CS 526

Advanced
Compiler
Construction

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

DEPENDENCE TRANSFORMS

The slides adapted from Vikram Adve and David Padua

Motivation

Memory hierarchy optimizations
Goal 1: Improving reuse of data values within loop nest
Goal 2: Exploit reuse to reduce cache, TLB misses

Tiling
Goal 1: Exploit temporal reuse when data size > cache size
Goal 2: In parallel loops, reduce synchronization overhead

Software Prefetching
Goal: Prefetch predictable accesses k iterations ahead

Software Pipelining
Goal: Extract ILP from multiple consecutive iterations

Automatic parallelization Also, auto-vectorization
Goal 1: Enhance parallelism
Goal 2: Convert scalar loop to explicitly parallel
Goal 3: Improve performance of parallel code

Loop Interchange

Informal Definition: Change nesting order of loops in a **perfect loop nest**, with no other changes.

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

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

Uses of Loop Interchange

1. Move independent loop innermost
2. Move independent loop outermost
3. Make accesses stride-1 in memory
4. Loop tiling (combine with strip-mining)
5. Unroll-and-jam (combine with unrolling)

Loop Interchange

Direction Vectors and Loop Interchange:

If δ is a direction vector of a particular dependence S1 → S2 in a loop nest and the order of loops in the loop nest is permuted, then the same permutation can be applied to δ to obtain the new direction vector for the conflicting instances of S1 and S2

Direction Matrix: A matrix where each row is the direction vector of a single dependence, i.e.,

each row ↔ a dependence

each column ↔ a loop

Direction Matrix

Direction Matrix:

each row ↔ a dependence

each column ↔ a loop

A[i,j]/A[i,j]	= =
A[i,j]/A[i-1,j]	+ =
B[I,j]/B[i-1,j-1]	+ +

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

Direction Matrix (Illegal)

Direction Matrix:

each row ↔ a dependence

each column ↔ a loop

A[i,j]/A[i,j]	=	=
A[i,j]/A[i-1,j]	+	-
B[1,j]/B[i-1,j-1]	+	+

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

Loop Interchange Properties

Legality: A permutation of the loops in a perfect nest is legal iff the direction matrix, after the permuation is applied, has no “–” direction as the leftmost non-“=“ direction in any row

Profitability: machine-dependent:

1. vector machines
2. parallel machines
3. caches with single outstanding loads
4. caches with multiple outstanding loads

Applying Loop Interchange

1. Single ’+’ entry: a “serial loop”
   - Move loop outermost for vectorization
   - Move loop innermost for parallelization

2. Multiple ’+’ entries: Outermost one carries dependence
   - Loop carrying the dependence *changes* after permutation!
   - May still benefit by moving carried-dependences to outermost loop

Loop Reversal

Informal Definition: Reverse the order of execution of the iterations of a loop

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

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

Loop Reversal

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

=	+	-
+	+	-

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

=	+	+
+	+	+

Uses of Loop Reversal

Convert a ’>’ to a ’<’ in a direction vector to enable other transformations, e.g., loop interchange.

Scalarize a vector statement (e.g., in Fortran 90) by ensuring that values are read before being written.

- Vectorized code: `A[2:64] = A[1:63] * e`
- Scalarized code:

```
do i = 64, 2, -1
    A[i] = A[i-1] * e
enddo
```

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

Unimodular Loop Transformations

These transfomations can be represented by a unimodular transformation matrix T.

For Loop Interchange

$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

For Loop Reversal

$$\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

For Loop Skewing

$$\begin{pmatrix} 1 & 0 \\ \alpha & 1 \end{pmatrix}$$