1000-9825/2002/13(07)1193-06 ©2002 Journal of Software 软 件 学 报 Vol.13, No.7

Mapping of Nested Loops to Multiprocessors*

YIN Xin-chun[1,2], CHEN Ling[1,2], XIE Li[1]

[1](*State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing* 210093, *China*);

[2](*Department of Computer Science and Engineering, Yangzhou University, Yangzhou* 225009, *China*)

E-mail: xcyin@dislab.nju.edu.cn; xcyin@yzu.edu.cn

http://www.nju.edu.cn

Received May 10, 2001; accepted January 11, 2002

Abstract: Two new methods for partitioning and mapping nested loops with non-constant dependencies into distributed memory multiprocessors are presented. By partitioning the dependencies vectors or using direction vectors, the methods can partition the loops with non-constant dependencies into independent parts without any mutual dependencies. These parts can be processed independently so as to be mapped into multiprocessors and be executed in parallel.

Key words: nested loops; partition; multiprocessor

One of the major tasks of the parallel compilers in the multiprocessors is to partition the sequential algorithm into several independent parts and then assign them onto different processors for parallel processing. In the algorithms of most numerical and non-numerical problems, nested loop structure consumes most of the computing time.

Let $i_1, i_2,\ldots, i_m$ are the loop variables and l_j, u_j are the limits of loops, we call the set of integer vectors $I^m=\{(i_1, i_2,\ldots, i_m)^T|\ l_j \le i_j \le u_j,\ 1 \le j \le m\}$ the index space of the nested loop. I^m of a depth m nested loop is a subset of Z^m, here Z is the set of integers. We use m-dimension vector $v=(i_1, i_2,\ldots, i_m)^T$ to denote one step of recurrence computation, and the inner most loop body can be written as follows:

$$s(v)=F\ [s(H_1v+h_1), s(H_2v+h_2),\ldots, s(H_mv+h_m)], \tag{1}$$

here F is a given function, H_i and h_i are $m\times m$ matrix and $m\times 1$ vector respectively, s is an m-dimensional data array. We denote the loop body statement (1) as $S(v)$, which is called a statement instance. Vector $d_i=(H_iv+h_i)-v=(H_i-I)v+h_i$ (i=1,2,…,m) is called dependency vector.

Let P_i be a subset of I^m, if for every statement instance $S(v)$ in P_i, all statement instance having dependency relations with $S(v)$, including which depend on $S(v)$ and are depended by $S(v)$, are all in P_i, we call P_i an dependency-free part in I^m. Particularly, I^m itself is a dependency-free part. It is possible that a dependency-free part can be further partitioned into several smaller dependency-free parts. An independent partition of I^m is to partition it into several dependency-free parts. In an independent partition of I^m, the number of dependency-free parts r is just the number of the processors used in parallel computation.

* Supported by the National Natural Science Foundation of China under Grant No.60074013 (国家自然科学基金); the 333 Project Foundation of Jiangsu Province of China under Grant No.20018 (江苏省 333 工程基金)

YIN Xin-chun was born in 1962. He is currently an associate professor of Yangzhou University and a Ph.D. candidate at the Nanjing University. His main research interests include parallel computing, distributed processing. **CHEN Ling** was born in 1951. He is a professor of Yangzhou University. His research interests include parallel computing, distributed processing. **XIE Li** was born in 1942. He is a professor and doctoral supervisor of Nanjing University. His research interests include distributed system, distributed database and parallel processing.

There already exist some methods of independent partition for nested loop. For instance: partition vector method by Shang and Fortes[1], unimodular method by D'Hollander[2] and other methods[3~6]. But they are all for nested loops with constant dependencies. Zhang and Chen[7] presented several methods for partitioning nonuniform linear recurrence(NLR) into multiprocessors, but their methods lack generality and efficiency. In this paper, we extend their work to more general cases, and present two new methods for partitioning and mapping nested loops with non-constant dependencies into distributed memory multiprocessors. By partitioning the dependency vectors or using direction vectors, our methods can get more independent parts than the method of Ref.[7].

1 Method Using Dependency Vector

We divide the dependency vector $d_i=(H_iv+h_i)-v=(H_i-I)v+h_i$ into two parts: one is $(H_i-I)v$ in which variable v is involved, and the other is h_i which is just a constant vector. We use the constant vectors of the two parts as column vectors to form a $m\times w$ matrix B as follows:

$$B=[H_1-I, H_2-I,\dots, H_m-I, h_1, h_2,\dots, h_m]. \tag{2}$$

Here $w=m\times(m+1)$. Suppose rank of B is r, and a base of B's column space is $r_1, r_2,\dots, r_r$. If every column of B can be linearly expressed by the base $r_1, r_2,\dots, r_r$ with integer coefficients , we call vectors $r_1, r_2,\dots, r_r$ a set of integer base of B.

Theorem 1. In nested loop (1), let B in (2) be with rank of $r(r\le m)$, v_0 be an element in I^m. If $m\times1$ integer vectors $r_1,r_2,\dots,r_r$ form an integer base of B's column space, then set $P=\{v|v=v_0+\sum_{i=1}^{r} l_i r_i,\ v\in I^m,\ l_i\in Z\}$ is a dependency-free part of I^m.

Proof. Suppose v is an element of P, then there must exist integers $l_1, l_2,\dots, l_r$, so that $v=v_0+\sum_{i=1}^{r} l_i r_i=v_0+[r_1, r_2,\dots,r_r](l_1,l_2,\dots,l_r)^T=v_0+Rl$, here $R=[r_1,r_2,\dots,r_r]$ is an $m\times r$ matrix, $l=(l_1,l_2,\dots,l_r)^T$ is an $r\times1$ vector. Since $r_1,r_2,\dots,r_r$ is a set of integer base of B's column space and every column of H_i-I and h_i are columns of B, we can rewrite H_i-I and h_i as RW_i and Rw_i respectively $(i=1,2,\dots,m)$, here W_i is an $r\times m$ integer matrix and w_i is an $r\times1$ integer vector. Thus we have

$$H_iv+h_i=(RW_i+I)(v_0+Rl)+Rw_i=RW_iv_0+v_0+RW_iRl+Rl+Rw_i=v_0+R(W_iv_0+W_iRl+l+w_i).$$

Since all elements of R, W_i, l, w_i and v_0 are integers, vector $W_iv_0+W_iRl+l+w_i$ is an integer vector. Denote this integer vector as $y=(y_1, y_2,\dots, y_r)^T$, then $H_iv+h_i=v_0+Ry=v_0+\sum_{j=1}^{r} y_j r_j$, therefore $H_iv+h_i\in P$.

Conversely, if there exists an element $v_1\notin P$. Let set $P_1=\{v|v=v_1+\sum_{i=1}^{r} k_i r_i$, every k_i is an integer$\}$, then $P\cap P_1=\varnothing$. The reason is: if there exists a vector v which belongs to both P and P_1, then $v=v_0+\sum_{j=1}^{r} l_j r_j$, and $v=v_1+\sum_{j=1}^{r} k_j r_j$, here all l_j and k_j are integers. We have $v_1=v_0+\sum_{j=1}^{r} l_j r_j-\sum_{j=1}^{r} k_j r_j=v_0+\sum_{j=1}^{r}(l_j-k_j)r_j$, this is in contradiction with the fact $v_1\notin P$. Since $v_1\in P_1$, it is easy to proof that $H_iv_1+h_i\in P_1(i=1,2,\dots,m)$, and hence $H_iv_1+h_i\notin P$. Therefore P is a dependency-free part of I^m.

For set $P=\{v|\ v=v_0+\sum_{i=1}^{r} l_i r_i,\ v\in I^m,\ l_i\in Z\}$, we call the index v_0 the start point of P. For a set P, the start point is not unique. In fact, every element in P can be treated as the start point of P.

From Theorem 1 we know that to find the independent partition for nested loop (1), first we must find a set of B's integer base $r_1,r_2,\dots,r_r$ and form the matrix R. Start points $v_1,v_2,\dots,v_p$ are used to form dependency-free parts of

I^m: P_i={v| v= v_i+ $\sum_{j=1}^{r} l_j r_j$, l_j is integer} and $P_1,P_2,\ldots,P_p$ form an independent partition of I^m, here p is the number of processors used in parallel processing.

To find the start points $v_1,v_2,\ldots,v_p$ and the value of p, we use the matrix R. Suppose rank(R)=m, first R is transformed into an upper triangular matrix R^t, so that $R^t=RK$, here $R^t=[r_{ij}']$ is a lower triangular matrix with integer elements and K is a nonsingular matrix with integer elements. Let $p=r_{11}'\times r_{22}'\times\ldots\times r_{mm}'$, here $r_{11}',r_{22}',\ldots,r_{mm}'$ are diagonal elements of R^t .Then I^m can be partitioned into p independent sets.

To transform R into an upper triangular matrix R^t, simple elimination methods do not suffice, because each dependency vector in R must be covered by the new dependency vectors in R^t .

To find such R^t, we first find r_{11}'=GCD($r_{11},r_{12},\ldots,r_{1m}$). Since $r_{11},r_{12},\ldots,r_{1m}$ are not all zeros, find a nontrivial solution $(k_{11},k_{21},\ldots,k_{m1})\in Z^m$ so that

$$r_{11}k_{11}+r_{12}k_{21}+\ldots+r_{1m}k_{m1}=\mathrm{GCD}(r_{11},r_{12},\ldots,r_{1m})=r_{11}'.$$

Then, for all integers i such that $2 \le I \le m$, find the matrix K that minimizes

$$r_{ii}'=\sum_{j=1}^{m} r_{ij}k_{ji} \quad (i=2, 3,\ldots, m)$$

subject to $\sum_{j=1}^{m} r_{ij}k_{jl}$ =0 (l=1,2,…, i–1) and $\sum_{j=1}^{m} r_{ij}k_{ji}$ >0.

This is an integer programming problem, which is NP-complete. However, the number of variables and the number of constrains m are usually very small. The computation time consumed is not quite large.

After the diagonal elements of R^t and all elements of K are computed, we can determine the off-diagonal elements of R^t by $R^t=R\times K$.

If rank(R)=$r<m$, the number of columns in R is also m. In this case, we first obtain the first r columns of R^t following the way illustrated above. The rest m-r columns of R^t, denoted by $r^t_{*r+1}, r^t_{*r+2},\ldots, r^t_{*m}$, can be determined as: $r^t_{*j}=(0,\ldots,0,s_j,0,\ldots,0)^T$ in which the j-th element is $s_j=u_j-l_j$, the range of the j-th dimension of the nested loop.

Using diagonal elements of R^t, we can determine the start points. We select an arbitrary element $v^{(0)}=(v_1^{(0)}, v_2^{(0)},\ldots, v_m^{(0)})$ in I^m, then the set of start points is the Cartesian product of the sets $\{v_j^{(0)}, v_j^{(0)}+1,\ldots, v_j^{(0)}+r_{jj}'-1\}$ for j=1, 2,…,m. For convenience, we can take $v^{(0)}$=(0,0,…, 0). In this case, suppose a start point $v=(v_1,v_2,\ldots,v_m)$, the arrange of v_i is $[0,r_{ii}'-1]$. For instance, let $R^t=\begin{bmatrix}1 & 0\\ 5 & 7\end{bmatrix}$, take v=(0, 0), then the set of start points is {0}×{0,1,2,3,4,5, 6}={(0,0), (0,1), (0,2), (0,3), (0,4), (0,5), (0,6)}.

Theorem 2. Each $w\in I^m$ belongs to exactly only one partitioned set P_i.

Proof. First we prove for every $w\in I^m$, there exist unique integer vectors $l=(l_1,l_2,\ldots,l_m)^T$ and $v=(v_1,v_2,\ldots,v_m)^T$ in which $v_j\in[0, r_{jj}']$, so that w can be expressed as $w=v+\sum_{j=1}^{r} l_j r'_{*j}$, here r'_{*j} is the j-th column of R^t.

We prove it by induction. Let $w=(w_1,w_2,\ldots,w_n)^T$, since w= $v+\sum_{j=1}^{r} l_j r'_{*j}$, we have

$$w_1=v_1+l_1r_{11}'$$
$$w_2=v_2+l_1r_{21}'+l_2r_{22}'$$
$$\ldots\ldots$$
$$w_m=v_m+l_1r_{m1}'+l_2r_{m2}'+\ldots+l_mr_{mn}'$$

When j=1, we prove l_i and v_1 above are unique. Since w_1 , $r_{11}'\in Z$ and $r_{11}'\neq 0$, then there exist unique integers q and t so that $w_1=t+qr_{11}'$, here $t\in[0,r_{11}']$. Because $v_1\in[0,r_{11}']$, we have $v_1=t$ and $l_1=q$. Therefore, v_1 and l_1 are uniquely determined.

Assume that for w_j , there exist unique integers $l_1, l_2,\ldots, l_i$, so that $w_j=v_j+l_1r_{j1}'+l_2r_{j2}'+\ldots+l_jr_{jj}'$. Let

$$w_{j+1}=t+l_1r_{j+1,1}'+l_2r_{j+1,2}'+\ldots+l_jr_{j+1,j}'+q\,r_{j+1,j+1}'$$

here $t\in[0, r_{j+1,j+1}']$. That is

$$w_{j+1}-(l_1r_{j+1,1}'+l_2r_{j+1,2}'+\ldots+l_jr_{j+1,j}')=t+q\,r_{j+1,j+1}'$$

Using the same way as in the case of $j=1$, we can prove that there must be unique l_{j+1} and v_{j+1} to satisfy the equation above.

By induction, we conclude that for every $w\in I^m$, there exist unique integers $l_1, l_2,\ldots, l_m$ and vector v so that $w=v+\sum_{j=1}^{r} l_j r'_{*j}$. We rewrite it into the form of matrix: $w=v+R^t l$. Since $R^t=RK$, therefore $w=v+RKl$. Since K and l are all uniquely determined, vector Kl is also uniquely determined. We denote vector Kl as $u=(u_1,u_2,\ldots,u_r)^T$, then $w=v+Ru=v+\sum_{i=1}^{r} u_i r_i$. This means w belongs to exactly only one partitioned set.

Example 1. In nested loop

$$s(i, j)=F[s(2i+3j-1, 2i+2j-2), s(4i+j+3, i+3j+1), s(2i+j+1,2i+3j+2)], \tag{3}$$

$$H_1=\begin{bmatrix}2 & 3\\ 2 & 2\end{bmatrix}, H_2=\begin{bmatrix}4 & 1\\ 1 & 3\end{bmatrix}, H_3=\begin{bmatrix}2 & 1\\ 2 & 3\end{bmatrix}, h_1=\begin{bmatrix}-1\\ -2\end{bmatrix}, h_2=\begin{bmatrix}3\\ 1\end{bmatrix}, h_3=\begin{bmatrix}1\\ 2\end{bmatrix},$$

$$B=\begin{bmatrix}1 & 3 & 3 & 1 & 1 & 1 & -1 & 3 & 1\\ 2 & 1 & 1 & 2 & 2 & 2 & -2 & 1 & 2\end{bmatrix}$$

It can easily be seen that $r_1=\begin{bmatrix}1\\ 2\end{bmatrix}$, $r_2=\begin{bmatrix}3\\ 1\end{bmatrix}$ form a integer base of B. The base dependency matrix is $R=\begin{bmatrix}1 & 3\\ 2 & 1\end{bmatrix}$. Then R is transformed into an lower triangular matrix $R^t=\begin{bmatrix}1 & 0\\ -3 & 5\end{bmatrix}$, so that $R^t=RK$, here K is an inversable matrix with integer elements:

$$K=\begin{bmatrix}-2 & 3\\ 1 & -1\end{bmatrix}$$

Therefore, the index space of nested loop (3) I^2 can be partitioned into 5 dependency-free parts. The set of start points is $\{v_k|v_k=(0,k)\ (k=0,1,2,3,4)\}$, then $P_k=\{(i,j)|(i,j)=(0,k)+l_1(1,2)+l_2(3,1), l_1,l_2\in Z\}$ $(k=0,1,2,3,4)$ are dependency-free parts of I^2.

Figure 1 shows the dependency-free part $P_0=\{v|v=(0,0)+l_1(1,2)+l_2(3,1),\ l_1, l_2\in Z\}$ with start point (0, 0).

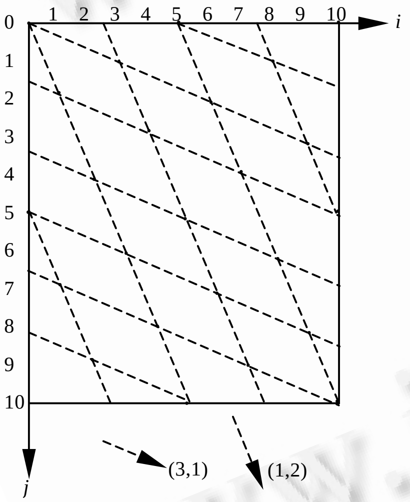

Fig.1

2 Method Using Direction Vector

Theorem 3. Suppose P_1 and P_2 are two dependency-free parts in I^m and set $P=P_1\cap P_2$ is not empty, then P is also a dependency-free part in I^m.

Proof. Suppose an index vector $v\in P$, then $v\in P_1$ and $v\in P_2$. Since P_1 and P_2 are two dependency-free parts in I^m, we have $H_iv+h_i\in P_1$ and $H_iv+h_i\in P_2$ $(i=1,2,\ldots,m)$. Therefore $H_iv+h_i\in P$. Conversely, suppose $v\notin P$, then $v\notin P_1$ or $v\notin P_2$. If $v\notin P_1$, then $H_iv+h_i\notin P_1$, thus $H_iv+h_i\notin P$. Similarly, it is easy to know that if $v\notin P_2$, then $H_iv+h_i\notin P$. Therefore, $P=P_1\cap P_2$ is a dependency-free part of I^m.

Theorem 4. Suppose for $i=1,2,\ldots,m$, H_i has eigenvalues 1 or –1, and eigenvectors $x_1^T, x_2^T,\ldots, x_r^T$. Each x_j^T

satisfies one of the following conditions: (i) x_j^T belongs to the eigenvalue 1 of H_i and $x_j^T h_i=0$; (ii) x_j^T belongs to the eigenvalue –1 of H_i and $x_j^T h_i=t_j$.

Let $p_j=\min_{v\in I^m}\{\min(x_j^T v, t_j-x_j^T v)\}$, $q_j=\max_{v\in I^m}\{\max(x_j^T v, t_j-x_j^T v)\}$, and $w_j=q_j-p_j+1$. We denote the consecutive integers from p_j to q_j as $c_{1j}, c_{2j}, \ldots, c_{w_j j}$, and let $P_{ij}=\{v|\ x_j^T v=c_{ij}\}\cup\{v|\ x_j^T v=t_j-c_{ij}\}$. Then set $\bigcap_{j=1}^{r} P_{i(j),j}$ is a dependency-free part of $I^m(i(j)=1,2,\ldots,w_j)$. Here we call $x_j\ (j=1,2,\ldots,r)$ the direction vectors of the partition.

Proof. From Theorem 2 of Ref.[7], we know that for every $j=1,2,\ldots,r$ and $i(j)=1,2,\ldots,w_j, P_{i(j),j}$ is a dependency-free part. By Theorem 3 it can be easily seen that $\bigcap_{j=1}^{r} P_{i(j),j}$ is a dependency-free part of $I^m\ (i(j)=1,2,\ldots,w_j)$.

From Theorem 4, we know that if nested loops satisfy the conditions of Theorem 4, I^m can be partitioned into $\bigcap_{j=1}^{r} w_j$ dependency-free parts. To test if nested loops satisfy the condition of Theorem 4, we can construct a matrix B with H_i-I or H_i+I and $h_i(i=1,2,\ldots,m)$ as its columns, and then find the zero vectors of B's column space. If H_i satisfies condition (i) in Theorem 4, columns of H_i-I and vector h_i have to be included in matrix B. If H_i satisfies condition (ii), columns of H_i+I should be included in B.

Example 2. In nested loop

$$s(i, j)=F[s(2-j,\ i-2j+1),\ s(2i+2,\ i+j+2)] \tag{4}$$

$$H_1=\begin{bmatrix}0 & -1\\ 1 & -2\end{bmatrix},\ H_2=\begin{bmatrix}2 & 0\\ 1 & 1\end{bmatrix},\ h_1=\begin{bmatrix}2\\ 1\end{bmatrix},\ h_2=\begin{bmatrix}2\\ 2\end{bmatrix},\ B=[H_1+I,\ H_2-I,\ h_2]=\begin{bmatrix}1 & -1 & 1 & 0 & 2\\ 1 & -1 & 1 & 0 & 2\end{bmatrix}$$

Since rank(B)=1, we choose zero vector: $x^T=(1,-1)$ of B's column space as the direction vector. Since $t=x^T h_1=1$, for an integer c, $P=\{(i,j)|i-j=c\}\cup\{(i,j)|i-j=1-c\}$ forms a dependency-free part of I^2. Figure 2 shows the dependency-free parts of nested loop (4). Two lines connected by a dotted curve form a dependency-free part.

Example 3. In nested loop

$$s(i,j)=F[s(2j-i+2, 2j-i+1), s(3i-2, i+j-1)], \tag{5}$$

$$H_1=\begin{bmatrix}-1 & 2\\ -1 & 2\end{bmatrix},\ H_2=\begin{bmatrix}3 & 0\\ 1 & 1\end{bmatrix},\ h_1=\begin{bmatrix}2\\ 1\end{bmatrix},\ h_2=\begin{bmatrix}-2\\ -1\end{bmatrix},\ B=[H_1\text{-}I,\ H_2\text{-}I,\ h_1,\ h_2]=\begin{bmatrix}-2 & 2 & 2 & 0 & 2 & -2\\ -1 & 1 & 1 & 0 & 1 & -1\end{bmatrix}.$$

Since rank(B)=1. We choose its zero vector $x^T=(1,-2)$ as the direction vector. Let c_k be an integer, set $P_k=\{(i,j)|\ i-2j=c_k\}$ forms a dependency-free part of I^2. Let integers $c_1=\min_{(i,j)\in J^2}(i-2j)$, $c_p=\max_{(i,j)\in J^2}(i-2j)$, $c_i=c_1+i-1(i=1,2,\ldots,p)$. The index space of nested loop (5) can be partitioned into p dependency-free parts $P_1, P_2, \ldots, P_p$ which can be computed in parallel using p processors. The dependency-free parts of nested loop (5) are shown in Fig.3 in which every line is a dependency-free part.

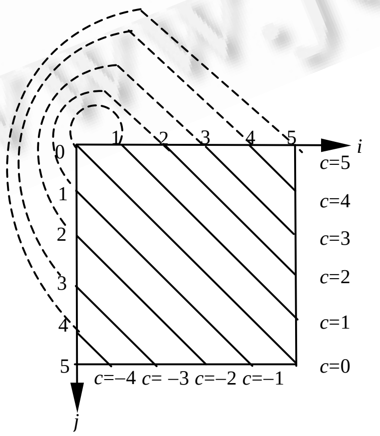

Fig.2

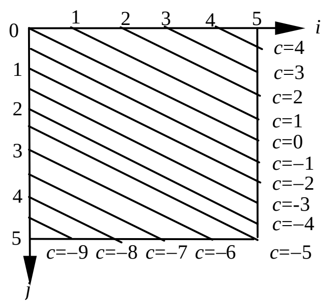

Fig.3

For a nested loop, both methods using Theorem 1 and Theorem 4 are all based on matrix $B=[H_1-I,H_2-I,\ldots,H_m-I,h_1,h_2,\ldots,h_m]$. But the range for applying Theorem 1 is larger than that of Theorem 4. For instance, in nested loop (3) of Example 1, since rank of B is 2, its column space has no zero vector, Theorem 4 can not be applied, but by using Theorem 1 it can be partitioned into 5 dependency-free parts. For the same nested loop, the number of dependency-free parts obtained by using Theorem 1 may be larger then that by using Theorem 4.

Example 4. In nested loop

$$s(i, j)=F[s(3i-2, j-2i+2)]$$

$$H=\begin{bmatrix} 3 & 0 \\ -2 & 1 \end{bmatrix}, h=\begin{bmatrix} -2 \\ 2 \end{bmatrix}, B=[H-I, h]=\begin{bmatrix} 2 & 0 & -2 \\ -2 & 0 & 2 \end{bmatrix}$$

If Theorem 4 is used, zero vectors of B's column space is $x^T=(1,1)$. For an integer c, subset of I^2: $P=\{(i, j) \mid i+j=c\}$ is a dependency-free part. For example, (1,0) and (0,1) are two elements of I^2, they both belong to the dependency-free part $P_0=\{(i, j) \mid i+j=1\}$.

If Theorem 1 is used, we find (2,–2) as the integer base of B's column space. For a start point $x_0=(i_0, j_0)\in I^2$, subset of I^2: $P=\{(i,j) \mid (i,j)=(i_0,j_0)+k(2,-2)$, k is an integer$\}$ is a dependency-free part. (1,0) belongs to dependency-free part $P_1=\{(i,j) \mid (i,j)=(1,0)+k(2,-2)$, k is an integer$\}$ and (0,1) belongs to dependency-free part $P_2=\{(i,j) \mid (i,j)=(0,1)+k(2,-2)$, k is an integer$\}$. They belong to two different dependency-free parts. In fact, dependency-free part obtained by Theorem 1 is $P_0=\{(i, j) \mid (i, j)=(0,1)+k(1,-1)$, k is an integer$\}=P_1\cup P_2$.

3 Conclusions

Two new methods for partitioning and mapping nested loops with non-constant dependencies into distributed memory multiprocessors are presented. Our methods partition the nested loops into independent parts without any mutual dependencies. These parts can be computed independently so as to be allocated to multiprocessors and be executed in parallel.

References:

[1] Shang, W., Fortes, J.A.B. Independent partitioning of algorithms with uniform dependencies. IEEE Transactions on Computers, 1992,41(2):190~206.
[2] D'Hollander, E.H. Partitioning and labeling of loops by unimodular transformations. IEEE Transactions on Parallel and Distributed Systems, 1992,3(4):465~476.
[3] Lee, Pei-Zong. Techniques for compiling programs on distributed memory multiprocessors. Parallel Computing, 1995,21(12): 1895~1923.
[4] Johnson, S.P., Ierotheou, C.S., Cross, M. Automatic parallel code generation for storage passing on distributed memory system. Parallel Computing, 1996,22(2):227~258.
[5] Ali, Adnan. Paralleling of relational inquires in indexed data structures. Industrial Simulation, 1998,15(6):769~777.
[6] Huang, K.C., Wang, F.J., Tsai, J.H. Two designing patterns for data parallel computations based on master-slave model. Information Processing Letters, 1999,70(4):197~204.
[7] Zhang, De-fu, Chen, Ling. Partitioning nonuniform linear recurrence onto multiprocessors. Chinese Journal of Computers, 1998, 21(supl):46~51 (in Chinese).

附中文参考文献:

[7] 张德富,陈崚.非一致线性递推计算到多处理机上的分解.计算机学报,1998,21(增刊):46~51.

嵌套循环到多处理机的映射

殷新春[1,2], 陈 崚[1,2], 谢 立[1]

[1](南京大学 计算机软件新技术国家重点实验室,江苏 南京 210093);

[2](扬州大学 计算机科学与工程系,江苏 扬州 225009)

摘要: 给出了将具有变相关的嵌套循环映射到具有分布式存储的多处理机上的两种方法.通过相关向量的分解或由相关向量导入方向向量,可将具有变相关的嵌套循环分解成若干互相没有相关关系的独立部分.由于它们可以被独立地执行,从而可以被映射到各个处理机上并行处理.

关键词: 嵌套循环;分解;多处理机
中图法分类号: TP311 文献标识码: A