Improving Parallelism of Nested Loops with Non-uniform Dependences

Sam Jin Jeong and Kun Hee Han

Division of Information and Communication Engineering,
Cheonan University Anseo-dong 115, Cheonan City, Korea 330-704
{sjjeong, hankh}@cheonan.ac.kr

Abstract. This paper defines the properties of FDT (Flow Dependence Tail set) and FDH (Flow Dependence Head set), and presents two partitioning methods for finding two parallel regions in two-dimensional solution space. One is the region partitioning method by intersection of FDT and FDH. Another is the region partitioning method by two given equations. Both methods show how to determine whether the intersection of FDT and FDH is empty or not. In the case that FDT does not overlap FDH, we will divide the iteration space into two parallel regions by a line. The iterations within each area can be fully executed in parallel. So, we can find two parallel regions for doubly nested loops with non-uniform dependences for maximizing parallelism.

1 Introduction

The evolutionary transition from sequential to parallel computing offers the promise of quantum leap in computing power [1]. In the past few years, many techniques for exploiting parallelism within nested loops have been developed, and they have been automated and collected to form parallelizing compilers.

Example l.

```
do i = 1, 10
  do j = 1, 10
    A(2i+3, j+1) = . . .
          . . . = A(i+2j+1, i+j+1)
  enddo
enddo
```

Several works has been done for loops with non-uniform dependences, but show us poor performance. Some techniques, based on Convex Hull theory [5] that has been proven to have enough information to handle non-uniform dependences, are the minimum dependence distance tiling method [4], the unique set oriented partitioning method [3], and the three region partitioning method [2], [7].

Fig. 1(a) shows the dependence patterns of Example 1 in the iteration space.

This paper will focus on parallelization of flow and anti dependence loops with non-uniform dependences. Especially, it shows us two partitioning methods to find two parallel regions in doubly nested loops with non-uniform dependences.

H. Jin, D. Reed, and W. Jiang (Eds.): NPC 2005, LNCS 3779, pp. 205 – 212, 2005.

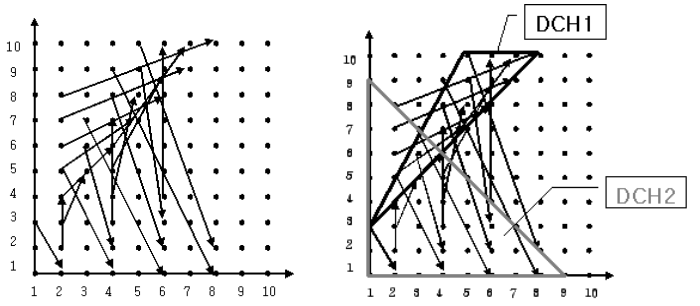

Fig. 1. (a) Iteration Spaces (b) CDCH of Example 1

The rest of this paper is organized as follows. Chapter two describes our loop model, and introduces the concept of Complete Dependence Convex Hull (CDCH). In chapter three, we define the properties of FDT (Flow Dependence Tail set) and FDH (Flow Dependence Head set), and show how to find FDT and FDH. We also present two partitioning methods to find two parallel regions in the given space. Chapter four shows comparison with related works. Finally, we conclude in chapter five with the direction to enhance this work.

2 Program Model and Dependence Analysis

The loop model considered in this paper is doubly nested loops with linearly coupled subscripts and both lower and upper bounds for loop variables should be known at compile time. The loop model has the form in Fig. 2, where $f_1(I, J)$, $f_2(I, J)$, $f_3(I, J)$, and $f_4(I, J)$ are linear functions of loop variables.

```
do I = l1, u1
  do J = l2, u2
          A(f1(I, J), f2(I, J)) = . . .
          . . . = A(f3(I, J), f4(I, J))
  enddo
enddo
```

Fig. 2. A doubly nested loop model

The loop in Fig. 2 carries cross iteration dependences if and only if there exist four integers (i_1, j_1, i_2, j_2) satisfying the system of linear diophantine equations given by (1) and the system of inequalities given by (2). The general solution to these equations can be computed by the extended GCD or the power test algorithm [6] and forms a **DCH** (Dependence Convex Hull).

$$f_1(i_1, j_1) = f_3(i_2, j_2) \text{ and } f_2(i_1, j_1) = f_4(i_2, j_2) \tag{1}$$

$$l_1 \le i_1, i_2 \le u_1 \text{ and } l_2 \le j_1, j_2 \le u_2 \tag{2}$$

From (1), (i_1, j_1, i_2, j_2) can be represented as

$$(i_1, j_1, i_2, j_2) = (g_1(i_2, j_2), g_2(i_2, j_2), g_3(i_1, j_1), g_4(i_1, j_1))$$

where g_i are linear functions.

From (2), two sets of inequalities can be written as

$$l_1 \le i_1 \le u_1 \text{ and } l_2 \le j_1 \le u_2 \text{ and} \quad (3)$$

$$l_1 \le g_3(i_1, j_1) \le u_1 \text{ and } l_2 \le g_4(i_1, j_1) \le u_2$$

$$l_1 \le i_2 \le u_1 \text{ and } l_2 \le j_2 \le u_2 \text{ and} \quad (4)$$

$$l_1 \le g_1(i_2, j_2) \le u_1 \text{ and } l_2 \le g_2(i_2, j_2) \le u_2$$

And, (3) and (4) form DCHs denoted by DCH1 and DCH2, respectively [3]. Clearly, if we have a solution (i_1, j_1) in DCH1, we must have a solution (i_2, j_2) in DCH2, because they are derived from the same set of equations (1). The union of DCH1 and DCH2 is called Complete DCH (**CDCH**), and all dependences lie within the CDCH. Fig. 1(b) shows the CDCH of Example 1.

If iteration (i_2, j_2) is dependent on iteration (i_1, j_1), then we have a dependence vector $d(i_1, j_1) = (d_i(i_1, j_1), d_j(i_1, j_1)) = (i_2\text{-}i_1, j_2\text{-}j_1)$

So, for DCH1, we have

$$d_i(i_1, j_1) = g_3(i_1, j_1) - i_1 = (\alpha_{11} - 1)i_1 + \beta_{11}j_1 + \gamma_{11} \text{ and} \quad (5)$$

$$d_j(i_1, j_1) = g_4(i_1, j_1) - j_1 = \alpha_{12}i_1 + (\beta_{12} - 1)j_1 + \gamma_{12}$$

For DCH2, we have

$$d_i(i_2, j_2) = i_2 - g_1(i_2, j_2) = (1 - \alpha_{21})i_2 - \beta_{21}j_2 - \gamma_{21} \text{ and} \quad (6)$$

$$d_j(i_2, j_2) = j_2 - g_2(i_2, j_2) = -\alpha_{22}i_2 + (1 - \beta_{22})j_2 - \gamma_{22}$$

We can write these dependence distance functions in a general form as

$$d(i_1, j_1) = (d_i(i_1, j_1), d_j(i_1, j_1)), d(i_2, j_2) = (d_i(i_2, j_2), d_j(i_2, j_2)) \quad (7)$$

$$d_i(i_1, j_1) = p_1*i_1 + q_1*j_1 + r_1, d_j(i_1, j_1) = p_2*i_1 + q_2*j_1 + r_2$$

$$d_i(i_2, j_2) = p_3*i_2 + q_3*j_2 + r_3, d_j(i_2, j_2) = p_4*i_2 + q_4*j_2 + r_4$$

where p_i, q_i, and r_i are real values and i_1, j_1, i_2, and j_2 are integer variables of the iteration space. The properties of DCH1 and DCH2 can be found in [3].

The set of inequalities and dependence distances of the loop in Example 1 are computed as follows.

$$\text{DCH1} : 1 \le i_1 \le 10, 1 \le j_1 \le 10$$

$$1 \le -2i_1 + 2j_1 - 2 \le 10, 1 \le 2i_1 - j_1 + 2 \le 10$$

$$d_i(i_1, j_1) = -3i_1 + 2j_1 - 2,\ d_j(i_1, j_1) = 2i_1 - 2j_1 + 2$$

$$\text{DCH2} : 1 \le i_2/2 + j_2 - 1 \le 10, 1 \le i_2 + j_2 \le 10$$

$$1 \le i_2 \le 10, 1 \le j_2 \le 10$$

$$d_i(i_2, j_2) = i_2/2 - j_2 + 1, d_j(i_2, j_2) = -i_2$$

3 Region Partitioning Methods for Two Parallel Regions

In this section, we propose two partitioning methods to find two parallel regions in the given space. One is the region partitioning method by intersection of FDT and FDH. Another is the region partitioning method by two given equations. Both methods show how to determine whether the intersection of FDT and FDH is empty or not.

3.1 Region Partitioning Method by Intersection of FDT and FDH

We define the flow dependence tail set (FDT) and the flow dependence head set (FDH) as follows.

Definition 1. *Let L be a doubly nested loop with the form in Fig. 2. If line $d_i(i_1, j_1) = 0$ intersects DCH1, the flow dependence tail set of the DCH1, namely **FDT**(L), is the region H, where H is equal to*

$$\text{DCH1} \cap \{(i_1, j_1) \mid d_i(i_1, j_1) \geq 0 \text{ or } d_i(i_1, j_1) \leq 0 \} \tag{8}$$

Definition 2. *Let L be a doubly nested loop with the form in Fig. 2. If line $d_i(i_2, j_2) = 0$ intersects DCH2, the flow dependence head set of the DCH2, namely **FDH**(L), is the region H, where H is equal to*

$$\text{DCH2} \cap \{(i_2, j_2) \mid d_i(i_2, j_2) \geq 0 \text{ or } d_i(i_2, j_2) \leq 0 \} \tag{9}$$

Property 1. *Suppose line $d_i(i, j) = p*i+q*j+r$ passes through CDCH. If $q > 0$, FDT(FDH) is on the side of $d_i(i_1, j_1) \geq 0$ ($d_i(i_2, j_2) \geq 0$), otherwise, FDT(FDH) is on the side of $d_i(i_1, j_1) \leq 0$ ($d_i(i_2, j_2) \leq 0$).*

We can form two regions, FDT and FDH, by the algorithm of finding FDT or FDH in two-dimensional solution space in Fig. 3, which is similar to the algorithm presented in [5].

Fig. 4 shows the head and tail sets of flow dependence, anti dependence, and FDH and FDT of the loop in Example 1.

By Property 1, we can know the area of the flow dependence head set (FDH) of DCH1 and the flow dependence tail set (FDT) of DCH2 in Example 1 as shown in Fig. 4. In this example, because the intersection of FDT and FDH is empty, FDT does not overlap FDH and the iteration space is divided into two parallel regions by the line $d_i(i_2, j_2) = 0$. From equation (7), we can get $d_i(i_2, j_2) = i_2/2 - j_2 + 1$, and the equation is $j = i/2+1$. So, the iteration space is divided into two parallel regions, AREA1 and AREA2, by the line $j = i/2+1$. The execution order is AREA1 → AREA2.

Transformed loops are given as follows.

```
/* AREA1 – parallel region */
doall i    l1   u1
  doall j    max(l2, ⌈i/2+1⌉), u2
      A(2i+3, j+1) = . . .
        . . . = A(i+2j+1, i+j+1)
  enddoall
enddoall
```

```
/* AREA2 – parallel region */
doall i    l1   u1
  doall j    l2   min(u2, ⌈i/2+1⌉)
      A(2i+3, j+1) = . . .
            . . . = A(i+2j+1, i+j+1)
  enddoall
enddoall
```

```
Algorithm FDT (or FDH)
Input: A list of 9 half spaces (Def. 1 or 2)
Output: An FDT (or FDH);
          struct node {
            float (x, y);
            int zoom;
            struct node *next;
              struct node *prev; };
          max = 9999999;
BEGIN
Build the initial FDT (or FDH) ring which is composed of four
nodes:
    (x1, y1) = ( max, max);
    (x2, y2) = ( max, -max);
    (x3, y3) = (-max, -max);
    (x4, y4) = (-max, max);
  while (the input list is not empty)
    Pop a half space from the list, named HS;
    Scan the ring;
      { Determine the zoom value for each node; }
      if ((x, y) ∈ HS ) then
         zoom = 0;
      else
         zoom = 1;
      if (the zoom is different from previous node)then
         { Compute the intersection point
           and give it zoom = 0;
           Insert it into the ring between the
           current node and the previous node };
      endif
    Scan the ring again;
      { Remove the nodes with zoom = 1 };
    if (the ring is empty) STOP;
  end while
END FDT
```

Fig. 3. Algorithm of finding FDT (or FDH) in two-dimensional solution space

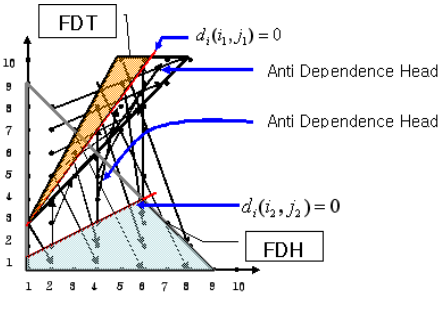

Fig. 4. FDT and FDH in Example 1

3.2 Region Partitioning Method by Two Given Equations

In our proposed algorithm in Fig. 5, Algorithm Region_Partition, we can determine whether the intersection of FDT and FDH is empty by position of two given lines $d_i(i_1, j_1) = 0$ and $d_i(i_2, j_2) = 0$, and two real values q_1 and q_3 given in (7). If the intersection of FDT and FDH is not empty, we divide the iteration space into two parallel regions and one serial region by two appropriate lines as given in the three region partitioning method [2], [7]. If the intersection of FDT and FDH is empty, we divide the iteration space into two parallel regions by the line $d_i(i_1, j_1) = 0$ or $d_i(i_2, j_2) = 0$.

```
Algorithm Region_Partition
INPUT: two lines (d_i(i_1, j_1) = 0, d_i(i_2, j_2) = 0) and two real values
(q_1, q_3)
OUTPUT: two parallel regions
BEGIN
If (line d_i(i_1, j_1) = 0 is on the left side of line d_i(i_2, j_2) =0)
     If (q_1 > 0 and q_3 < 0){
        /* AREA1 does not overlap AREA2 */
       AREA1: {(i_1, j_1) | d_i(i_1, j_1) ≥ 0}
       AREA2: {(i_1, j_1) | d_i(i_1, j_1) < 0} }
Else if (d_i(i_1, j_1) = 0 is on the right side of d_i(i_2, j_2) = 0)
     If (q_1 < 0 and q_3 > 0) {
        /* AREA1 does not overlap AREA2 */
       AREA1: {(i_1, j_1) | d_i(i_1, j_1) ≤ 0}
       AREA2: {(i_1, j_1) | d_i(i_1, j_1) > 0} }
Else Call Three Region Partitioning Method
END Region_Partition
```

Fig. 5. Algorithm of determining the intersection of FDT and FDH

From property 1, we know that the real value $q_1(q_3)$ determines whether the position of FDT(FDH) is on side of the line $d_i(i_1, j_1) \geq 0$ ($d_i(i_2, j_2) \geq 0$) or not. The line is the bounds of two parallel loops.

In this algorithm, the line $d_i(i_1, j_1) = 0$ is expressed by $j = Ai+B$, where $A = (1 - \alpha_{11})/\beta_{11}$, $B = -\gamma_{11}/\beta_{11}$, which are derived from (5). We know that the line can be the upper or lower bound in the transformed loops based on the corresponding region of the loop technique. The line $d_i(i_1, j_1) = 0$ is the upper boundary in AREA2 and lower boundary in AREA1 in Example 1. In this case, the iteration space is divided into two parallel regions, AREA1 and AREA2, by line $j = 3/2*i+1$ as shown in Fig 4. The execution order is AREA1 → AREA2.

Transformed loops are loops are given as follows.

```
/* AREA1 – parallel region */
doall i    l1   u1
  doall j    max(l2,⌈3/2*i+1⌉), u2
       A(2i+3, j+1) = . . .
         . . . = A(i+2j+1, i+j+1)
  enddoall
enddoall
```

```
/* AREA2 – parallel region */
doall i    l1   u1
  doall j    l2   min(u2,⌈3/2*i+1⌉)
     A(2i+3, j+1) = . . .
              . . . = A(i+2j+1, i+j+1)
  enddoall
enddoall
```

4 Performance Analysis

Theoretical speedup for performance analysis can be computed as follows. Ignoring the synchronization, scheduling and variable renaming overheads, and assuming an unlimited number of processors, each partition can be executed in one time step. Hence, the total time of execution is equal to the number of parallel regions, N_p, plus the number of sequential iterations, N_s. Generally, speedup is represented by the ratio of total sequential execution time to the execution time on parallel computer system as follows:

$$Speedup = (N_i * N_j)/(N_p + N_s)$$

where N_i, N_j are the size of loop *i, j,* respectively

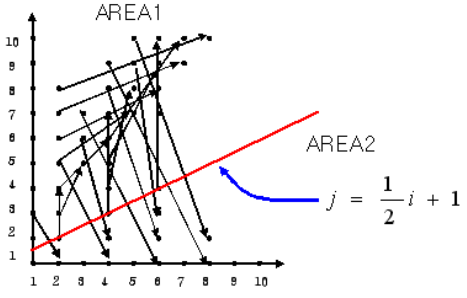

Fig. 6. Regions of the loop partitioned by the unique sets oriented partitioning in Example 1

By using an example given in Example 1, the unique set oriented partitioning method [3] divides the iteration space into one parallel region, AREA2, and one serial region, AREA1, as shown in Fig. 6. So, the speedup is (10*10)/(1+69) =1.4.

Applying the minimum dependence distance tiling method to this loop illustrates case 2 of this technique [4], which is the case that line $d_i(i, j) = 0$ and $d_j(i, j) = 0$ pass through the IDCH. The minimum values of $d_i(i, j)$, d_{imin}, and $d_j(i, j)$, d_{jmin}, occur at the extreme point (1, 1) and both $d_{imin} = 1$ and $d_{jmin} = 1$. There is only serial region, and no speedup for this method.

Our proposed two methods divide the iteration space into two parallel areas by line $j = 1/2*i+1$ and line $j = 3/2*i+1$, respectively. The speedup for these methods is (10*10)/2 = 50.

5 Conclusions

In this paper, we have studied the parallelization of flow and anti dependence loops with non-uniform dependences to improve parallelism.

By variable renaming, there remains only flow dependence sets in the nested loop. We then divide the iteration space into the flow dependence head and tail sets.

We defined the properties of FDT (Flow Dependence Tail set) and FDH (Flow Dependence Head set), and show how to find FDT and FDH in two-dimensional solu-

tion space. We also present two partitioning methods to find two parallel regions in the given space. One is the method by intersection of FDT and FDH. Another is the method by two given equations. Both methods show how to determine whether the intersection of FDT and FDH is empty or not. If FDT does not overlap FDH, a line $d_i(i, j) = 0$ between two sets divides the iteration space into two areas. The iterations within each area can be fully executed in parallel. So, we can find two parallel regions for doubly nested loops with non-uniform dependences.

In comparison with some previous partitioning methods, our proposed methods give much better speedup and extract more parallelism than other methods in the case which FDT does not overlap the FDH. Our future research work is to develop a method for improving parallelization of higher dimensional nested loops.

References

1. V. Kumar, A. Grama, A. Gupta, and G. Karypis, *Introduction to Parallel Computing*, The Benjamin/Cummings Publishing Company, Inc., 1994.
2. C. K. Cho and M. H. Lee, "A loop parallelization method for nested loops with non-uniform dependences", in *Proceedings of the International Conference on Parallel and Distributed Systems,* pp. 314-321, December 10-13, 1997.
3. J. Ju and V. Chaudhary, "Unique sets oriented partitioning of nested loops with non-uniform dependences," in *Proceedings of International Conference on Parallel Processing*, vol. III, pp. 45-52, 1996.
4. S. Punyamurtula and V. Chaudhary, "Minimum dependence distance tiling of nested loops with non-uniform dependences," in *Proceedings of Symposium on Parallel and Distributed Processing*, pp. 74-81, 1994.
5. T. Tzen and L. Ni, "Dependence uniformization: A loop parallelization technique," *IEEE Transactions on Parallel and Distributed Systems*, vol. 4, no. 5, pp. 547-558. May 1993.
6. M. Wolfe and C. W. Tseng, "The power test for data dependence," *IEEE Transactions on Parallel and Distributed Systems*, vol. 3, no. 5, pp. 591-601, September 1992.
7. A. Zaafrani and M. R. Ito, "Parallel region execution of loops with irregular dependences," in *Proceedings of the International Conference on Parallel Processing*, vol. II, pp. 11-19, 1994.