Parallelism for Nested Loops with Non-uniform and Flow Dependences

Sam-Jin Jeong

Dept. of Information & Communication Engineering, Cheonan University, 115, Anseo-dong, Cheonan, Chungnam, 330-180, Korea.
sjjeong@cheonan.ac.kr

Abstract. Many methods are proposed in order to parallelize loops with non-uniform dependence, but most of such approaches perform poorly due to irregular and complex dependence constraints. This paper proposes an efficient method of tiling and transforming nested loops with non-uniform and flow dependences for maximizing parallelism. Our approach is based on the Convex Hull theory that has adequate information to handle non-uniform dependences, and also based on minimum dependence distance tiling, the unique set oriented partitioning, and three region partitioning methods. We will first show how to find the incrementing minimum dependence distance. Next, we will propose how to tile the iteration space efficiently according to the incrementing minimum dependence distance. Finally, we will show how to achieve more parallelism by loop interchanging and how to transform it into parallel loops. Comparison with some other methods shows more parallelism than other existing methods.

1 Introduction

Parallel processing is recognized as an important vehicle for the solution of many areas of computer applications. Most of the computing time is spent in loops in such applications. The existing parallelizing compilers can parallelize most of the loops with uniform dependences, but they do not satisfactorily handle loops with non-uniform dependences. Most of the time, the compiler leaves such loops running sequentially. Unfortunately, loops with non-uniform dependences are not so uncommon in the real world.

Several works have been done for loops with non-uniform dependences. All of the existing techniques do a good job for some particular types of loops, but show us a poor performance on some other types of loops.

Some techniques, based on Convex Hull theory [7] that has been proven to have enough information to handle non-uniform dependences, are the minimum dependence distance tiling method [5], [6], the unique set oriented partitioning method [4], and the three region partitioning [1], [3].

This paper will focus on parallelizing perfectly nested loops with non-uniform and flow dependences.

The rest of this paper is organized as follows. Section two describes our loop model, and reviews some fundamental concepts in non-uniform and flow dependence

M. Bubak et al. (Eds.): ICCS 2004, LNCS 3038, pp. 1146–1152, 2004.

loop. Section three presents an improved tiling method for parallelization with nested loops with non-uniform and flow dependences. In this section, we show how to find the incrementing minimum dependence distances in the iteration space. Then, we discuss how to tile the iteration space efficiently according to the incrementing minimum dependence distance and how to achieve more parallelism by loop interchanging. Section four shows comparison with related works. Finally, we conclude in section five with the direction to enhance this work.

2 Data Dependence Analysis in Non-uniform and Flow Dependence Loop

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. 1.

```
do i = l1, u1
    do j = l2, u2
        A(a11*i + b11*j + c11, a12*i + b12*j + c12) = ...
            ... = A(a21*i + b21*j + c21, a22*i + b22*j + c22)
    enddo
enddo
```

Fig. 1. A doubly nested loop model

The dependence distance function $d(i_1, j_1)$ in flow dependence loops gives the dependence distances $d_i(i_1, j_1)$ and $d_j(i_1, j_1)$ in dimensions i and j, respectively. For uniform dependence vector sets these distances are constant. But, for the non-uniform dependence sets these distances are linear functions of the loop indices. We can write these dependence distance functions in a general form as

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

where p_i, q_i, and r_i are real values and i_1 and j_1 are integer variables of the iteration space.

The properties and theorems for tiling of nested loops with flow dependence can be described as follows.

Theorem 1. *If there is only flow dependence in the loop, DCH1 contains flow dependence tails and DCH2 contains flow dependence heads.*

Theorem 2. *If there is only flow dependence in the loop, then $d_i(x, y) = 0$ or $d_j(x, y) = 0$ does not pass through any DCH.*

If there exists only flow dependence in the loop, then $d_i(x_1, y_1) = 0$ or $d_j(x_1, y_1) = 0$ does not pass through any IDCH(Integer Dependence Convex Hull) because the IDCH is a subspace of DCH(Dependence Convex Hull) [5].

Theorem 3. *If there is only flow dependence in the loop, the minimum and maximum values of the dependence distance function $d(x_1, y_1)$ appear on the extreme points.*

Theorem 4. *If there is only flow dependence in the loop, the minimum dependence distance value d_{imin} is equal or greater than zero.*

From theorem 4, we know that when there is only flow dependence in the loop and d_{imin} is zero, d_{jmin} is greater than zero. In this case, since $d_j(x_1, y_1) = 0$ does not pass through the IDCH, the minimum value of $d_j(x_1, y_1)$, d_{jmin}, occurs at one of the extreme points.

Theorem 5 *If there is only flow dependence in the loop, the difference between the distance of a dependence and that of the next dependence, d_{inc}, is equal to or greater than zero.*

Thus, d_{inc} is equal to to or greater than zero when there is only flow dependence in the loop.

3 Improved Tiling Method

Cho and Lee [2] present a more general and powerful loop splitting method to enhance all parallelism on a single loop. The method uses more information from the loop such as increment factors, and the difference between the distance of dependence, and that of the next dependence. Cho and Lee [3] derive an efficient method for nested loops with simple scripts from enhancing [2].

The minimum dependence distance tiling method [6] presents an algorithm to convert the extreme points with real coordinates to the extreme points with integer coordinates. The method obtains an IDCH from a DCH. It can compute d_{imin}, the minimum value of the dependence distance function $d_i(i_1, j_1)$ and d_{jmin}, the minimum value of the dependence distance function $d_j(i_1, j_1)$ from the extreme points of the IDCH. The first minimum dependence distances d_{imin} and d_{jmin} are used to determine the uniform tile size in the iteration space.

3.1 Tiling Method by the Incrementing Minimum Dependence Distance

From theorem 5, when $p_1 > 0$ and $q_1 \geq 0$, we know that the difference between the distance of a dependence and that of the next dependence in loop with flow dependence, d_{inc}, is equal to or greater than zero.

For each i_1, d_{imin} is incremented as the value of i_1 is incremented. So, the second d_{imin} is equal to or greater than the first one, and the third one is greater than the second one, and so on.

The improved tiling method for doubly nested loops with non-uniform and flow dependence is described as **Procedure Tiling_Method**, which is the algorithm of tiling loop by the incrementing minimum dependence distance as shown in Fig. 2.

This algorithm computes the incrementing minimum dependence distance, tiles the iteration space efficiently according to the incrementing minimum dependence distance, and transforms it into parallel loops.

```
Procedure Tiling_Method(i1, j1, l1, l2, u1, u2, di(i1, j1))
  i1, j1: i and j value for the source of the first minimum
    dependence in the loop computed by the extreme points
    of the IDCH
  l1, l2, u1, u2: the lower and upper bounds of outer loop
    and inner loop, respectively
  di(i1, j1): the dependence distance function of the IDCH
  begin
    Step 1: when the first source point, (i1, j1), is given,
      the first minimum dependence distance dimin and first
      tile size are computed.
    Step 2: Next dimin is computed.
    If (next sink point is greater than bound), Goto Step
      4.
    Step 3: Next tile size is computed, and Goto Step 2.
    Step 4: the original loop is transformed into n
      parallel tiles.
  end Tiling_Method.
```

Fig. 2. Algorithm of tiling loop by the incrementing minimum dependence distence

Example 1

```
do i = 1, 50
    do j = 1, 50
        A(3*i+1, 4*i+2*j+1) = . . .
            . . .= A(2*i-4, i+j-4)
    enddo
enddo
```

An example given in Example 1 illustrates the case that there is non-uniform and flow dependence. Fig. 3(a) shows CDCH(Complete Dependence Convex Hull) of Example 1. As the example, we can obtain the following results using the improved tiling method proposed in this section**.**

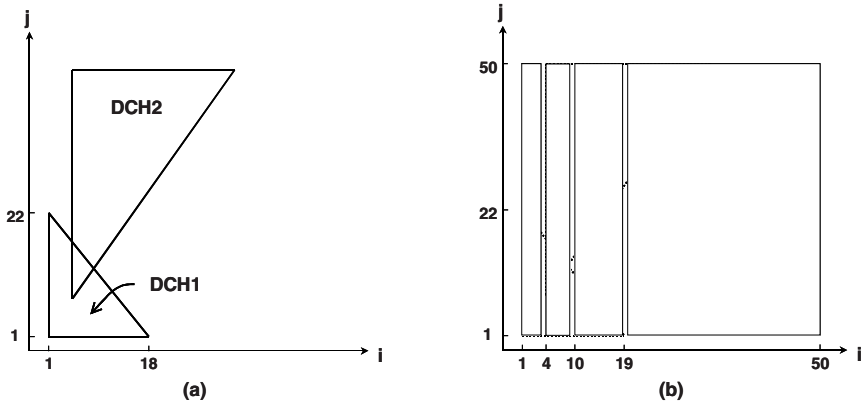

Fig. 3. (a) CDCH, (b) Tiling by minimum dependence distance in Example 1.

From the algorithm to compute a two-dimensional IDCH in [5], we can obtain the extreme points such as (1, 1), (1, 22), and (18, 1) as shown in Fig. 3(a). The first minimum value of $d_i(i_1, j_1)$ occurs at one of the extreme points. The i value for the source of the first dependence in the second tile is 4. The i value in the third tile is 10, and next values are 19, 31, and 49. Then, we can divide the iteration space by four tiles as shown in Fig. 3(b).

3.2 Loop Tiling Method Using Loop Interchanging

When there is only flow dependence in the loop, we can tile the iteration space into tiles with width = d_{imin} or width = d_{jmin}. In case $d_{jmin} > d_{imin}$, we can tile the iteration space into tiles with width = d_{jmin}.

In Example 1, because $d_j(i_1, j_1)$ (= 5/2*i + j + 5/2) is greater than $d_i(i_1, j_1)$ (= 1/2*i + 5/2), we can use an changed form of the example that the outer loop i and the inner loop j are interchanged as shown in Fig. 4.

```
do j = 1, 50
    do i = 1, 50
        A(3*i+1, 4*i+2*j+1) = . . .
            . . .= A(2*i-4, i+j-4)
    enddo
enddo
```

Fig. 4. Another form of Example 1 by loop interchanging.

If the upper limits of loop i and j are 100 by 100 as an example given in Fig. 4, the number of tiles for the original loop is six as shown in Fig. 5(a), and for the interchanged loop is five as shown in Fig. 5(b). When $d_{jmin} > d_{imin}$ in this loop, we can achieve greater parallelism by loop interchanging.

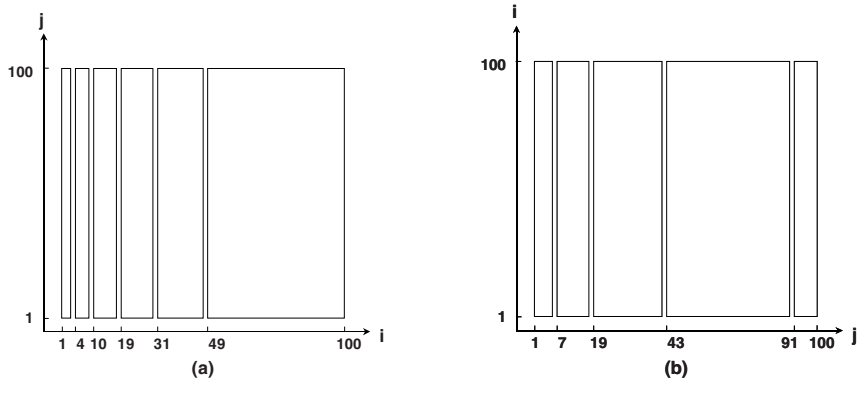

Fig. 5. (a) Tiling by the incrementing minimum dependence distance, (b) Tiling by Loop Interchanging in Example 1.

4 Performance Analysis

This section discusses the performance analysis of our proposed methods through the comparisons with related works theoretically.

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

We will compare our proposed methods with the minimum dependence distance tiling method and the unique set oriented partitioning method as follows:

Let's consider the loop shown in Example 1. Fig. 3(a) shows original partitioning of Example 1. This example is the case that there is only flow dependence and DCH1 overlaps DCH2. Applying the unique set oriented partitioning to this loop illustrates case 2 of [4]. This method can divide the iteration space into four regions: three parallel regions, AREA1, AREA2 and AREA4, and one serial region, AREA3, as shown in Fig. 6. The speedup for this method is (100*100)/(3+44) = 212.8.

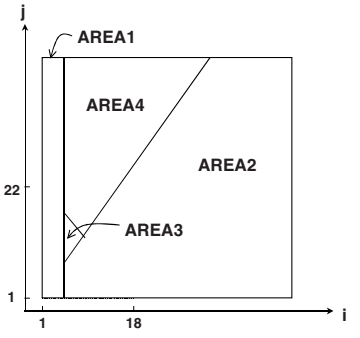

Fig. 6. Regions of the loop partitioned by the unique sets oriented partitioning in Example 1.

Applying the minimum dependence distance tiling method to this loop illustrates case 1 of this technique [5], which is the case that line $d_i(i, j) = 0$ does not pass through the IDCH. The minimum value of $d_i(i, j)$, d_{imin}, occurs at the extreme point (1, 1) and $d_{imin} = 3$. The space can be tiled with width = 3, thus 34 tiles are obtained. The speedup for this method is (100*100)/34 =294.

Let's apply our proposed method - the improved tiling method as given in section 3. This loop is tiled by six areas as shown in Fig. 5(a). The iterations within each area can be fully executed in parallel. So, the speedup for this method is (100*100)/6 = 1666.

Applying the loop interchanging method in this example, this loop is tiled by five areas as shown in Fig. 6(b). So, the speedup for this method is (100*100)/5 = 2000.

If the upper bounds of loop i, j are 1000 by 1000, the speedup for the original loop is (1000*1000)/11 = 90909, and the speedup for the interchanged loop is (1000*1000)/8 = 125000. Because $d_{jmin} > d_{imin}$ in this example, we can achieve more parallelism by loop interchanging.

5 Conclusions

In this paper, we have studied the problem of transforming nested loops with non-uniform and flow dependences to maximize parallelism.

When there is only flow dependence in the loop, we propose the improved tiling method. The minimum dependence distance tiling method tiles the iteration space by the first minimum dependence distance uniformly. Our proposed method, however, tiles the iteration space by minimum dependence distance values that are incremented as the value of i_1 is incremented. Furthermore, when $d_{jmin} > d_{imin}$ in the given loop, loop parallelism can be improved by loop interchanging.

In comparison with some previous partitioning methods, the improved tiling method gives much better speedup than the minimum dependence distance tiling method and the unique set oriented partitioning method in the case that there is only flow dependence and DCH1 overlaps DCH2.

Our future research work is to develop a method for improving parallelization of higher dimensional nested loops.

References

1. A. 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, (1994) 11-19
2. C. K. Cho, J. C. Shim, and M. H. Lee, "A loop transformation for maximizing parallelism from single loops with non-uniform dependences," in *Proceedings of High Performance Computing Asia '97*, (1997) 696-699
3. C. K. Cho and M. H. Lee, "A loop parallization method for nested loops with non-uniform dependences", in *Proceedings of the International Conference on Parallel and Distributed Systems*, (1997) 314-321
4. 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, (1996) 45-52
5. 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*, (1994) 74-81
6. S. Punyamurtula, V. Chaudhary, J. Ju, and S. Roy, "Compile time partitioning of nested loop iteration spaces with non-uniform dependences," *Journal of Parallel Algorithms and Applications,* (1996)
7. T. Tzen and L. Ni, "Dependence uniformization: A loop parallelization technique," *IEEE Transactions on Parallel and Distributed Systems,* vol. 4, no. 5, (1993) 547-558