Note for *The Parallel Execution of DO Loops*

Ying Cao

March 19, 2024

Contents

1 Basics **1**

2 The hyperplane method **2**

2.1 Definitions . . . 2

2.2 Assumptions . . . 2

2.3 Formulation of the problem . . . 3

2.4 Concurrent executions of the loop body . . . 3

2.5 Conditions for a legal rewriting . . . 3

2.6 The hyperplane theorem . . . 4

2.6.1 The existence of π . . . 4

2.6.2 The optimality of π . . . 5

1 Basics

The analysis in paper [1] is performed from the standpoint of a compiler for a *multiprocessor computer*. It considers loops of the following form:

$$\begin{array}{l} \textbf{\textit{DO}} \quad I^1 = l^1, u^1 \\ \qquad \vdots \\ \textbf{\textit{DO}} \quad I^n = l^n, u^n \\ \qquad \boxed{\text{loop body}} \\ \textbf{\textit{CONTINUE}} \end{array} \tag{1}$$

where l^j and u^j may be integer-valued expresions, possibly involving $I^1,...,I^{j-1}$. Let $\mathbf{Z}^n$ denote the set of n-tuples of integers. So the index set ζ of loop (1) is the subset of $\mathbf{Z}^n$ consisting all values assumed by $(I^1,...,I^n)$ during execution of the loop.

The loop body is executed multiple times — once for each point $(i^1,...,i^n)$ in the ***index set***

$$\zeta = \{(i^1,...,i^n) : l^1 \le i^1 \le u^1,...,l^n \le i^n \le u^n\}.$$

Usually, one execution of the loop body is called an *instance*.

The goal is, **we want to speed up the computation by performing some of these executions concurrently**. As solutions, this paper proposed two general methods:

1) *hyperplane method*, applicable to:

   1. multiple instruction stream computer

2. single instruction stream computer

2) *coordinate method*, applicable to:

   1. single instruction stream computer

Both methods rewrite the original loop programs into the form loop (2):

$$
\begin{aligned}
&\textbf{\textit{DO}} \quad \alpha \quad J^1 = \lambda^1, \mu^1 \\
&\qquad\qquad \vdots \\
&\textbf{\textit{DO}} \quad \alpha \quad J^k = \lambda^k, \mu^k \\
&\textbf{\textit{DO}} \quad \alpha \quad \textbf{\textit{CONC FOR ALL}} \\
&\qquad (J^{k+1}, \dots, J^n) \in \mathscr{S}_{J^1,\dots,J^k} \\
&\qquad\qquad \boxed{\text{loop body}} \\
&\textbf{\textit{CONTINUE}}
\end{aligned}
\tag{2}
$$

2 The hyperplane method

2.1 Definitions

- VAR:a program array variable.
- ***occerence***:any appearance of VAR in the loop body.
- ***generation/ use***:VAR appears on the left - hand side of an assignment statement; Such an occurrence is called *generation*; otherwise, a *use*.
   1. generations modify values of array elements which uses do not.
   2. an occurrence is a *generation*or *use*.
- ***occurrence mapping***:$T_f : \zeta \to Z^d$ where f is an occerence. d is the dimensionality of the array to access.
   1. the occurrence mapping maps points in index set ζ to array indices.
   2. the occurrence mapping relates time and space.

2.2 Assumptions

Following assumptions are made to the loop body:

(A1) It contains no I/O statement.

(A2) It contains no transfer of control to any statement outside the loop.

(A3) It contains no subroutine or function call which can modify data.

(A4) Any occurrence in the loop body of a generated variable VAR is of the form VAR($e^1, \dots, e^\tau$), where each e^i is an expression not containing any generated variable.

2.3 Formulation of the problem

The hyperplane method formulates performing the rewriting procedure as constructing a one-to-one linear mapping $J : \mathbf{Z}^n \to \mathbf{Z}^n$ of the form:

$$
\begin{aligned}
J[(I^1, ..., I^n)] &= \left(\sum_{j=1}^{n} a_j^1 I^j, ..., \sum_{j=n}^{n} a_j^n I^j \right) \\
&= (J^1, ..., J^n)
\end{aligned} \tag{3}
$$

In fact, for the *finite - dimensional vector spaces with a defined basis*, a vector spaces we're mostly interested in, any linear mapping can be represented by a matrix T that is multiplied by the input vector [2]. So, we can write equation (3) in the matrix form.

$$
J[(I^1, ..., I^n)] = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} \end{pmatrix} \begin{pmatrix} I_1 \\ I_2 \\ \vdots \\ I_n \end{pmatrix} \tag{4}
$$

Once the one-to-one mapping J is constructed, we then choose:

1) the λ^i, μ^i and $\mathscr{S}_{J^1,...,J^k}$ to assure that the index set ζ of loop (2) equals $J(\zeta)$.
2) write the loop body of loop (2).

To perform rewriting, two important questions should be answered:

Question 1

1 Under what conditions, the rewritten loop (2) is equivalent to the given loop (1)?

2 How to construct the one-to-one linear mapping J (or construct the matrix T)?

2.4 Concurrent executions of the loop body

Define mapping $\pi : \mathbf{Z}^n \to \mathbf{Z}^k$ by $\pi[(I^1, ..., I^n)] = (J^1, ..., J^k)$:

- mapping $\pi(P)$ contains the first k coordinates of $J(P)$, which are sequential loops.
- the set defined by $\{P : \pi(P) = \text{constant} \in \mathbf{Z}^k\}$ are parallel $(n-k)$-dimensional planes in $\mathbf{Z}^n$. Loop body is executed concurrently for elements of ζ lying on these sets.
- these sets are parallel $(n-k)$-dimensional planes in $\mathbf{Z}^n$, hence the name "hyperplane method".

2.5 Conditions for a legal rewriting

***Sufficient condition* for loop (2) to be equivalent to loop (1)**

(C1) For ***every*** variable, and ***every*** ordered pair of occurrences f, g of that variable, at least one of which is a generation: if $T_f(P) = T_g(Q)$ for $P, Q \in \zeta$ with $P < Q$, then π must satisfy the relateion $\pi(P) < \pi(Q)$.

For most loops, (C1) is also a necessary condition.
However, (C1) requires to consider many pairs of points P, Q in ζ. To address this problem and ease the analysis, the author uses a set descriptor $\langle f, g \rangle$ rather than directly considering all the pairs of P, Q.

Define $\langle f,g\rangle$ a subset of $\mathbf{Z}^n$ by:

$$\langle f,g\rangle = \left\{X : T_f(P) = T_g(P+X) \quad \text{for some } P \in \mathbf{Z}^n\right\}$$

Set $\langle f,g\rangle$ defines pairs of *use* and *generation* that accesses the same memory location. Though this notation is not implicitly called dependence vectors in this paper, I think it is $\mathbf{X}$ essentially the dependence vector in later research works. Then, we have a more strigent rule:

Rule for loop (2) to be equivalent to loop (1)

(C2) For every variable, and every ordered pair of occurrences f,g of that variable, at least one of which is a generation: for every $\mathbf{X} \in \langle f,g\rangle$ with $\mathbf{X} > \mathbf{0}$, π must satisfy $\pi(\mathbf{X}) > \mathbf{0}$.

To guarantee that it is feasible to find a mapping π which satisfies (C2), the author futher make some restriction (see (A5) in the paper) on the forms of variable occurrences. This restriction actually leads to ***constant dependence vectors***.

2.6 The hyperplane theorem

2.6.1 The existence of π

C2 gives a set of constraints on the mapping $\pi : \mathbf{Z}^n \to \mathbf{Z}$, and the Hyperplane Theorem proves ***the existence*** of a π satisfying those constraints. The proof can also serve as an algorithm for constructing a mapping π, but it is not always the optimal.

HYPERPLANE CONCURRENCY THEOREM. *Assume that loop (1)satisfies (A1) - A(5), and that none of the index variables $I^2,...,I^n$ are missing variable. Then it can be rewritten in the form of (2)for k* $= 1$. *Moreover, the mapping J used for the rewriting can be chosen to be independent of the index set ζ.*

Notations

- The mapping π is defined by:
$$\pi[(I^1,...,I^n)] = a_1 I^1 + ... + a_n I^n$$
- $\Theta = \{\mathbf{X}_r\}, r = 1,...,N$, $\mathbf{X}_r > \mathbf{0}$ is all the elements of $\langle f,g\rangle$ referred to in (C2). Since I^1 is the only index variable that may be mising, $X_r = (x_r^1,...,x_r^n)$ or $X_r = (+,x_r^2,...,x_r^n)$.
- $\Theta_j = \left\{\mathbf{X}_r : x_r^1 = ... = x_r^{j-1} = 0, x_r^j \neq 0\right\}$. Θ_j is the set of all $\mathbf{X}_r$ whose jth coordinate is the left-most nonzero one. Each $\mathbf{X}_r$ is an element of some Θ_j.

Proof. If we can always construct the mapping $\pi : \mathbf{Z}^n \to \mathbf{Z}$, and from π obtrain the one-one mapping J, then the theorem is proved.

1. Construct the mapping $\pi : \mathbf{Z}^n \to \mathbf{Z}$ for each $\mathbf{X}_r \in \Theta$.
   (a) replace each $X_r = (+,x_r^2,...,x_r^n)$ by $(1,x_r^2,...,x_r^n)$. This is because we require $\pi[(+,x_r^2,...,x_r^n)] > 0$, then it is sufficient to consider replacing x with the smallest positive integer 1.
   (b) for $j = n, n-1,...,1$, let a_j be the smallest nonnegative integer such that $a_j x_r^j + ... + a_n x_r^n > 0$ for each
   $X_r = (0,...,0,x_r^j,...,x_r^n) \in \Theta_j$.
2. construct $J[(I^1,...,I^n)] = (\pi[(I^1,...,I^n)],...)$. If $a_j = 1$, define J as follows: for each $l \geq 2$, let J^k equal some distinct I^{l_k} with $l_j \neq j$.

□

2.6.2 The optimality of π

The hyperplane theorem proves the existence of π. So far, there is still a problem left, the optimal π.

$k = 1$ means the outermost loop is sequential while all the inner loops are parallelable. Therefore, it is reasonable to minimize the number of steps in the outer ***DO*** J^1 loop(2). This means minimizing $\mu^1 - \lambda^1$ in loop(2).

Setting $M^i = \mu^i - l^i$, it is easy to see that $\mu^1 - \lambda^1$ equals:

$$M^1|a_1| + ... + M^n|a_n| \tag{5}$$

Thus, finding an optimal π is reduced to the integer programming problem: ***find integers*** $a_1, ..., a_n$ ***satisfying inequalities given by (C2) which minimize expression 5***.

This problem is to find solution to the linear Diophantine equation [3, 4]. For some linear Diophantine equations, it is algorithmatically solvable, but there are no practical method of finding a solution in the general case.

Info: Below theorem is from [5]. It is referred as an important conclusion in [6]. I just think the conclusion below further improves the **hyperplane concurrency theorem** by:

1. propose the concept "fully permutable loop" which gives a more strigent conditions of loop programs are able to obtain the optimal parallelism. The concept "fully permutable loop" is also the foundation for tiling.
2. give the transformation that leads to the optimal parallelism for "fully permutable loop".

A nest of n fully permutable loops can be transformed to code containing at least $n-1$ degress of parallelism. In the degenerate case when no dependences are carried by these n loops, the degree of parallelism is n.

References

[1] L. Lamport, “The parallel execution of do loops,” *Communications of the ACM*, vol. 17, no. 2, pp. 83–93, 1974.

[2] E. Bendersky, “Affine transformations.” `https://eli.thegreenplace.net/2018/affine-transformations/`. Accessed Feburary 4, 2020.

[3] “Diophantine equation.” `https://en.wikipedia.org/wiki/Diophantine_equation`. Accessed Feburary 4, 2020.

[4] L. J. Mordell, *Diophantine equations*. Academic Press, 1969.

[5] F. Irigoin and R. Triolet, “Supernode partitioning,” in *Proceedings of the 15th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pp. 319–329, 1988.

[6] M. E. Wolf and M. S. Lam, “A loop transformation theory and an algorithm to maximize parallelism,” *IEEE Transactions on Parallel & Distributed Systems*, no. 4, pp. 452–471, 1991.