CS 526

Advanced Compiler Construction

http://misailo.cs.Illinois.edu/courses/cs526

DEPENDENCE TRANSFORMS

The slides adapted from Vikram Adve

Loop Skewing

Informal Definition: Increase dependence distance by n by substituting loop index j with jj = j + n ∗ i.

E.g., For n = 1, we use jj = j + 1

```
do i=2,N
  do j=2,N
    A[i,j] = A[i-1,j]
           + A[i,j-1]
  enddo
enddo
```

```
do i=2,N
  do jj=i+2,i+N
    A[i,jj-i] = A[i-1,jj-i]
              + A[i,jj-i-1]
  enddo
enddo
```

Uses of Loop Skewing

- Improve parallelism by converting '=' to '+' in a direction vector
- Improve vectorization in a similar way
- (Rarely) Could be used to *simplify* index expressions

Loop Distribution

Informal Definition: Convert a loop nest containing two or more statements into two or more distinct loop nests so that each statement appears in only a single resulting loop nest.

```
      do i=2,N
S1:     A[i] = B[i] + C[i]
S2:     D[i] = A[i] * 2.0
S3:     B[i+1] = A[i] * 3.0
     enddo
```

```
      do i=2,N
S1:     A[i] = B[i] + C[i]
S3:     B[i+1] = A[i] * 3.0
     enddo
     do i=2,N
S2:    D[i] = A[i] * 2.0
     enddo
```

Loop Distribution Applications

- Create perfect loops nests for other transformations like loop interchange
- Convert a loop-carried dependence within a loop into a loop-independent dependence crossing two loops:

```
     do i=2,N
S1:      A[i] = B[i] + C[i]
S2:      D[i] = A[i-1] * 2.0
     enddo
```

```
     do i=2,N
S1:     A[i] = B[i] + C[i]
     enddo
     do i=2,N
S2:     D[i] = A[i-1] * 2.0
     enddo
```

Maximal Loop Distribution

- Identify the SCCs of the data dependence graph, to group statements in an SCC in a single loop nest
- Sort the SCCs using a topological sort on the dependence graph
- Generate distinct loop nests, one for each SCC, in sorted order

Loop Fusion

Informal Definition: Merge two or more distinct (perhaps non-adjacent) loops with identical loop bounds into a single loop.

```
do i=1,N
   A[i] = i*i
enddo
do i=1,N
   B[i] = A[i] + 1
enddo
```

```
do i=1,N
   A[i] = i*i
   B[i] = A[i] + 1
enddo
```

Loop Fusion

```
do i=1,M
  do j=1,N-1
    A[j,i] = i*i + j*j
  enddo

  do j=1,N
    B[j,i] = A[j,i] + i + j
  enddo
enddo
```

```
do i=1,M
  do j=1,N-1
    A[j,i] = i*i + j*j
    B[j,i] = A[j,i] + i + j
  enddo
  // peel last iteration:
    j=N
    B[j,i] = A[j,i] + i + j
enddo
```

Loop Fusion Motivation

- Increase cache reuse (if same array accessed in two loops) Fundamental optimization for array languages (e.g., Fortran 90, HPF, MATLAB, APL)

  Example in F90:

```
A[1:M, 1:N] = B[1:M, 1:N] * 2
C[1:M, 1:N] = A[1:M, 1:N] + 1
```

- Increase granularity of parallelism (work per iteration) Important for shared-memory parallelism (the model with parallel loop and barriers)

Legality of Loop Fusion

Fusion-Preventing Dependence: A loop-independent dependence from S1 to S2 in different loops is fusion-preventing if fusing the two loops causes the dependence to become a loop-carried dependence from S2 to S1.

Legality of Loop Fusion: Two loops can be fused if all 3 conditions are satisfied:

1. Both have identical bounds (*transform loops if needed*)
2. There is no fusion-preventing dependence between them.
3. There is no path of loop-independent dependences between them that contains a loop or statement that is not being fused with them.

Loop Fusion: Illegal Cases

```
do i=1,M
  do j=2,N
    A[j,i] = B[j-1,i] * 2
  enddo

  do j=2,N
    B[j,i] = A[j,i] * 3
  enddo
enddo
```

```
do i=1,M
  do j=2,N
    t[j] = B[j-1,i]
  enddo

  do j=2,N
    A[j,i] = t[j] * 2
    B[j,i] = A[j,i] * 3
  enddo
enddo
```

Create temporary array to make fusion possible

Loop Strip Mining

Informal Definition Convert a single loop into two nested loops for a specified "block size" (*Always safe.*)

```
do i=1,N
     A[i] = x + B[i] * 2
enddo
```

```
do ii=1,N,B
   do i=ii, min(ii+B-1, N), 1
       A[i] = x + B[i] * 2
   enddo
enddo
```

Loop Strip Mining Applications

- **Loop tiling:** strip-mine and then interchange multiple uses. Can be useful for increasing cache locality or blocking parallel loops;
- **Prefetching:** strip-mine by cache line size; prefetch once per outer iteration
- **Instruction scheduling:** strip-mine and then unroll inner loop

Loop Alignment

Informal Definition: Eliminate a carried dependence by increasing the number of iterations and executing statements on different subsets of the iterations (*Always safe)*

```
do i=2,N
   A[i] = B[i] + C[i]
   D[i] = A[i-1] * 2.0
enddo
```

```
i = 1
D[i+1] = A[i] * 2

do i=2,N
   A[i] = B[i] + C[i]
   D[i+1] = A[i] * 2.0
enddo

i = N
A[i] = B[i] + C[i]
```

Unroll and Jam

Informal Definition: Unroll the outer loop by k, then fuse the resulting k inner loops into a single loop

```
do i = 1, n
   do j = 1, n
      a(i) = a(i) + b(j)
   enddo
enddo
```

```
do i = 1, n, 2
   do j = 1, n
      a(i) = a(i) + b(j)
      a(i+1) = a(i+1) + b(j)
   enddo
enddo
```