=m+1})
for ($k=\min\{r|w(e)_r\neq 0\}$;k<up;k++)
$dist(l_i,l_j)_k=\max(dist(l_i,l_j)_k,-w(e)_k)$;

④for (H 上的任意回路 C,不妨设 $C=e_0,e_1,e_2,\cdots,e_s,e_0$)
if(存在 $k,0\leqslant k\leqslant s$,使得 $dist(e_k)\neq\vec{0}$)
该循环序列不能进行流水优化,退出;

⑤在循环合并图上对 Align 进行传播
$Align(v)=\max\{Align(u)+dist(u,v)|(u,v)\in F\}$

⑥根据性质3,得到循环对准变换后的循环间依赖关系图 $G'=(U,E,W')$

性质4 循环套 NEST,紧嵌深度是 d,其中 d 层循环体是 n 个分割一致的循环套序列 $l_1,l_2,\cdots,l_n$,可分割循环在 p 层;各循环套采用算法1求出的对准向量进行对准变换后,得到循环套 $l'_1,l'_2,\cdots,l'_n$,则 $l'_1,l'_2,\cdots,l'_n$ 之间不存在阻碍合并的依赖。

证明:设 λ 是 NEST′ 中任意的数据依赖,依赖源在 l_i 中,依赖目的在 l_j 中,其距离向量为 v',l_i、l_j 的合并层次是 m。设 λ 在原循环套 NEST 中依赖距离向量是 v。则由性质3知 $v'=v+Align(l_j)-Align(l_i)$。下面分3步证明。1)由算法1的步骤⑤中的传播知:$Align(l_j)-Align(l_i)\geqslant dist(l_i,l_j)$,由步骤③知 $dist(l_i,l_j)\geqslant\vec{0}$,则有 $v'\geqslant v$。若 λ 在原循环套 NEST 中不是阻碍合并的依赖,由 $v'\geqslant v$,可知 λ 在 NEST′ 中也不是阻碍合并的依赖。2)若 $v_p<0$,则由步骤③的3.1知有 $dist(l_i,l_j)_p\geqslant -v_p$,而 $v'\geqslant v+dist(l_i,l_j)$,则 $v_p'\geqslant 0$。如果 λ 在原循环套 NEST 中是阻碍合并的(1)类依赖,在步骤③的3.2中令 $lo=\min\{r|w(e)_r\neq 0\}$,则 lo 是距离向量 v 中第一个负分量的下标,up 是 v 中第一个正分量的下标或满足 up=m+1。由步骤③的3.2知,$\forall k, lo\leqslant k<up$,有 $dist(l_i,l_j)_k\geqslant -v_k$;而 $v'\geqslant v+dist(l_i,l_j)$,则有 $v'_k\geqslant 0$,从而 λ 在新循环套 NEST′ 中是前向的依赖。而 $v'_p\geqslant 0$,从而 λ 不是阻碍合并的依赖。3)若 λ 在原循环套 NEST 中是阻碍合并的(2)类依赖,由2)知 $v'_p\geqslant 0$,又 $v'\geqslant v$,从而 λ 在新循环套 NEST′ 中不是阻碍合并的依赖。证毕。

例1 以下程序片断取自 NAS 的 NPB2.3-serial 中的 SP 程序,全局数组 frct 和 u 按最高维分布,局部数组 flux,ue 复制分布。循环体由4个分割一致的可分割循环组成。

```
do j=2, ny-1
do i=2,nx-1
(1)do k=2,nz
      flux(2, k)=u(2,i,j,k) * u(4,i,j,k)/u(1,i,j,k)
   enddo
(2)do k=2,nz-1
      rsd(2,i,j,k)=rsd(2,i,j,k)- tz2 * (flux(2,k+1)-flux(2,k-1))
   enddo
(3)do k=2,nz
      flux(2, k)=tz2 * (1/u(1,i,j,k)-1/u(1,i,j,k-1) * u(2,i,j,k-1) )
   enddo
(4)do k=2,nz-1
      rsd(2, i, j, k)=rsd(2,i,j,k)+tz2 * (flux(2,k+1)-flux(2,k) )
   enddo
enddo
enddo
```

对准变换会使循环套对非本地数据的存取区间变大,这是因为对准变换导致引用数组的下标偏移增大。对例1实施算法1,得到如图2的循环间依赖关系图、循环合并图。各循环套的对准向量分别为:0,1,2,3。如果对循环(1)(2)的局部数组 flux 进行换名,则循环间依赖关系图如图3所示。各循环的对准向量分别是:0,1,0,1。

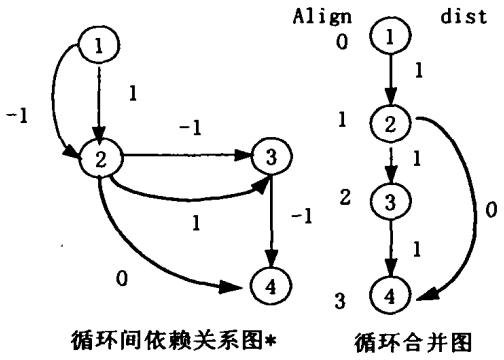


图2 换名前

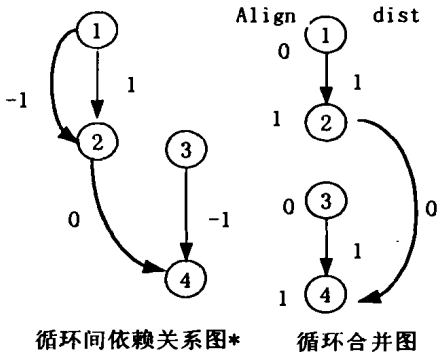


图3 换名后

注:为简化起见,上图中循环间依赖关系图中顶点对应的是循环套,而不是语句;标记为负的边对应阻碍合并的依赖关系。

4 发掘流水并行

程序 $P=\{l_1,l_2,\cdots,l_n\}$,循环套 $l_1,l_2,\cdots,l_n$ 是分割一致的,可分割循环在 p 层。第3节的对准变换能消除所有阻碍可分割循环合并的依赖,从而循环套可以逐对合并(在合并层次)。由性质2知,合并后 p 层循环是可分割循环或可流水的循环。合并时,(1)类依赖在 j≠p 时(j 是距离向量中负分量的最小下标)会导致对其他循环层的对准,这使得其他循环层的循环头出现不相容,合并后的循环将引入新的 guard。

流水所带来的通信有:1)流水通信;2)反向更新;3)复制分布数组的更新。流水通信由流依赖引起,当流依赖的距离向量在可分割层的分量非0时,这就是该定值引起的流水宽度;反向更新由分布数组上的输出依赖引起,当某输出依赖在可分割层的分量非0时,流水结束后,数组分布区的数据不是最新的,某些片断的最新值在逻辑后一个节点。因此流水结束后需要一次反向更新(与流水方向相反);若复制分布的数组在流水循环中被定值,且在循环后是活跃的,则流水结束后需要对复制分布数组更新。

流水的粒度对程序性能有很大的影响,太细的粒度使通信的开销激增。如果能判定合并后的可流水循环不能与任何外层循环交换,我们将放弃流水的优化。循环交换是一种单模变换,一个数据依赖称作阻碍循环交换的,当且仅当其方向向量在单模变换后违反字典序。

维普资讯 http://www.cqvip.com

性质5 循环套 NEST，紧嵌深度是 d，其紧嵌的各层循环依次为 $L_1', L_2', \cdots, L_d'$，其中 L_d' 的循环体是 n 个分割一致的循环套序列 $l_1, l_2, \cdots, l_n$，可分割循环在 p 层，取 $1 \leqslant j \leqslant d$。数据依赖关系 λ 不是阻碍可分割循环层与 L_j' 交换的。则采用算法1的对准变换后，λ 对应的依赖仍不是阻碍可分割循环层与 L_j' 交换的。

证明：设 λ 在对准变换后的循环套中对应的距离向量是 r′、方向向量是 v′。可证明：对准变换不改变循环套内部的数据依赖的距离向量，只改变循环间依赖的距离向量。若 λ 是循环套内的依赖，则有 $r_p'=0$，即 v_p' 是＝。若 λ 是循环套间依赖，由性质4知 λ 在对准变换后的循环套中不是阻碍可分割循环合并的依赖。由性质2的证明知，$r_p' \geqslant 0$，即 v_p' 是＞。两种情况均可推出，v′ 在任何交换变换（p 层循环与外层 L_j' 的交换）后不违反字典序。证毕。

算法2 循环合并的流水优化

输入：循环套 NEST，紧嵌深度是 d，其 d 层循环体是 n 个分割一致的循环套序列；

输出：流水优化的循环套 NEST′；

①调用算法1，求循环合并图 H；

②if（可分割循环不能与任何外层循环交换，且 NEST′是可流水循环）放弃进行优化的企图，返回；

③按照拓扑序遍历合并图 H，逐对进行循环合并；

④调整流水的粒度，必要时把可分割循环与外层循环交换；

⑤进行循环分割；

⑥根据 NEST′的依赖关系图确定：流水通信的片段、反向更新的通信片段；

⑦确定流水结束后，是否更新复制分布数组；

再次分析例1。对循环套(1)(2)中 flux 数组进行换名，如图2得到各个循环套的对准向量分别是：0，1，0，1。对准变换没有导致对分布数组的跨节点的输出（或反）依赖，从而不会引入反向更新。flux 数组在循环后不活跃，不需要更新。循环(1)和循环(3)中 tmp、flux 数组的定值均引起流水通信。因为可分割层在内层，需考虑循环交换以调整流水的粒度，为此对数组 flux、tmp 扩维。流水并行处理后，代码如下：

```
do j=2, ny-1
  从前一节点接收流水宽度是2的 tmp 片段，和流水宽度是1的 flux 片段.
  do k=ap_loop_l, ap_loop_u / * 分割后的循环界 * /
  do i=2,nx-1
    tmp(2,k,i)=u(2,i,j,k) * u(4,i,j,k)/u(1,i,j,k)
    if (k>=3) rsd(2,i,j,k-1)=rsd(2,i,j,k-1)- tz2 * (tmp(2,k,i)-
      tmp(2,k-2,i))
    if (k>=2) flux(2,k,i)=tz2 * (1/u(1,i,j,k)-1/u(1,i,j,k-1) *
      u(2,i,j,k-1))
    if (k>=3) rsd(2,i,j,k-1)=rsd(2,i,j,k-1)+tz2 * (flux(2,k,
      i)-flux(2,k-1,i)
  enddo
  enddo
  向后一节点发送流水宽度是2的 tmp 片段，和流水宽度是1的 flux
    片段.
enddo
```

5 测试结果与分析

我们对 NAS 并行程序集的 NPB2.3-serial 中的几个程序片断进行了测试。下面的实验数据来自程序中能够实现流水优化的子程序。对比数据来自其它4种变换：1）shift-and-peel 变换；2）直接分割（通信发生在最内层，且是双向通信）；3）通信外提一层，需要循环分布和数组扩维；4）对最外层循环分割（数组重分布）。

shift-and-peel 变换与直接分割的性能相当。1）的 shift-and-peel 变换作了优化：先对循环分布、再考虑循环的 shift-and-peel 变换。这里也需要数组扩维。

实验在并行机曙光2000-I 上进行。每个节点是一个 PowerPC 604e 处理器，主频300MHz，节点间的高速通讯网采用 Myrinet。运行数据取各节点上运行时间的最大值，每个实验做了5次，每次运行舍弃第一次调用的数据（考虑到 cache 局部性），取各节点运行时间最大值再求平均值。每个程序均比较 A、B 两个规模，A 规模在 XYZ 三方向上分别取64个格点，B 规模分别取102格点。对 A 规模，采用4节点计算；对于 B 规模，采用6节点计算（BT 采用8节点）。

SP 的 rhs 过程计算稳态剩余子，在主迭代中出现，串行计算时间占总计算时间的17.1%，其第3个循环是可分割循环与数组分布不一致的复杂循环。流水具有最好的性能；通信外提方式略好于 shift-and-peel；最差的是重分布方式和直接分割。

BT 的 jacz 过程是对 Z 方向扫描，构造三对角块系统，串行计算时间占总时间的15%。流水具有最好的性能；通信外提的优化明显好于 shift-and-peel 变换，这是因为 jacz 中涉及的分布数组规模较大，shift-and-peel 变换中数组定值与分布不一致的矛盾突出；重分布方式非常差。B 规模时重分布方式由于时间太长，被放弃。

从实验可以看出，流水优化具有最好的性能。而 shift-and-peel 变换和通信外提的优化方式都需要循环分布的支持，而很多情况下，循环分布是不一定能够实现的。流水具有最大的适用范围。

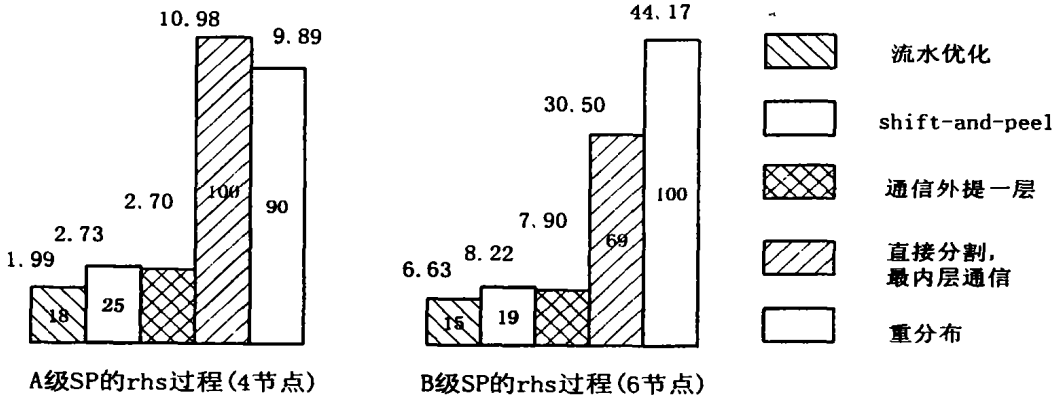
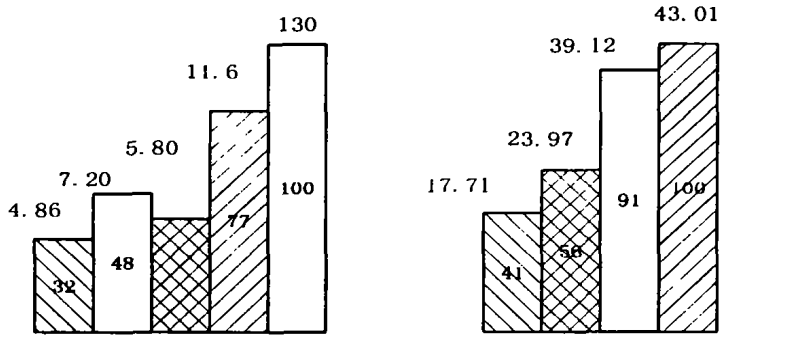


图4 （时间单位是秒）

维普资讯 http://www.cqvip.com



A级BT的jacz过程(4节点)　　B级BT的jacz过程(8节点)

图5 (时间单位:秒)

结论及将来的工作 本文讨论了通过发掘节点间流水,有效提高程序并行性和降低通信开销的技术,它对于解决规则的数组引用问题,具有普遍的积极意义。

流水循环中粒度的选取是一个重要课题,我们现在的方法是采用静态工作集估算的方法,粒度的选取不一定是最佳的。将来的工作包含:采用运行时流水粒度选择,处理更复杂的数据分布形式、支持更复杂的循环分割形式及如何使目标代码更为简洁。

参 考 文 献

1 Hiranandani S, et al. Evaluation of Compiler Optimization for FortranD on MIMD. In: 1992 Intl. Conf. on Supercomputing, Washington, D.C, July 1992

2 Banerjee P,et al. An Overview of the PARADIGM Compiler for Distributed Memory Message-Passing Multicomputers. IEEE Computer, Oct. 1995. 37～47

3 Lowenthal D K, James M. Run-Time Selection of Block Size in Pipelined Parallel Programs. Second Merged IPPS/SPDP Symposium,1999. 82～87

4 Van der Wijngaart R F, et al. Analysis and Optimization of Software Pipeline Performance on MIMD Parallel Computers. Journal of Parallel and Distributed Computing, 1996

5 Wolfe M. High Performance Compilers for Parallel Computing. Addison-Wesley Reading, MA, 1996

6 Callahan C D. A Global Approach to the Detection of Parallelism: [PhD thesis]. Dept. of Computer Science, Rice Univ., Mar. 1987

7 McKinley K S,Carr S,Tseng C-W. Improving Data Locality With Loop Transformations. ACM Transactions on Programming Languages and Systems,1996,18(4)

8 Singhai S K,McKinley K S. An Algorithm for Improving Parallelism and Cache Locality. The Computer Journal,1997,40(7)

9 Manjikian N,Abdelrahman T S. Fusion of Loops for Parallelism and Locality. IEEE Transaction on Parallel and Distributed Systems,1997,8(2)

10 Abdelrahman T, et al. Locality Enhancement for Large-Scale Shared-Memory Multiprocessor. Languages, Compilers, and Run-Time Systems for Scalable Computers 4th International Workshop, LCR'98,LNCS 1511,Springer-Verlag, 1998

11 Lim A W, et al. Maximizing Parallelism and Minimizing Synchronization with Affine Partitions. Parallel Computing, 1998, 24(3-4):445～468

(上接第11页)

7 ReaCT-ILP laboratory. Trimaran: An Infrastructure for Research in Instuction-level Parallelism. http://www.trimaran.org

8 Gyllenhaal J C. A Machine Description Language for Compilation: [Master Thesis]. University of Illinois at Urbana-Champaign, 1994

9 Fraser C W,Hanson D R. A Retargetable C Compiler: Design and Implementation. Benjamin/ Cummings Pub Co, Redwood City, CA, USA, 1995

10 Gough J. Bottom up Tree Rewriting with MBURG: The MBURG Reference Manual. ftp.fit.qut.edu.au. 1995

11 Emmelmann H, Schroer F W, Landwehr R. BEG－A Generator for Efficient Back Ends. In:Proc. of the SIGPLAN'89 Conf. on Programming Language Design and Implementation, SIGPLAN Notices, 1989, 24(7): 227～237

12 Fraser C W,Hanson D R,Proebsting T A. Engineering a Simple, Efficient Code Generator Generator. ACM Letters on Programming Languages and Systems, 1992, 1(3): 213～226

13 Gao G R, Amaral J N, Dehnert J,Towle R. The SGI Pro64 Compiler Infrastructure: A tutorial. The International Conference on Parallel Architecture and Compilation Techniques (PACT2000), Oct. 2000

14 Ganapathi M,et al. Affix Grammar Driven Code Generation. ACM Transactions of Programming Languages and Systems, 1985, 7(4): 560～599

15 Moona R. Processor Models for Retargetable Tools. In: Proc. of 11th Intl. Workshop on Rapid System Prototyping, IEEE, 2000. 34～39

16 Wilson R P,et al. SUIF: An Infrastructure for Research on Parallelizing and Optimizing Compilers. ACM SIGPLAN Notices, 1994, 29(10): 31～37