Applications of High Performance Computing in Engineering VI, C.A. Brebbia, M. Ingber & H. Power (Editors)

Partitioning and mapping nested loops on multicomputers

Tzung-Shi Chen[1] & Jang-Ping Sheu[2]
[1]*Department of Information Management, Chang Jung University, Taiwan*
[2]*Department of Computer Science and Information Engineering, National Central University, Taiwan*

Abstract

Minimizing interprocessor communication is the key to a parallelized program running on multicomputers. This paper addresses a compilation technique to achieve the goal of generating an efficient parallelized code with both reducing the incurred communication cost and preserving parallelism. First, we transform a nested loop into a transformed structure with supporting an evaluation function to evaluate these transformed structures to obtain a certain parallelized code with less parallel executing time. Next, a mapping strategy is proposed to map the transformed structure onto hypercubes to be executed in parallel in a way with workload balance and low communication cost over processors.

1 Introduction

In order to minimize the amount of communication overhead, several researchers try to partition iterations of a nested loop into independent blocks so that it is executed in a communication-free execution manner [2, 5]. However, in practice, many programs cannot be partitioned into independent blocks by applying their partitioning methods. Thus, these programs will be executed sequentially; other techniques are needed to reduce communication overhead and exploit the parallelism.

One of important compiling techniques, *tiling* or called *supernode partitioning* [1, 3, 4, 10, 11], has been proposed to reduce the communication overhead for distributed memory multicomputers and to improve cache performance [6]. These approaches are to tile the related iterations of a nested loop together so that the total amount of message startup time can be re-

duced. Although communication cost is reduced by the above techniques, they may not exploit all the potential parallelism existing in nested loops and not efficiently map the nested loops onto a real multicomputer with a specific topology.

In this paper, our approach proposed here tries both to reduce the amount of communication overhead and exploit the parallelism of programs running on hypercubes. The focus and main contribution in this paper are on automatically partitioning and mapping nested loops with constant data dependences [10] onto hypercube multicomputers. Nested loops with constant data dependences occur frequently in digital signal processing and scientific computing applications. First, we analyze the relations of data dependences in an n-nested loop represented as an n-dimensional *computational structure*. It can then be projected into a k-dimensional hyperplane to form a k-dimensional *projected structure*, $0 \leq k \leq n$, with less amount of communication overhead.

Next, a transformation method is proposed to transform the *projected structures* into parallel execution loops. In addition, an evaluation model is presented for estimating which one of the $n+1$ projected structures has minimum executing time. Finally, the *projected structures* are mapped onto hypercubes with a fixed number of processors. A mapping algorithm is proposed to minimize both the communication overhead and the workload imbalance among processors.

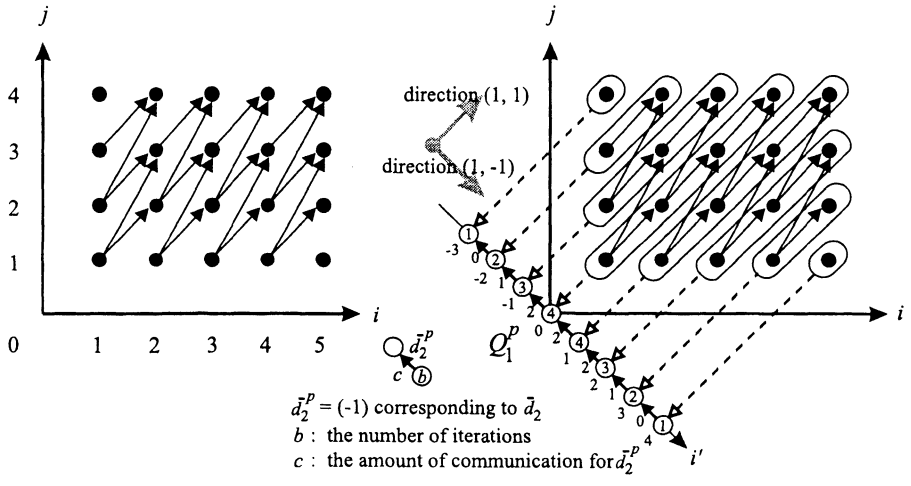
2 Preliminaries

Throughout this paper, some basic notations and assumptions are introduced. The symbol $|S|$ denotes the number of elements in set S. In this paper, we concentrate on the flow dependences because the antidependences, output dependences, and input dependences can be removed by techniques proposed in [10]. In an n-nested loop, if a variable A is generated at iteration $\bar{i} = (i_1, i_2, \ldots, i_n)$ and used at iteration $\bar{j} = (j_1, j_2, \ldots, j_n)$, then the *dependence vector* $\bar{d}$ of variable A is defined as a vector $(d_1, d_2, \ldots, d_n) \in \mathbf{Z}^n$ where $d_k = j_k - i_k$ for $1 \leq k \leq n$.

In general, an n-nested loop L can be represented as a directed graph Q in an n-dimensional space. Each vertex in Q represents an iteration and has a coordinate $(i_1, i_2, \ldots, i_n)$ in I^n. There is an arc from vertex v_i to vertex v_j, if the iteration corresponding to v_j depends on the iteration corresponding to v_i. Such a graph is called the *computational structure* [8] of the nested loop L. Therefore, a computational structure of an n-nested loop L can be denoted as $Q = (V, D)$, where $V = \{\ \bar{i} \mid \bar{i} = (i_1, i_2, \ldots, i_n) \in I^n\ \}$ is the set of vertices and D is the set of dependence vectors.

Given an n-nested loop L, we define a subspace $\Psi = \mathrm{span}(X)$, denoted as the *projection space*, where X consists of g linearly independent vectors, i.e., $\dim(\Psi) = g$. By *orthogonal projection*, iterations can be projected to the subspace $\mathrm{Ker}(\Psi)$, denoted as the null space of Ψ, along the projection space Ψ to form points denoted as the *projected points*. This is because the

Applications of High Performance Computing in Engineering VI, C.A. Brebbia, M. Ingber & H. Power (Editors)



(a) The computational structure of $L1$. (b) The 1-projected structure Q_1^p.

Figure 1: The computational structure and the 1-projected structure Q_1^p with the axis i' for loop $L1$.

inserted into its original loop body. Both of these compiling techniques are taken from [9].

In the following, an evaluation model is to be proposed for estimating which one of the $n+1$ *projected structures* generated by our partition strategy has the minimum execution time. Assume the target machine is fully connected and the iterations corresponding to each projected point are allocated onto one processor. Based on the pipelined and concurrent execution, the time of executing a number of tasks in parallel is dominated by the following three major factors: (1) *bottleneck computations* within some task, (2) *bottleneck communication* which occurs between some task and the others, and (3) the *critical path time* including computation and communication time during parallel executing tasks.

Therefore, the time of executing the *α-projected loop* [9], corresponding to the *α-projected structure* Q_α^p, is modeled as

$$T_{exe}^{\alpha} = T_{comp}^{\alpha} + T_{comm}^{\alpha} + T_{maxlength}^{\alpha} \tag{1}$$

for $0 \leq \alpha \leq n$. The last three terms in formula 1 corresponding to the above three respective factors are individually specified as follows. The bottleneck computation time T_{comp}^{α} is the maximum execution time within some projected point, denoted as

$$T_{comp}^{\alpha} = (\max_{1 \leq i \leq N_\alpha} b_i) t_{comp} \tag{2}$$

where N_α is the number of projected points and b_i is the number of iterations in the *i-th* projected point. The bottleneck communication time T_{comm}^{α}

Applications of High Performance Computing in Engineering VI, C.A. Brebbia, M. Ingber & H. Power (Editors)

g-dimensional subspace Ψ is perpendicular to the k-dimensional subspace $\mathrm{Ker}(\Psi)$ where $n = g + k$. Each projected point can be represented by k new index variables, denoted by I'_j for $1 \leq j \leq k$, easily derived from the orthogonal projection transformation. For a computational structure $Q = (V, D)$ of the n-nested loop L, a vertex $v_i \in V$ and a dependence vector $\bar{d}_i \in D$ are transformed into $v_i^p \in \mathbf{Z}^k$ and $\bar{d}_i^p \in \mathbf{Z}^k$, which are called *projected point* and *projected dependence vector*, respectively. The following definition formalizes the above descriptions.

Definition 1: [k-**projected structure** Q_k^p] [9]

Given a computational structure $Q = (V, D)$ of an n-nested loop L, the graph $Q_k^p = (V_k^p, D_k^p, \delta, \phi_1, \ldots, \phi_{|D_k^p|})$ is called a *k-projected structure* where
(1) V_k^p is the set of projected points,
(2) D_k^p is the set of projected dependence vectors which are not equal to $\mathbf{0}^k$,
(3) the iteration weight function $\delta : v^p \longrightarrow \mathbf{Z}$ is a mapping from a projected point to the number of iterations it contains, and
(4) the communication weight function $\phi_i : v^p \longrightarrow \mathbf{Z}$ is a mapping from a projected point to the amount of data communication concerning with the direction of the corresponding projected dependence vector $\bar{d}_i^p \in |D_k^p|$.

□

Example 1: Consider a 2-nested loop.

```
for i = 1 to M1
    for j = 1 to M2
        A[i,j] := (A[i-1,j-1] + A[i-1,j-2])/2 ;      (L1)
    end
end
```

Let $M_1 = 5$ and $M_2 = 4$. The iteration space $I^2 = \{(i,j) | 1 \leq i \leq 5, 1 \leq j \leq 4\}$. The dependence vectors of variable A are $\bar{d}_1 = (1,1)$ and $\bar{d}_2 = (1,2)$, and the corresponding set of dependence vectors $D = \{\bar{d}_1, \bar{d}_2\}$. Thus the computational structure of loop $L1$ is depicted in Fig. 1(a). We use our proposed method [9] to produce a 1-projected structure Q_1^p, with projection along the direction $\bar{d}_1$, to the loop $L1$ as shown in Fig. 1(b).

3 Partition of Nested Loops

To start with this subsection, the partition strategy is briefly described as follows. Given an n-nested loop L, the problem is how to transform the nested loop into a *k-projected structure*, $0 \leq k \leq n$, such that the amount of inter-block communication is as small as possible. The partition strategy is composed of two phases: the projection phase and the transformation phase. In the projection phase, we choose the best *projection space* so that the amount of communication cost can be reduced after transforming the computational structure into the *projected structure*. Next, during the transformation phase, each *projected structure* can be transformed into a parallel execution form with appropriate communication primitives automatically

Applications of High Performance Computing in Engineering VI, C.A. Brebbia, M. Ingber & H. Power (Editors)

is the maximum communication time that some projected point needs to communicate with the adjacent projected points, denoted as

$$T^{\alpha}_{comm} = (\max_{1 \le i \le N_\alpha} c_i)(t_{start} + t_{comm}) \tag{3}$$

where c_i is the number of data transfer in the i-th projected point which must send messages to the adjacent projected points. The time spent at the estimated critical path $T^{\alpha}_{maxlength}$ in Q^p_α, is denoted as

$$T^{\alpha}_{maxlength} = ((t_{start} + t_{comm})|D^p_\alpha| + t_{comp})MaxLength \tag{4}$$

where the $MaxLength$, critical path in Q^p_α, is modeled as $\sum_{k=1}^{|D^p_\alpha|} LPath_k$ which is the upper bound of the longest path. The $LPath_k$ is the maximum length of paths which are only connected by the edges in Q with the direction $\bar{d}_k \in D$ corresponding to $\bar{d}^p_{\alpha_k} \in D^p_\alpha$. If more than one of projected dependence vectors in D^p_α have the same direction, only one with the maximum value of $LPath_k$ is chosen as computing the value of $MaxLength$. This is because the time as sending the messages for the same direction in Q^p_α is dominated by only the longest path. In the critical path, each projected point needs to perform one iteration and send $|D^p_\alpha|$ messages to the adjacent projected points; thus, the critical path time is modeled as the formula 4.

To illustrate the accuracy of our evaluation model, reconsider Example 1. First, the estimated execution time of Q^p_1 is stated as follows. The bottleneck computation time $T^1_{comp} = 4t_{comp}$ within the projected point $v^p = (0)$. The bottleneck communication time $T^1_{comm} = 2(t_{start} + t_{comm})$ within the projected point $v^p = (0)$. The estimated critical path time $T^1_{maxlength}$ is $((t_{start} + t_{comm})|D^p_1| + t_{comp})MaxLength$ where $|D^p_1| = 1$ and $MaxLength = 1$. Thus, the estimated total execution time is

$$\begin{aligned} T^1_{exe} &= T^1_{comp} + T^1_{comm} + T^1_{maxlength} \\ &= 4t_{comp} + 2(t_{start} + t_{comm}) + ((t_{start} + t_{comm}) + t_{comp}) \\ &= 5t_{comp} + 3(t_{start} + t_{comm}). \end{aligned}$$

Comparing the estimated execution time T^1_{exe} and exact execution time $T_1 = 4t_{comp} + 2(t_{start} + t_{comm})$, T^1_{exe} approximates to T_1. Next, by our partition strategy for loop $L1$, the estimated execution time T^{α}_{exe} of the *α-projected structures* for $0 \le \alpha \le 2$ are shown in Table 1 where if the value of each entry is less than 0, then it is set to 0. Therefore, the best transformed parallel execution form with minimum total execution time is the 1-*projected structure* Q^p_1 and $t_{comp} = a(t_{start} + t_{comm})$ for $0 < a < 5$.

The formal partition algorithm is described below.

Algorithm 1: Partition algorithm

Input: A computational structure $Q = (V, D)$ of an n-nested loop L.

Output: A *k-projected loop* with the minimum execution time based on our proposed evaluation model.

Applications of High Performance Computing in Engineering VI, C.A. Brebbia, M. Ingber & H. Power (Editors)

Table 1: Estimated execution time of the three *projected loops* for loop $L1$.

α	Q_α^p	N_α	$\lvert D_\alpha^p \rvert$	$\max(b_i)$	$\max(c_i)$	$MaxLength$
0	Q_0^p	1	0	M_1M_2	0	0
1	Q_1^p	M_1+M_2-1	1	$\min(M_1,M_2)$	$\min(M_1,M_2)-2$	$\lfloor\frac{M_2-1}{2}\rfloor$
2	Q_2^p	M_1M_2	2	1	2	$\min(M_1,M_2)$ $+\lfloor\frac{M_2-1}{2}\rfloor-1$

Q_α^p	T_{exe}^α
Q_0^p	$M_1M_2t_{comp}$
Q_1^p	$(\min(M_1,M_2)+\lfloor\frac{M_2-1}{2}\rfloor)t_{comp}+(\min(M_1,M_2)+\lfloor\frac{M_2-1}{2}\rfloor-2)(t_{start}+t_{comm})$
Q_2^p	$(\min(M_1,M_2)+\lfloor\frac{M_2-1}{2}\rfloor)t_{comp}+2(\min(M_1,M_2)+\lfloor\frac{M_2-1}{2}\rfloor)(t_{start}+t_{comm})$

Step 1: /* Projection Phase */
By applying the projection technique in [9] to loop L, $n+1$ *projected structures* Q_α^p, $0 \le \alpha \le n$, can be generated.

Step 2: /* Transformation Phase */
Transform the original nested loop into the *α-projected loop* corresponding to the *α-projected structure* Q_α^p for $0 \le \alpha \le n$ [9].

Step 3: Calculate T_{exe}^α according to formula 1 for $0 \le \alpha \le n$.

Step 4: Select the number k such that $T_{exe}^k = \min_{0\le\alpha\le n} T_{exe}^\alpha$. Output the *$k$-projected loop* with minimum execution time based on our evaluation model.

□

4 Mapping Projected Structures onto Hypercubes

In this section, we discuss the problem of mapping the *projected structures* onto hypercube multicomputers. To start with, we formalize the mapping problem and present an objective function that we attempt to minimize. The *projected structure* is regarded as a task interaction graph (TIG) [7]. In a TIG, the vertices represent projected points and the edges represent communication requirements between projected points. Ideally, the total computational load should be distributed uniformly among all processors and no communication cost should be incurred. That is, in order to obtain the best mapping, we should minimize the penalties for communication overhead and for computation imbalance. The following objective function is used as a measure of the task assignment.

$$\min(COST) = \min(K_{comm}\sum_{j=1}^{M} C_j^{comm} + K_{comp}\sum_{j=1}^{M} \mathrm{abs}(W_{ave}^{load} - W_j^{load}))$$

Applications of High Performance Computing in Engineering VI, C.A. Brebbia, M. Ingber & H. Power (Editors)

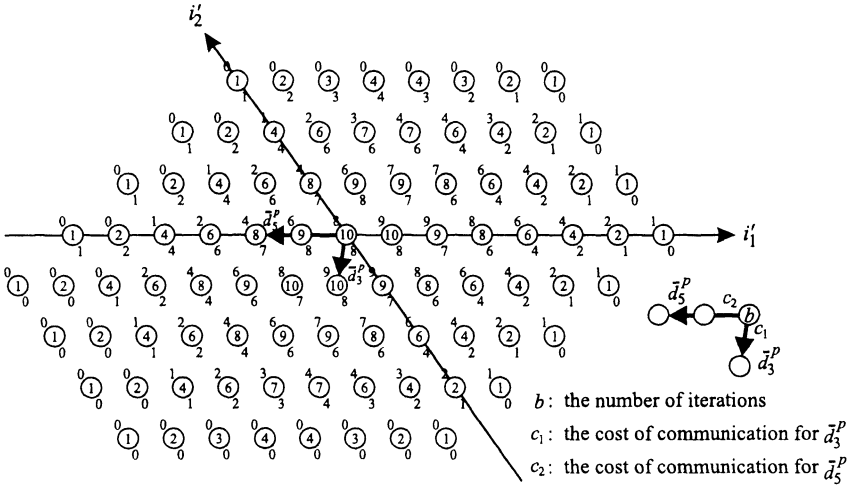


Figure 2: The 2-projected structure Q_2^p of loop $L2$.

The function abs(a) denotes the absolute value of $a \in \mathbf{Q}$. In the objective function, K_{comm} and K_{comp} are proportionality constants reflecting the relative penalties for communication and computational load imbalance. When K_{comm} is larger than K_{comp}, the communication overhead is more important than the workload imbalance, and vice versa. M is the number of processors. C_j^{comm} is the total communication cost of communicating between processors j and each processor i for $i \neq j$. W_j^{load} is the total computation cost of all iterations mapped onto processor j for $1 \leq j \leq M$. $W_{ave}^{load} = \frac{\sum_{j=1}^{M} W_j^{load}}{M}$ is the average workload.

Before describing our mapping algorithm, the following example is considered throughout this section.

Example 2: Consider a 4-nested loop $L2$. Here assume that its iteration space is $I^4 = \{(i_1, i_2, i_3, i_4) | \ 0 \leq i_1, i_2 \leq 3, \ 0 \leq i_3, i_4 \leq 4\}$. Let the dependence vectors in $L2$ be $\bar{d}_1 = (1, 0, -1, 0)$, $\bar{d}_2 = (0, 1, 2, 1)$, $\bar{d}_3 = (0, 0, 1, 1)$, $\bar{d}_4 = (1, 1, 1, 1)$, and $\bar{d}_5 = (0, 1, 0, 1)$. We denote that an iteration $\bar{i} = (i_1, i_2, i_3, i_4)$ for the data $message_j(i_1, i_2, i_3, i_4)$ is data dependent on other iterations through dependence vectors $\bar{d}_j$, respectively, for $1 \leq j \leq 5$. According to our proposed partition strategy in the previous section, the *projection space* $\Psi_2 = \text{span}(\{\bar{d}_1, \bar{d}_2, \bar{d}_4\})$ is chosen to generate the 2-*projected structure* Q_2^p as shown in Fig. 2.

□

Given an n-nested loop L, an algorithm for assigning a k-*projected structure* Q_k^p, $2 \leq k \leq n$, onto the hypercube is described in below. In general, the problem size, the number of tasks, is more larger than the machine size. Without loss of generality, assume that the number of partitioned blocks

corresponding to projected points $|V_k^p| = N$ is larger than the number of processors $M = 2^m$ in an m-dimensional hypercube.

In what follows, the basic idea of the heuristic mapping algorithm is stated. The *k-projected structure* is to be partitioned into as many pieces as the number of processors. To start with, we divide the whole *k-projected structure* into two clusters and divide each resulting clusters into two subclusters until there are M clusters generated. While separating each cluster into two subclusters, determining the partition of clusters is based on the following statements. For each hyperplane in a given set, we use the hyperplane to divide each cluster into two subclusters such that the penalty of computational load imbalance is minimal. Then, we evaluate the penalty of communication cost according to the assignment of each separated cluster to one processor in hypercube using the numbering scheme of Grade code. Thus the penalties of computational load imbalance and communication cost can be obtained. Finally, the partition results using one of these hyperplanes to divide the clusters with the minimum penalty are chosen. Based on the above scheme, we repeatedly divide each subcluster until M clusters are generated.

The mapping algorithm is formally described below.

Algorithm 2: Mapping algorithm on a hypercube for a higher dimensional *projected structure.* /* Finding the assignment by a heuristic approach */

Input: A *k-projected structure* Q_k^p described above with the set of projected dependence vectors $D_k^p = \{\bar{d}_1^p, \bar{d}_2^p, \ldots, \bar{d}_{|D_k^p|}^p\}$ and an m-dimensional hypercube with the number of processors $M = 2^m$.

Output: An assignment suited for parallel execution on the hypercube.

begin /* main */
Let $\bar{n}_i \in \mathbf{Z}^k$ be the normal vector of $\bar{d}_i^p$; $\bar{n}_i^t \cdot \bar{d}_i^p = 0$, for $1 \le i \le |D_k^p|$;
$W_{total} := \sum_{1 \le i \le N} W_i$; /* total workload of the *k-projected structure* */
Let the set *Cluster* consist of one cluster Q_k^p;
for $j = 1$ **to** m
$Penalty := \infty$;
for $i = 1$ **to** $|D_k^p|$
(1) Partition each cluster in *Cluster* into two subclusters to form the set $Cluster_i$ by the hyperplane with the normal vector $\bar{n}_i$ such that the following penalty of computational load imbalance is minimal
$Pen_{comp} = K_{comp} \sum_{c \in Cluster_i} \text{abs}(\frac{W_{total}}{2^j} - \text{workload of } c)$;
(2) Perform the procedure task-assignment(j, $Cluster_i$);
(3) Evaluate the penalty of communication cost
$Pen_{comm} = K_{comm}$(total communication cost)
based on the above assignment;
(4) Set β to i and set $Penalty$ to $Pen_{comp} + Pen_{comm}$
if $(Pen_{comp} + Pen_{comm}) < Penalty$;

Applications of High Performance Computing in Engineering VI, C.A. Brebbia, M. Ingber & H. Power (Editors)

end
Set *Cluster* to $Cluster_\beta$;
end
Perform the procedure task-assignment(m, *Cluster*);
end.

procedure task-assignment(j, *Cluster*)
/* Use j-bit binary codes to number these partitioned clusters, and assign each cluster into the corresponding processor in the j-dimensional hypercube. */
begin

(1) Let the *k-projected structure* be divided p_i times by the hyperplane with the normal vector $\bar{n}_i$ for $1 \leq i \leq |D_k^p|$. Because there are 2^j clusters generated and 2^{p_i} partitions in the normal vector $\bar{n}_i$, $j = p_1 + p_2 + \cdots + p_{|D_k^p|}$; that is, $2^j = 2^{p_1} \cdot 2^{p_2} \cdots 2^{p_{|D_k^p|}}$. Thus, we use p_i-bit Gray code for the normal vector $\bar{n}_i$. Then each cluster in *Cluster* has a unique j-bit binary representation which is obtained by concatenating the i-th coordinate for $1 \leq i \leq j$.

(2) Each cluster in *Cluster* is assigned to the processor whose binary number is the same as that of the cluster.

end.

□

In Algorithm 2, the purpose of using the hyperplane with the normal vector $\bar{n}_i$ of a selected $\bar{d}_i^p$ while partitioning the *k-projected structure* is to eliminate additional data communication between clusters for $\bar{d}_i^p$. Therefore, the amount of data transmission between clusters can be reduced. In Step (3) of the Algorithm 2, the *k-projected structure* is also divided into clusters such that the penalty for computational load imbalance is degraded. From the above two considerations, our proposed mapping algorithm can efficiently decrease the penalties for computational load imbalance and communication cost.

The time complexity of Algorithm 2 is step by step to be analyzed as follows. For the procedure task-assignment(j, *Cluster*), the time-consuming parts which are to concatenate the Gray code of each cluster and to assign each cluster to the corresponding processor must take the time complexity $O(j2^j)$. Step (1) to Step (4) must be performed $m|D_k^p|$ times. For each time, it requires to take the time $O(j2^j)$ and $O(|D_k^p|N)$ to perform the procedure task-assignment(j, *Cluster*) and evaluate the penalties for computational load imbalance and communication cost, respectively. Therefore, this algorithm has to take the time complexity $O(m|D_k^p| \max(mM, |D_k^p|N))$. Finally we take into account the following example for illustrating the above mapping algorithm.

Example 3: We have a 2-dimensional hypercube consisting of four pro-

Applications of High Performance Computing in Engineering VI, C.A. Brebbia, M. Ingber & H. Power (Editors)

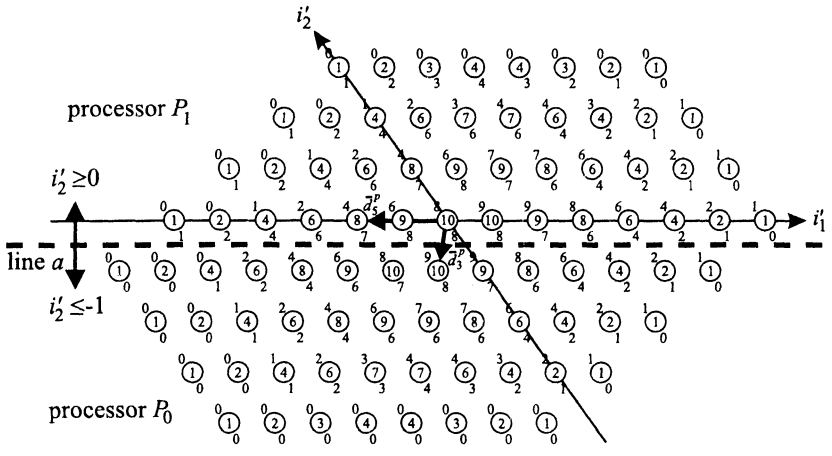


Figure 3: Choosing the first hyperplane with the normal vector $\bar{n}_2$ to divide the 2-projected structure Q_2^p of loop $L2$ into two subclusters.

cessors $P_{0,0}$, $P_{0,1}$, $P_{1,0}$ and $P_{1,1}$, and the 2-*projected structure* of loop $L2$ already depicted in Fig. 2. Since there are 2^2 processors, the 2-*projected structure* has to be divided 2 times. From Fig. 2, the respective normal vectors of projected dependence vectors $\bar{d}_3^p = (-1,-1)$ and $\bar{d}_5^p = (-2,0)$ are $\bar{n}_1 = (1,-1)$ and $\bar{n}_2 = (0,1)$ because $\bar{n}_1^t \cdot \bar{d}_3^p = 0$ and $\bar{n}_2^t \cdot \bar{d}_5^p = 0$.

For $j = 1$, by Algorithm 2, choosing the first hyperplane to partition the 2-*projected structure* into two subclusters is discussed in the following. First, we use the hyperplane with the normal vector $\bar{n}_1$ to partition the 2-*projected structure* into two subclusters. It must pay the minimum penalty 60 time units for computational load imbalance. After evaluating the penalty for communication cost which is 77 time units, the total penalty by the above dividing has to take $137(= 60 + 77)$ time units. Similarly, repeating the above processing for the hyperplane with the normal vector $\bar{n}_2$, the total penalties for computational load imbalance, 0 time unit, and communication cost, 64 time units, have to take $64(= 0 + 64)$ time units as shown in Fig. 3. Therefore, because the penalty 64 time units is less than 137 time units, the first hyperplane with the normal vector $\bar{n}_2 = (0,1)$ is chosen and the 2-*projected structure* is separated into two subclusters by the line a as shown in Fig. 3 where one is located in the side $i_2' \geq 0$, and another is located in the side $i_2' \leq -1$.

For $j = 2$, similarly, performing Step (1) to Step (4) of Algorithm 2, the second hyperplane with the normal vector $\bar{n}_1 = (1,-1)$ is chosen. Thus, two clusters can be, respectively, separated into two subclusters by the line b and line c as shown in Fig. 4. By the above partition, the penalty for computational load imbalance is 0. However, the penalty for communication cost takes 152 time units in which 136 time units are spent in communication between neighboring processors and $16(= 8 \times 2)$ time units are spent

Applications of High Performance Computing in Engineering VI, C.A. Brebbia, M. Ingber & H. Power (Editors)

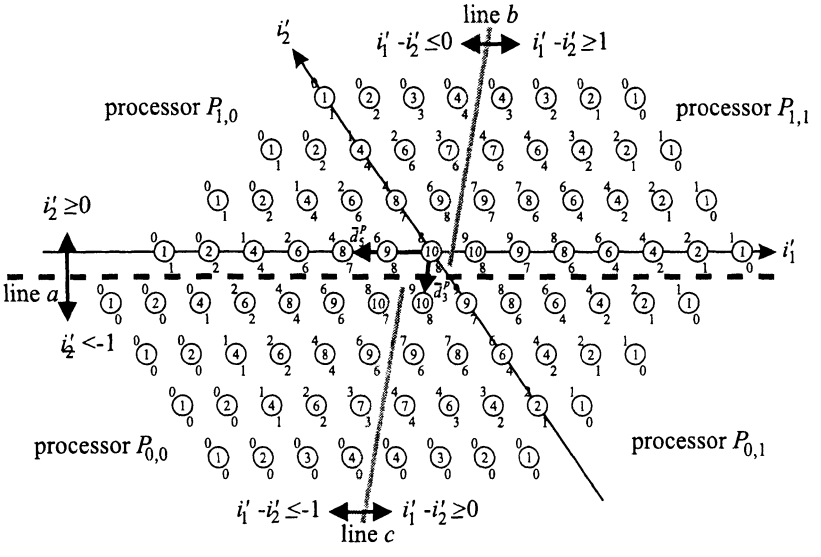


Figure 4: Mapping the 2-projected structure Q_2^p of loop $L2$ onto a 2-dimensional hypercube.

in communication between processors $P_{1,0}$ and $P_{0,1}$. The total minimum penalty need to take 144 time units. In Fig. 4, for the cluster in the side $i'_2 \geq 0$, one subcluster is located in the side $i'_1 - i'_2 \leq 0$ and another is located in the side $i'_1 - i'_2 \geq 1$. For the cluster in the side $i'_2 \leq -1$, one subcluster is located in the side $i'_1 - i'_2 \leq -1$ and another is located in the side $i'_1 - i'_2 \geq 0$. Finally, choosing the 1-bit Gray code for $\bar{n}_1$ and 1-bit Gray code for $\bar{n}_2$ by the procedure task-assignment, we can assign each cluster into the corresponding processor as shown in Fig. 4.

After completing the preceding assignment, the parallel execution code can be easily transferred based on the transformation of *projected loop* derived in the previous section. The program segments in the processors $P_{0,0}$, $P_{0,1}$, $P_{1,0}$, and $P_{1,1}$ will be executed in parallel.

□

5 Conclusions

In this paper, a systematic partition strategy was proposed. First, an n-nested loop can be transformed into a *k-projected structure*, $0 \leq k \leq n$, with less communication overhead. After partitioned, one of $n+1$ *projected structures* can be chosen so that it has minimum execution time based on our presented evaluation model. Concentrating on higher dimensionality k of *projected structures* for $2 \leq k \leq n$, an efficient heuristic mapping algorithm, was proposed for dividing a *projected structure* through the selected

hyperplanes into subclusters mapped onto processors of a given hypercube.

Acknowledgements

This work was supported in part by the National Science Council of Republic of China under Grant #NSC88-2213-E-309-002.

References

[1] P. Boulet, A. Darte, T. Risset, and Y. Robert," "(Pen)-Ultimate Tiling," *Integration, VLSI J.*, Vol. 17, pp. 33-51, 1994.

[2] T.-S. Chen and J.-P. Sheu, "Communication-Free Data Allocation Techniques for Parallelizing Compilers on Multicomputers," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 5, No. 9, pp. 924-938, Sep. 1994.

[3] E. Hodzic and W. Shang, "On Supernode Transformation with Minimized Total Running Time," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 9, No. 5, pp. 417-428, May 1998.

[4] F. Irigoin and R. Triolet, "Supernode Partitioning," *Proceedings of the Fifteenth Annual ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages*, pp. 319-329, January 1988.

[5] A. W. Lim and M. S. Lam, "Communication-Free Parallelization via affine Transformations," in *Proceedings of 7th International Workshop on Languages and Compilers for Parallel Computing*, USA, Aug. 1994.

[6] G. Rivera and C.-W. Tseng, "A Comparison of Compiler Tiling Algorithms," in the *Proceedings of the 8th International Conference on Compiler Construction*, Amsterdam, Netherlands, March 1999.

[7] P. Sadayappan and F. Ercal, "Nearest-Neighbor Mapping of Finite Element Graphs onto Processor Meshes," *IEEE Transactions on Computers*, Vol. C-36, No. 12, pp. 1408-1424, December 1987.

[8] J.-P. Sheu and T.-H. Tai, "Partitioning and Mapping Nested Loops on Multiprocessor Systems," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 2, No. 4, pp. 430-439, October 1991.

[9] J.-P. Sheu and T.-S. Chen, "Partitioning and Mapping of Nested Loops for Linear Array Multicomputers," *The Journal of Supercomputing*, Vol. 9, pp. 183-202, 1995.

[10] M. J. Wolfe, "Optimizing Supercompilers for Supercomputers," London and Cambridge, MA: Pitman and the MIT Press, 1989.

[11] J. Xue, "On Tiling as a Loop Transformation," *Parallel Processing Letters*, Vol. 7, No. 4, pp. 409-424, 1997.