Parallel Programming WS 2014/2015 - Solution 5

Kastens, Pfahler
Institut für Informatik, Fakultät für Elektrotechnik, Informatik und Mathematik, Universität Paderborn
Jan 05, 2015

Solution for Exercise 4

Original loop:

```
for I = 1 to 3 do
  for J = 1 to 3 do
    A[I, J] = A[I - 1, J + 1] + 1
  endfor
endfor
```

a) The iteration space including a dependence vector.

```
    x x x
    x x x
    x x x

D = 1
   -1
```

b) It is illegal to permute the I and J loops because the resulting dependence vector is negative:

```
T = 0 1
    1 0

T * D = 0 1 * 1  = -1
        1 0  -1     1
```

c) Applying a skewing transformation with factor 1:

```
T = 1 0
    1 1

T * D = 1 0 *  1  = 1
        1 1   -1    0
```

Legal.

d) The resulting iteration space including a dependence vector:

```
    x       coord (3,6)
  x x
x x x
x x
x           coord (1,2)

D = 1
    0
```

e) Permuting the I and J loops of the resulting code:

```
T = 0 1
    1 0

T * D = 0 1 * 1  = 0
        1 0   0    1
```

Legal.

f) The resulting iteration space including a dependence vector:

```
    x x x      coord (4,3) - coord(6,3)
  x x x
x x x          coord (2,1) - coord(4,1)
```

```
D = 0
    1
```

g) The loop bounds of the transformed loop derived from the drawing:

```
i’ runs from 2 to 6
j’ runs from max(1, i’ - 3) to min(3, i’ - 1)
```

h) Homework: Deriving the loop bounds mathematically:

```
Skewing matrix S:     1 0
                      1 1

Permutation matrix:   0 1
                      1 0

Overall transformation:
T = P * S:            1 1
                      1 0
```

The inverse transformation matrix T^{-1}:

```
0 1
1 -1
```

The bounds equation derived from the original program is

```
B * i  <=  -1
    j       3
           -1
            3
```

==>

```
-1  0  * i  <=  -1
 1  0    j       3
 0 -1           -1
 0  1            3
```

The bounds equation derived for the transformed program is

```
B * T^-1  * i’  <=  -1
            j’       3
                    -1
                     3
```

==>

```
 0 -1  * i’  <=  -1
 0  1    j’       3
-1  1            -1
 1 -1             3
```

which yields

```
j’ >= 1
j’ <= 3
j’ - i’ <= -1
i’ - j’ <= 3
```

==>

```
2 <= i’ <= 6
j’ >= max (1, i’ - 3)
j’ y= min (3, i’ - 1)
```

which corresponds to the bounds we derived graphically. Additionally we compute the transformed loop body:

The old iteration variables in terms of the new ones:

```
   T -1  * i'  =  i
           j'     j
```

==>

```
   0  1  * i' = i
   1 -1    j'   j
```

==>

```
   i = j'
   j = i' - j'
```

Together the transformed program looks as follows:

```
   for i' = 2 to 6 do
     for j' = max (1, i' - 3) to min (3, i' - 1) do
       A[j', i' - j'] = A[j' - 1, i' - j' + 1] + 1
     endfor
   endfor
```

It computes the array elements in the order

`A[11], A[12], A[21], A[13], A[22], A[31], A[23], A[32], A[33]`

Solution for Exercise 5

```
   for I = 0 to N do
     for J = 0 to M do
       A[I + 1, J] = A[I, J - 1] + A[I + 1, J - 2]
     endfor
   endfor
```

These are the necessary steps:

- 1. Draw the iteration space.

  A rectangle from (0,0) to (3,3):

  ```
  x x x x
  x x x x
  x x x x
  x x x x
  ```

- 2. Compute the dependence vectors and draw examples of them into the iteration space.

  The dependence vectors are $(1, 1)^T$ and $(0, 2)^T$.
- 3. Apply a skewing transformation with factor 1 and draw the iteration space.

  ```
        x
      x x
    x x x
  x x x x
  x x x
  x x
  x
  ```

- 4. Apply a permutation transformation and draw the iteration space.

  ```
        x x x x
      x x x x
    x x x x
  x x x x
  ```

- 5. Compute the matrix of the composed transformation and use it to transform the dependence vectors.

```
Skewing matrix S:    1 0
                     1 1

Permutation matrix:  0 1
                     1 0

T = P * S:           1 1
                     1 0

Transformed dependence vectors T * D:
 1 1       1 0       2 2
      *         =
 1 0       1 2       1 0
```

- 6. Compute the inverse of the transformation matrix and use it to transform the index expressions.

  Inverse Transformation Matrix T^{-1}:

  ```
  0 1
  1 -1
  ```

  Transformation of index expressions:

  ```
  0 1       ip       i
       *         =
  1 -1      jp       j
  ```

  yields `i = jp` and `j = ip - jp`.

- 7. Specify the loop bounds by inequalities and transform them by the inverse of the transformation matrix.

  B * T^{-1} * IP <= c

  ```
  -1 0                                0
  1  0       0 1         ip           N
        *            *          <=
  0 -1       1 -1        jp           0
  0  1                                M
  ```

  simplifies to

  ```
  0 -1                   0
  0  1       ip          N
       *            <=
  -1 1       jp          0
  1  -1                  M
  ```

  such that

  ```
  0 <= ip <= M + N
  max(0, ip - M) <= jp <= min (ip, N)
  ```

- 8. Write the complete loops with new loop variables ip and jp and new loop bounds.

  ```
  for I = 0 to M+N do
    for J = max(0, I-M) to min(I, N) do
      A[J + 1, I - J] = A[J, I - J - 1] + A[J + 1, I - J - 2]
    endfor
  endfor
  ```

If N = M = 2 the transformed loop executes the computations:

```
s00: A[1, 0] = A[0, -1] + A[1, -2]
s10: A[1, 1] = A[0, 0] + A[1, -1]
s11: A[2, 0] = A[1, -1] + A[2, -2]
s20: A[1, 2] = A[0, 1] + A[1, 0]
s21: A[2, 1] = A[1, 0] + A[2, -1]
s22: A[3, 0] = A[2, -1] + A[3, -2]
s31: A[2, 2] = A[1, 1] + A[2, 0]
s32: A[3, 1] = A[2, 0] + A[3, -1]
s42: A[3, 2] = A[2, 1] + A[3, 0]
```

These are the same computations that are executed by the original loop. The computations of the inner loop are parallelizable.