利用 U 模变换增加并行粒度与改善数据访问局部性的方法

马国凯 王欣孃 王 鹏 臧斌宇 朱传琪

(复旦大学并行处理研究所 上海 200433)

摘 要 提出了一种利用循环变换增加循环并行粒度,改善循环数据访问局部性的方法.该方法利用了给定二重循环的相关向量集的某些性质,将外层循环变量不同而内层循环变量相等的若干次迭代合并,成为折叠后迭代空间的一个结点,并且保持内层循环的并行性不变,从而达到增加循环并行粒度的目的.对于更普遍的情况,该文讨论了如何根据给定循环的循环向量集,确定一个 U 模变换对迭代空间进行变换,达到内层循环可并行和扩大循环粒度两个目的.针对循环变换中数据访问局部性可能变差的问题,该文提出了对内层循环先合并,根据合并后的相关向量集变换迭代空间,以及折叠迭代空间的方法.该文的方法是 wavefront 循环并行化方法的一种扩展.

关键词 循环变换;并行化编译;U 模变换;迭代空间折叠;数据访问局部性;wavefront 方法

中图法分类号 TP303

Increase Parallel Granularity and Data Locality by Unimodular Metrics

MA Guo-Kai WANG Xin-Rang WANG Peng ZANG Bin-Yu ZHU Chuan-Qi

(*Institute of Parallel Processing*, *Fudan University*, *Shanghai* 200433)

Abstract In this paper we discussed a loop transformation method which would increase the granularity of the loop body and improve the data locality of the transformed loop: By analyzing the dependence vector set of the given nested double-loop, we could merge several nodes in the iteration space, which have same outer loop variable value and different inner loop variable value into one node in the folded iteration space, while preserving the parallelism of inner loop at the same time. Thus, we increased the granularity of the parallel loop body. Furthermore, we discussed how to find a unimodular metrics to transform the given iteration space with the given dependence into an iteration space in which iteration nodes could be merged using our methods given above. We also present a method to preserve the locality of the original loop while doing our loop transformation and iteration space folding. Our method discussed in this article is the generalization of the wavefront method. Compared with the wavefront method, our method can achieve higher performance due to larger granularity and better data locality. We apply our method to ygx, a program of the IAPCM Benchmark, to evaluate the effect the technique. The experiment data show that our method can spare the execution time by 22% compared with the wavefront method when the program is parallel processed by 4 CPU's on SGI origin 200 system which is a typical 4 CPU's SMP architecture.

Keywords loop transformation; parallelizing compilers; unimodular metrics; iteration space folding; data locality; wavefront method

---

收稿日期:2002-08-16;修改稿收到日期:2003-06-09.本课题得到国家自然科学基金(60273046)与上海市科学技术委员会重点基础项目基金(02JC14013)资助.**马国凯**,男,1976 年生,硕士,主要研究领域为体系结构和并行编译优化.**王欣孃**,女,1978 年生,硕士,主要研究领域为体系结构和并行编译优化.**王 鹏**,男,1979 年生,硕士,主要研究领域为体系结构和并行编译优化.E-mail:pengwang@fudan.edu.cn.**臧斌宇**,男,教授,博士生导师,主要研究领域为体系结构和编译优化.**朱传琪**,男,教授,博士生导师,主要研究领域为体系结构和编译优化.

1 引　言

在程序自动并行化的过程中,并行粒度和数据访问局部性是两个被关注的问题.程序的并行粒度越大,并行化后的程序花在进程同步与通信上面的开销就越小,程序的并行加速比就越高.而数据访问局部性也会影响并行程序的加速比,如果并行化后的数据访问局部性不佳,那么程序的加速比就不能达到预料的效果.

循环变换是常用的自动并行化方法,例如 wavefront 方法.然而 wavefront 方法也有并行粒度小、数据访问局部性不佳等缺点.这是因为 wavefront 方法产生的是内层可并行化的循环,因此外层循环每迭代一次,都要进行一次同步操作.当紧嵌套循环的循环体计算量较小,并且内层循环的循环次数较少的时候,程序的并行加速比会受到很大影响.同时,循环变换会将原来相邻迭代之间的距离拉远,如果原循环的数据访问局部性很好,那么变形后循环的数据访问局部性就会变得很差.

本文在 wavefront 方法的基础上进行了推广,利用循环变换的思想,提出了一种利用循环变换增加内层循环并行粒度的方法.它利用循环变换中常见的 U 模变换,通过对迭代空间进行一定的变形,使内层循环变量的值保持不变,而外层循环变量的值不相等的若干个相邻迭代可以合并成为内层循环的一次迭代,增加了变形后的迭代空间一次迭代的粒度,并且依然保持内层循环的并行性.本文进行了四方面的工作,首先是给出了能够做这种结点合并,迭代空间中相关向量集应该满足的条件(第 4 节);其次是对于给定的初始循环和 U 模变换,给出了变换后的循环形式和确定循环界限的方法(第 5 节);第三是对于一个给定的循环和相关向量集,找出一个 U 模变换,使得相关向量集变换成为满足条件的形式(第 6 节);最后,对于 U 模变换引起的数据访问局部性不佳的问题,本文提出了一个解决的方法(第 7 节).

Banerjee 在循环并行化和迭代空间的线性变换中做了许多研究[2,4],本文在他的研究基础上对他的迭代空间变换算法做了改进,使其能够适应本文方法中的非线性变换.Michael Wolfe 在其著作中[1]对数据相关性以及循环变换有十分详尽的叙述,本文中的记号除一小部分来自 Banerjee 的著作外,主要参考了该著作.文献[12]利用语句调度消除循环中的同步和数据交换,也能够增加循环的并行粒度,但是对循环中的语句相关性有一些特殊要求,而我们的方法是没有这种要求的.文献[3,5]从不同角度对增加循环粒度进行了一些讨论,但主要是从挖掘外层循环可并行性角度出发的.

2 前提假设、名词解释和记号

(1)对二重循环而言,为了化简问题,我们假设二重循环在并行化变形前循环界限是常数,表现为如下形式:

for $I=0$ to u_{11}
  for $J=0$ to u_{22}
    $\boldsymbol{H}$ $//\boldsymbol{H}\left(\begin{pmatrix} I \\ J \end{pmatrix}\right)$
  endfor
endfor

对于上述的程序我们做出如下解释,$\boldsymbol{H}$ 是一段程序,其中包含文字 I 也包含文字 J.在下文中出现的程序,我们都会在注释中给出其矩阵形式.

(2)循环界限可以用两个对角矩阵表示

$$\boldsymbol{L}=\begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix};\quad \boldsymbol{U}=\begin{pmatrix} u_{11} & 0 \\ 0 & u_{22} \end{pmatrix}.$$

(3)我们假设循环是递增的,步长为 1,如果循环本身不符合这个条件,可以通过改写循环来满足这个条件,于是我们得到关于循环界限的不等式:

$$u_{11}\geq 0,\quad u_{22}\geq 0.$$

(4)在并行化之前循环的相关性用相关向量集 $D=\{\boldsymbol{d}_1,\boldsymbol{d}_2,\cdots,\boldsymbol{d}_n\}$ 表示.

(5)对程序进行的变形可以用一个 U 模变换来描述,它的形式为

$$\boldsymbol{T}=\begin{pmatrix} t_{11} & t_{12} \\ t_{21} & t_{22} \end{pmatrix};\quad \boldsymbol{T}^{-1}=\Delta\begin{pmatrix} t_{22} & -t_{12} \\ -t_{21} & t_{11} \end{pmatrix}.$$

其中 $\Delta=|\boldsymbol{T}|=\pm 1$,且 $t_{ij}\in I$; $i,j\in\{1,2\}$.

(6)经过 $\boldsymbol{T}$ 的变换,循环的相关向量集变换成为 $\boldsymbol{T}D=\{\boldsymbol{T}\boldsymbol{d}_1,\boldsymbol{T}\boldsymbol{d}_2,\cdots,\boldsymbol{T}\boldsymbol{d}_n\}$,变换后的循环变量 $\begin{pmatrix} i_1 \\ i_2 \end{pmatrix}$ 和原循环变量 $\begin{pmatrix} I \\ J \end{pmatrix}$ 的关系为 $\begin{pmatrix} i_1 \\ i_2 \end{pmatrix}=\boldsymbol{T}\begin{pmatrix} I \\ J \end{pmatrix}$.

(7)我们定义关于 $\boldsymbol{H}$ 的两个置换函数 R 和 S:

$R(I,J,I',J',\boldsymbol{H})$ 是将 $\boldsymbol{H}$ 中的 I 置换为 I', J 置换为 J'.当 I 和 J 在 $\boldsymbol{H}$ 中不作为左值出现时, R 置换是一个合法的置换.(在 Fortran 语言中,这总

成立；在 C 语言中，这要求循环变量在循环体中不能被赋值.)

$S(I,J,\boldsymbol{T}(I,J),\boldsymbol{H})$定义为 $R(I,J,t_{11}\cdot I+t_{12}\cdot J,t_{21}\cdot I+t_{22}\cdot J,\boldsymbol{H})$，其矩阵形式为 $\boldsymbol{H}\left(\boldsymbol{T}\begin{pmatrix}I\\J\end{pmatrix}\right)$.

(8)对于相关向量集 $D=\{\boldsymbol{d}_1,\boldsymbol{d}_2,\cdots,\boldsymbol{d}_n\}$，我们定义 $SplitX$ 函数：

$$SplitX(D,a)=\left\{\begin{pmatrix}\boldsymbol{d}_1\\\boldsymbol{d}_2\end{pmatrix}\middle|\left(\boldsymbol{d}_1=\left\lfloor\frac{u_1}{a}\right\rfloor\vee\boldsymbol{d}_1=\left\lceil\frac{u_1}{a}\right\rceil\right)\wedge\begin{pmatrix}u_1\\\boldsymbol{d}_2\end{pmatrix}\in D\right\}.$$

(9) 数据访问局部性. 如果程序执行时有良好的 Cache 和虚拟页面命中率，我们称程序的数据访问局部性比较高；如果 Cache 不命中或者缺页的情况比较严重，那么程序执行的数据访问局部性就比较低.

3 通过对迭代空间的变形增加并行粒度

首先考虑一种特殊情况，$\boldsymbol{T}=\begin{pmatrix}1&0\\0&1\end{pmatrix}$，$\boldsymbol{T}D=D=\{(1,0),(1,1)\}$，如图 1 所示，其中 i_1 代表外层循环，i_2 代表内层循环. 这个二重循环的内层循环是可以并行化的，由于内层循环的迭代次数很少，循环并行化后的效率并不高. 在这种情况下我们可以对迭代空间进行变形以增加并行粒度. 如图所示，如果我们对经过 $\boldsymbol{T}$ 变形后的迭代空间再进行一次 U 模变换$\begin{pmatrix}1&1\\0&1\end{pmatrix}$，得到 $\boldsymbol{T}'=\begin{pmatrix}1&1\\0&1\end{pmatrix}$，然后将变换后的结点两两合并(见图 1(b)中的白色大结点)，会发现同一行中的白色结点可以并行执行.

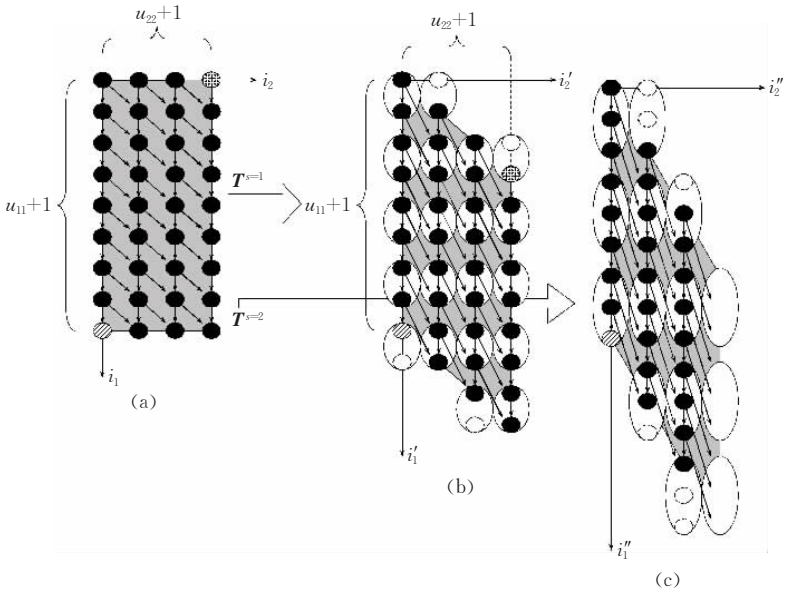


图 1

我们可以继续增加并行粒度，如图 1(c)所示，令$\boldsymbol{T}''=\begin{pmatrix}1&2\\0&1\end{pmatrix}$，我们可以将三个循环结点合并成一个循环结点. 在图 1(b)和(c)里面循环边界外有一些虚线的结点，这些虚线的结点实际上并不包含实际的运算，它们的存在只是为了方便结点合并.

可以把迭代空间想像成一个平面，而结点合并的过程就是迭代空间平面折叠的过程，几个不同的结点通过折叠映射成为一个结点，从而增加了迭代空间结点的粒度(循环并行粒度)，如图 2 所示.

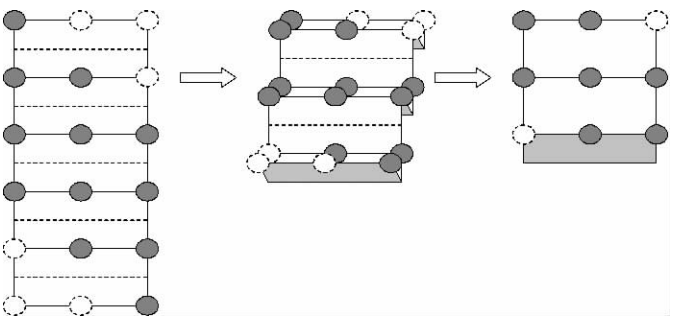


图 2

一般情况下增加并行粒度的的矩阵 $\boldsymbol{T}^s=\begin{pmatrix}1 & s\\ 0 & 1\end{pmatrix}$,通过 $\boldsymbol{T}^s$ 对循环进行变形后,可以将 $s+1$ 个结点合并成一个结点进行并行执行,经过变换后的循环如下所示:

```
for i1=0 to u11+u22·s, step s+1
  for i2=max(0,(i1-u11)/s) to min(u22,(i1+s)/s)
    if (0≤i1-s·i2≤u11. AND. 0≤i2≤u22)
      S(i1,i2,(T^s)^-1(i1,i2),H)
    endif
    if (0≤i1-s·i2+1≤u11. AND. 0≤i2≤u22)
      S(i1,i2,(T^s)^-1(i1+1,i2),H)
    endif
    …
    if (0≤i1-s·i2+s≤u11. AND. 0≤i2≤u22)
      S(i1,i2,(T^s)^-1(i1+s,i2),H)
    endif
  endfor
endfor
```

经过变形后的程序与原程序等价,并且内层循环可以并行化.循环粒度扩大为原来的 $s+1$ 倍.

4　结点合并的条件

在上面的例子中,我们注意到相关向量集{(1,0),(1,1)}经过变换矩阵$\begin{pmatrix}1 & s\\ 0 & 1\end{pmatrix}$变换后成为{(1,0),(1+$s$,1)},并且在最内层循环获得了大小为 $1+s$ 的并行粒度,这个条件可以一般化,如果我们在某个变换 $\boldsymbol{T}$ 后得到循环形式:

```
#Loop A
  for j1=m1 to M1
    for j2=m2(j1) to M2(j1)
      S(j1,j2,T^-1(j1,j2),H)
    endfor
  endfor
```

和相关向量集:

$\{(r_1,0)(r_2,0),\cdots(r_L,0),(p_1,q_1),(p_2,q_2),\cdots,(p_M,q_M),(s_1,-t_1),(s_2,-t_2),\cdots,(s_N,-t_N)\}$,$p_1,p_2,\cdots,p_M,q_1,q_2,\cdots,q_M,r_1,r_2,\cdots,r_L,s_1,s_2,\cdots,s_N,t_1,t_2,\cdots,t_N$为正整数.

那么我们可以对迭代空间进行折叠,把循环写成如下的形式:

```
#Loop B
for i1=m1 to M1 step w
  for i2=m2(i1) to M2(i1)
    S(i1,i2,T^-1(i1,i2),H)
    S(i1,i2,T^-1(i1+1,i2),H)
    …
    S(i1,i2,T^-1(i1+w-1,i2),H)
  endfor
endfor
```

其中 $0\le w\le\min\{p_1,p_2,\cdots,p_M,s_1,s_2,\cdots,s_N\}$,并且内层循环可以并行化.

证明.　任取 Loop B 中的两个迭代($i_1=x$, $i_2=y_1$)和($i_1=x$, $i_2=y_2$), $y_1\neq y_2$

在这两个迭代中各自任取一个语句

Stmt 1: $S(i_1,i_2,\boldsymbol{T}^{-1}(x+a,y_1),\boldsymbol{H})$和 Stmt 2: $S(i_1,i_2,\boldsymbol{T}^{-1}(x+b,y_2),\boldsymbol{H})$

由 Loop B 的形式我们可以知道$|b-a|<w\le\min\{p_1,p_2,\cdots,p_M,s_1,s_2,\cdots,s_N\}$

$\Theta y_1\neq y_2$,如果 Stmt 1 和 Stmt 2 之间存在相关性,那么根据 Loop A 和 Loop B 的关系,我们知道在$\{(p_1,q_1),(p_2,q_2),\cdots,(p_M,q_M),(s_1,-t_1),(s_2,-t_2),\cdots,(s_N,-t_N)\}$中至少存在一条相关向量对这个相关性是有贡献的.于是得到$|b-a|>=\min\{p_1,p_2,\cdots,p_M,s_1,s_2,\cdots,s_N\}$

然而$|b-a|<w\le\min\{p_1,p_2,\cdots,p_M,s_1,s_2,\cdots,s_N\}$,与上面的关系矛盾.

所以,Stmt 1 和 Stmt 2 之间不可能存在相关性,

Loop B 中迭代($i_1=x$, $i_2=y_1$)和($i_1=x$, $i_2=y_2$), $y_1\neq y_2$之间不存在相关性,

因此,Loop B 的内层循环可以并行化.　证毕.

如果 $gcd(r_1,r_2,\cdots,r_L,p_1,p_2,\cdots,p_M,s_1,s_2,\cdots,s_N)$为不等于 1 的正整数,那么我们可以把外层循环按照并行度为 $gcd(r,p,s)$进行并行[3].

5　一般 U 模变换情况

在第 6 节中,我们将讨论如何在已知原始相关向量集的情况下,通过 U 模变换得到一个循环粒度较大的内层可并行循环.在此之前,我们先讨论当我们通过某个 U 模变换 $\boldsymbol{T}$ 得到了相关向量集{(r_1,0),(r_2,0), …, (r_L,0), (p_1,q_1), (p_2,q_2), …, (p_M,q_M), (s_1,$-t_1$), (s_2,$-t_2$), …, (s_N,$-t_N$)}后,如何给出变形后的循环形式.一般而言,经过一个 U 模变换后,我们的迭代空间变换为为一个平行四边形,如图 3 所示.

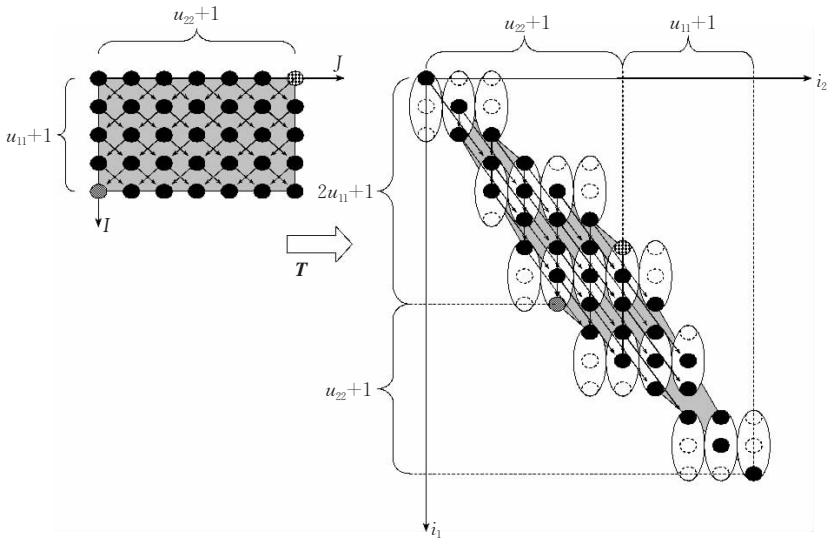


图 3

此时 $\boldsymbol{T}D=\{(r_1,0),(r_2,0),\cdots,(r_L,0),(p_1,q_1),(p_2,q_2),\cdots,(p_M,q_M),(s_1,-t_1),(s_2,-t_2),\cdots,(s_N,-t_N)\}$,在图 3 的例子中 $\boldsymbol{T}=\begin{pmatrix}2 & 1\\ 1 & 1\end{pmatrix}$, $\boldsymbol{T}D=\{(1,0),(3,2)\}$,根据第 4 节的结论,我们可以将循环改写成粒度为 3 的二重循环形式.我们通过修改后的 Banerjee 二维循环变形算法(参考文献[2]中的算法 4.1)得到变形后的循环.循环变形要解决的主要是循环界限问题.在 Banerjee 算法中给出了 U 模变换后的循环界限,我们的算法使用了这个结果.对于需要合并的若干行,我们取内层循环界限为各行循环界限的并,得到迭代空间折叠后的循环界限.而在内层循环的一次迭代中再去判断参加合并的各个结点(上图中白色大结点中的各个黑色小结点与虚线结点)是否是原始迭代空间中的结点(虚线结点不是原始迭代空间中的结点而黑色结点是).这样就可以得到结点合并后的循环形式:

$w=\min\{p_1,p_2,\cdots,p_M,s_1,s_2,\cdots,s_N\}$
for $i_1=m_1$ to M_1 step w
  for $i_2=m_2(i_1)$ to $M_2(i_1)$
    if $(1\le|\boldsymbol{T}|(t_{22}\cdot i_1-t_{12}\cdot i_2)\le u_{11}$. AND. $1\le|\boldsymbol{T}|(-t_{21}\cdot i_1+t_{11}\cdot i_2)\le u_{22})S(i_1,i_2,\boldsymbol{T}^{-1}(i_1,i_2),\boldsymbol{H})$
    endif
    if $(1\le|\boldsymbol{T}|(t_{22}\cdot i_1-t_{12}\cdot i_2+t_{22})\le u_{11}$. AND. $1\le|\boldsymbol{T}|(-t_{21}\cdot i_1+t_{11}\cdot i_2-t_{21})\le u_{22})S(i_1,i_2,\boldsymbol{T}^{-1}(i_1+1,i_2),\boldsymbol{H})$
    endif
    ...
    if $(1\le|\boldsymbol{T}|(t_{22}\cdot i_1-t_{12}\cdot i_2+(w-1)\cdot t_{22})\le u_{11}$. AND. $1\le|\boldsymbol{T}|(-t_{21}\cdot i_1+t_{11}\cdot i_2-(w-1)\cdot t_{21})\le u_{22})S(i_1,i_2,\boldsymbol{T}^{-1}(i_1+w-1,i_2),\boldsymbol{H})$
    endif
  endfor
endfor

6 对相关向量集 D 的讨论

我们来讨论一些特殊情况,在有些情况中 $D\neq\{(r_1,0),(r_2,0),\cdots,(r_L,0),(p_1,q_1),(p_2,q_2),\cdots,(p_M,q_M),(s_1,-t_1),(s_2,-t_2),\cdots,(s_N,-t_N)\}$,在另一些情况中并行粒度可以继续加大,我们通过 U 模变换把这些情况转换为第 5 节中讨论过的问题,并利用第 5 节中的算法进行求解.

我们称第 5 节中讨论的情况为情况 0,可以利用第 5 节中的方法进行求解.

(1) $D=\{(r,0),(1,q)\}$; r 为正整数,q 为非零整数,令 $\boldsymbol{T}=\begin{pmatrix}1 & s\\ 0 & 1\end{pmatrix}$,$s$ 的符号与 q 相同,且 s 的绝对值大于 0,我们得到 $\boldsymbol{T}D=\{(r,0),(1+qs,q)\}$,这样我们把问题化为情况 0 的形式,完毕.

(2) $D=\{(r,0),(0,q)\}$; r 为正整数,q 为非零整数,令 $\boldsymbol{T}=\begin{pmatrix}1 & s\\ 0 & 1\end{pmatrix}$,$s$ 的符号与 q 相同,且 qs 大于 1,于是我们得到 $\boldsymbol{T}D=\{(r,0),(qs,q)\}$,这样我们把问题化为情况 0 的形式,完毕.

可以看出在这种情况下,如果 $r=q=s=1$,问题实际上是 wavefront 并行化方法的一种情况,而我们的方法比 wavefront 方法更加一般化.

(3) $D=\{(r,0),(p,q),(s,-t)\}$; p,q,r,s,t 为正整数,且 $p-q>s$,令 $\boldsymbol{T}=\begin{pmatrix}1 & -1\\ 0 & 1\end{pmatrix}$,于是我们得到

$TD=\{(r,0),(p-q,q),(s+t,t)\}$,因为 $\min\{(p-q),(s+t)\}>\min\{p,s\}$,所以经过 T 变换后得到的并行粒度大于原来的并行粒度.我们可以重复执行这一步直到并行粒度无法再增加,或者并行粒度已经满足要求为止.

(4) $D=\{(r,0),(p,q),(s,-t)\}$; p,q,r,s,t 为正整数,且 $s-t>p$,令 $T=\begin{pmatrix}1 & 1\\ 0 & 1\end{pmatrix}$,于是我们得到 $TD=\{(r,0),(p+q,q),(s-t,-t)\}$,因为 $\min\{(p+q),(s-t)\}>\min\{p,s\}$,所以经过 T 变换后得到的并行粒度大于原来的并行粒度.我们可以重复执行这一步直到并行粒度无法再增加,或者并行粒度已经满足要求为止.

情况(1)~(4)都可以一般化.

根据上面的讨论,对于给定的相关向量集 TD,我们设法将其归入情况(1)~(4)中的一种,以获得较大的循环粒度,并将循环写成可并行化的形式.如果相关向量集 TD 无法归入这五种情况之一,那么或者外层循环可以并行化,或者循环无法并行化,或者无法再继续扩大循环的粒度.

7　U 模变换前通过结点合并增加数据访问的局部性

通过 U 模变换获得并行性虽然能带来扩大并行粒度的好处,但是也会对数据访问的局部性带来不利的影响.一般而言,原程序的内层循环对数据的访问是遵循数据相邻的原则的,这样在程序执行时一个 Cache 块或者一个虚拟页面中的数据可以被多次快速地访问,提高程序执行效率.然而在做过 U 模变换和外层循环的结点合并之后,原来相邻执行的迭代不再保持相邻关系,影响了数据访问的局部性,对程序执行的效率产生了负面的影响.为了消除这种不利的影响,我们可以在 U 模变换之前,对内层循环的结点进行适当的合并,这样数据访问的局部性便可以在以后的 U 模变换和结点合并中一直保持.

内层循环的结点合并是非线性变换,然而在循环界限是固定界限的情况下,程序的变换比较容易,给定初始的二重紧嵌套循环和内层循环的合并度 d:

```
for I=0 to u11
  for J=0 to u22
    H
  endfor
endfor
```

我们给出合并后的循环形式:

```
for I=0 to u11
  for J=0 to u22 step d
    for k=0 to min(d-1, u22-J)
      R(I,J,I,J+K,H)
    endfor
  endfor
endfor
```

合并后的相关向量集为 $D'=SplitX(D)$(见第 2 节中的定义).

我们把合并后的最内层循环看做外面两重循环的循环单元,把合并后的相关向量集作为新的相关向量集,在这个循环和相关向量集的基础上做 U 模变换和结点合并,便可以在保持数据访问局部性的基础上达到并行化和增加循环并行粒度的目的.

8　一般处理流程

我们给出一般情况下的循环并行化处理流程:

(1) 根据需要按照第 7 节中的方法合并内层结点,并得到合并后的相关向量集 D';

(2) 根据初始的相关向量集 D',利用第 6 节中的方法确定相应的 U 模变换 T,获得满意的并行粒度;

(3) 利用第 5 节中的算法得到变形后的循环形式;

(4) 将变形后的内层循环改写成并行循环.

完毕.

9　实验数据

我们通过 IAPCM Benchmark 中的一个程序来说明本文方法的效果.ygx 是 IAPCM Benchmark 中的一个程序,它的计算模式是迭代-收敛计算模式.在 ygx 中存在一些两层紧嵌套循环,可以通过 wavefront 方法对内层循环进行并行化,如图 4 所示.

```
DO I=1, M
DO J=0, P-1
  A(J,I)=A(J,I)+A(J-
    1, I)+A(J,I-1)+A(J,
    I+1) +A(J+1,I)
ENDDO
ENDDO
```

(a) 五点模式计算

```
DO II=2,M+N
DOALL J=max(1, II-M),
        min(N,II-1)
  I=II-J
  A(J,I)=A(J,I)+A(J-
    1,I)+A(J,I-1)+A(J,I
    +1)+A(J+1,I)
ENDDO
ENDDO
```

(b) 传统 wavefront 方法

图　4

在这种方法中,内层循环的迭代次数最多不超过 min(M, N). 在 ygx 中,外层循环的迭代次数,只有三四次,因此使用这种方法将得不偿失,因为并行循环本身的粒度太小. 为此我们用我们的方法对循环进行结点合并和并行化.

我们将原循环转换为我们的标准形式:

```
DO II=0, M-1
  DO JJ=0, P-1
    I=II+1
    J=JJ
    A(J, I)=A(J,I)+A(J-1,I)+A(J,I-1)+
    A(J,I+1)+A(J+1,I)
  ENDDO
ENDDO
```

此时的相关向量集为{(1,0), (0,1)}

由第 6 节的分析,此时为了达到并行粒度 s,采用 U 模矩阵$\begin{pmatrix}1 & s\\ 0 & 1\end{pmatrix}$对程序进行变形. 根据第 5 节中的分析,此时程序变形为:

```
DO II=0, M+S*P-(S+1), S
  DOALL JJ=max(-(M-1-II)/S, 0),
                min((II+S-1)/S, P-1)
    I=II-S*JJ+1
    J=JJ
    DO K=0, S-1
      A(J,I)=A(J,I)+A(J-1,I)+A(J,I-1)+
             A(J,I+1)+A(J+1,I)
      I++
    ENDDO
  ENDDO
ENDDO
```

简单的分析可以知道在这个例子中,第 5 节里变形后程序中的所有语句的判断是恒成立的,所以可以忽略.

我们设定合并度为 100,并在 Origin200 四机系统上比较结点合并前后,程序 ygx 并行加速比的变化如表 1 所示.

表 1 程序 ygx 并行加速比的变化

运行时间(s)							加速比						
使用方法	串行执行原程序	串行执行并行程序	单机并发	双机并行	三机并行	四机并行	使用方法	串行执行原程序	串行执行并行程序	单机并发	双机并行	三机并行	四机并行
结点合并		29.74	31.04	21.77	15.52	14.37	结点合并		1.04	0.99	1.42	1.99	2.16
不合并	30.88	30.84	30.66	24.76	18.92	18.81	不合并	1	1.00	1.01	1.25	1.63	1.64

四机并行时,能够结点合并的循环,其执行时间在结点合并前后分别为 4.76s 和 3.14s,分别占总执行时间的 22%和 25%,可以看到结点合并的效果是十分显著的.

10 小 结

本文的结点合并方法包含了 wavefront 方法的一种情况,是 wavefront 方法的一般化. 通过 U 模变换,可以同时达到程序并行化和结点合并两个目的,结点合并后对程序运行加速比的改善也有很大帮助. 在 wavefront 方法的某些情况下,循环的变换涉及到一种特殊的线性变换,其变换矩阵不再保持 U 模变换面积不变的性质. 这样的变换使得变换后的循环空间不再是致密的平行四边形. 对于这样的情况,本文没有进行讨论. 在后续的研究中,我们将把这种情况考虑进去,使得我们的方法更加完整.

参 考 文 献

1 Michael Wolfe. High Performance Compilers For Parallel Computing. The Addison-Wesley Publishing Company, California, 1995, 500

2 Utpal Banerjee. Unimodular transformations of double loops. In: Proceedings of Advances in Languages and Compilers for Parallel, Cambridge, Massachusetts, 1991, 192~219

3 Constantine D. Polychronopoulos. Compiler optimizations for enhancing parallelism and their impact on the architecture design. IEEE Transactions on Computers, 1988, 37(8):991~1004

4 Utpal Banerjee. Dependence Analysis for Supercomputing. Norwell: Kluwer Academic Publishers, 1988

5 Constantine D. Polychronopoulos, David J. Kuck, David A. Padua. Utilizing multidimensional loop parallelism on large-scale parallel processor systems. IEEE Transactions on Computers, 1989, 38(9):1285~1296

6 Allen, J. R., Kennedy, K.. Automatic loop interchange. In: Proceedings of the SIGPLAN'84 Symposium on Compiler Con-

struction, Montreal, Canada, 1984, 233～246

7 Banerjee, U.. Data dependence in ordinary programs[M. S. dissertation]. Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, Illinois, 1976,76-837

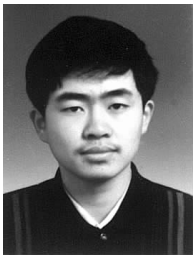
8 Irigoin F.., Triolet R.. Dependence approximation and global parallel code generation for nested loops. In:Cosnard M. *et al*. Eds. Parallel and Distributed Algorithms, Elsvier, North-Holland, New York, 1989, 297～308

9 Wolfe M. J.. Loop skewing: The wavefront method revisited. International Journal of Parallel Programming, 1986,15(4): 279～293

10 Kathryn S. McKinley, Steve Carr, Chau-Wen Tseng. Improving data locality with loop transformations. CAN Transactions on Programming Languages and Systems (TOPLAS), 1996,18(4): 424～453

11 Song Yong-Hong, Li Zhi-Yuan. New tiling techniques to improve cache temporal locality. In: Proceedings of the ACM SIGPLAN'99 Conference on Programming Language Design and Implementation, Atlanta, Georgia, United States, 1999, 215～228

12 Eduard Ayguade, Jordi Torres. Partitioning the statement per iteration space using non-singular matrices. In: Proceedings of the 1993 international conference on Supercomputing, Tokyo, 1993,407～415



MA Guo-Kai, born in 1976, master. His main research interests include computer architecture and parallel compiling optimization.

WANG Xin-Rang, born in 1978, master. Her main research interests include computer architecture and parallel compiling optimization.

WANG Peng, born in 1979, master. His main research interests include computer architecture and parallel compiling optimization.

ZANG Bin-Yu, professor, Ph. D. supervisor. His main research interests include computer architecture and compiling optimization.

ZHU Chuan-Qi, professor, Ph. D. supervisor. His main research interests include computer architecture and compiling optimization.

Background

The research focuses on increasing the granularity of the loop body and improving the data locality of the transformed loop to enhance the parallel processing performance. This topic has been researched by some investigators and several applicable methods have been developed. Eduard Ayguade and Jordi Torres[12] brought forward certain statement scheduling method to minimize the synchronization and data exchange in nested loops and then increase the granularity of loop body. But this method puts strict constraints on statements of the loop body. Constantine D. Polychronopoulos, David J. Kuck and David A. Padua[3,5] discussed how to increase the granularity of the loop body from the viewpoint of parallelizing outer loops. We propose a loop transformation technique based on the wavefront method to achieve inner loop parallelization and increase the granularity of loop body parallelized. We also present a method to preserve the locality of the original loop while doing our loop transformation.