Dependence: Theory and Practice

Introduction to loop dependence and loop transformation

The Big Picture

What are our goals?

- Find independent operations to evaluate in parallel
- Find operations that reuse the same data

What we will cover?

- Introduction to Dependences
- Loop-carried and Loop-independent Dependences
- Parallelization and Vectorization (details skipped)
- Simple Dependence Testing (details skipped)

- This chapter concentrates on data dependences
  - Chapter 7 deals with control dependences

Data Dependences

- There is a data dependence from statement S_1 to S_2 if:
  1. Both statements access the same memory location,
  2. At least one of them stores onto it, and
  3. There is a feasible run-time execution path from S_1 to S_2
- Classification of data dependence
  - True dependences (Read After Write hazard)
    S_2 depends on S_1 is denoted by $S_1\ \delta\ S_2$
  - Anti dependence (Write After Read hazard)
    S_2 depends on S_1 is denoted by $S_1\ \delta^{-1}\ S_2$
  - Output dependence (Write After Write hazard)
    S_2 depends on S_1 is denoted by $S_1\ \delta^{0}\ S_2$
- Simple example of data dependence:

```
S1  PI = 3.14
S2   R = 5.0
S3   AREA = PI * R ** 2
```

Transformations

- A transformation is safe if the transformed code has the same "meaning" as the original program
  - Two computations are equivalent if they always produce the same outputs on the same inputs:
- A reordering Transformation
  - Changes the execution order of the code, without adding or deleting any operations.
- Properties of Reordering Transformations
  - It does not eliminate dependences, but can change the ordering (relative source and sink) of a dependence
  - If a dependence is reverted by a reordering transformation, it may lead to incorrect behavior
- A reordering transformation is safe if it preserves the relative direction (i.e., the source and sink) of each dependence.

Dependence in Loops

```
    DO I = 1, N
S1    A(I+1) = A(I) + B(I)
    ENDDO
```

```
    DO I = 1, N
S1    A(I+2) = A(I) + B(I)
    ENDDO
```

- In both cases, statement S1 depends on itself
  - However, there is a significant difference
- We need to distinguish different iterations of loops
  - The iteration number of a loop is equal to the value of the loop index (loop induction variable)
  - Example:

```
    DO I = 0, 10, 2
S1      <some statement>
    ENDDO
```

- What about nested loops?
  - Need to consider the nesting level of a loop

Iteration Vectors

- Given a nest of n loops, iteration vector *i* is
  - A vector of integers $\{i_1, i_2, ..., i_n\}$ where i_k, $1 \le k \le n$ represents the iteration number for the loop at nesting level k
- Example:

```
DO I = 1, 2
   DO J = 1, 2
S1        <some statement>
   ENDDO
ENDDO
```

  - The iteration vector (2, 1) denotes the instance of S_1 executed during the 2nd iteration of the I loop and the 1st iteration of the J loop

The Iteration Space

- Ordering of Iteration Vectors (lexicographic order)
  - Iteration i precedes iteration j, denoted i < j, iff for some nesting level k

    1. i[i:k-1] < j[1:k-1], or

    2. i[1:k-1] = j[1:k-1] and $i_n < j_n$
  - Example: (1,1) < (1,2) < (2,1) < (2,2)
- Iteration Space
  - The set of all possible iteration vectors for a statement
  - Example:

```
DO I = 1, 2
  DO J = 1, 2
S1      <some statement>
  ENDDO
ENDDO
```

    The iteration space for S1 is { (1,1),(1,2),(2,1),(2,2) }

Formal Definition of Loop Dependence

Theorem 2.1 Loop Dependence:

There exists a dependence from statement S1 to S2 in a common nest of loops if and only if

- there exist two iteration vectors i and j for the nest, such that
  - (1) i < j or i = j and there is a path from S1 to S2 in the body of the loop,
  - (2) statement S1 accesses memory location M on iteration i and statement S2 accesses location M on iteration j, and
  - (3) one of these accesses is a write.

Follows the formal definition of dependence

Distance and Direction Vectors

- Consider a dependence in a loop nest of n loops
  - Statement S1 on iteration i is the source of dependence
  - Statement S2 on iteration j is the sink of dependence
- The distance vector is a vector of length n d(i,j) such that: $d(i,j)_k = j_k - I_k$
- The direction Vector is a vector of length n D(i,j) such that (Definition 2.10 in the book)

$$D(i,j)_k = \begin{cases} \text{“<”} & \text{if } d(i,j)_k > 0 \\ \text{“=”} & \text{if } d(i,j)_k = 0 \\ \text{“>”} & \text{if } d(i,j)_k < 0 \end{cases}$$

- What is the dependence distance/direction vector?

```
DO I = 1, N
  DO J = 1, M
    DO K = 1, L
S1      A(I+1, J, K-1) = A(I, J, K) + 10
```

Direction Vector Transformation

- A dependence cannot exist if it has a direction vector whose leftmost non "=" component is “>”
  - as this would imply that the sink of the dependence occurs before the source.
- Theorem 2.3. Direction Vector Transformation.
  - Let T be a loop reordering transformation that does not rearrange the statements in the loop body. The transformation is valid if, after it is applied, none of the dependence direction vectors has a leftmost non- “=” component that is “>”.
- Follows Fundamental Theorem of Dependence:
  - All dependences remain after transformation
  - None of the dependences have been reversed

Loop-carried and Loop-independent Dependences

- If in a loop statement S2 on iteration ***j*** depends on S1 on iteration ***i***, the dependence is
  - Loop-carried (Definition 2.11) if any of the following equivalent conditions is satisfied
    - S1 and S2 execute on different iterations i.e., $i \neq j$
    - $d(i,j) > 0$ i.e. $D(i,j)$ contains a "<" as leftmost non "=" component
  - Loop-independent (Definition 2.14) if any of the following equivalent conditions is satisfied
    - S1 and S2 execute on the same iteration i.e., $i=j$
    - $d(i,j) = 0$, i.e. $D(i,j)$ contains only "=" component
    - NOTE: there must be a control-flow path from S1 to S2 within the same iteration
- Example:

```
DO I = 1, N
S1      A(I+1) = F(I)+ A(I)
S2      F(I) = A(I+1)
ENDDO
```

Level of loop dependence

- The level of a loop-carried dependence is the index of the leftmost non-"=" of D(i,j)
  - A level-k dependence from S_1 to S_2 is denoted $S_1 \delta_k S_2$
  - A loop independent dependence from S1 to S_2 is denoted $S_1 \delta_\infty S_2$
- Example:

```
DO I = 1, 10
   DO J = 1, 10
      DO K = 1, 10
S1      A(I, J, K+1) = A(I, J, K)
S2      F(I,J,K) = A(I,J,K+1)
      ENDDO
   ENDDO
ENDDO
```

- Loop-carried Transformations(Theorem 2.4)
  - Any reordering transformation that
    (1) does not alter the relative nesting order of loops and
    (2) preserves the iteration order of the level-k loop
    preserves all level-k dependences.

Parallelization and Vectorization

- Theorem 2.8. It is valid to convert a sequential loop to a parallel loop if the loop carries no dependence.

- It is safe to convert loop:

```
DO I=1,N
     X(I) = X(I) + C
ENDDO
```

  to **X(1:N) = X(1:N) + C** (Fortran 77 to Fortran 90)

- However:

```
DO I=1,N
     X(I+1) = X(I) + C
ENDDO
```

  is not equivalent to **X(2:N+1) = X(1:N) + C**

Simple Dependence Testing

```
DO i1 = L1, U1, S1
    DO i2 = L2, U2, S2
      …
        DO in = Ln, Un, Sn
S1        A(f1(i1,…,in),…,fm(i1,…,in)) = …
S2      … = A(g1(i1,…,in),…,gm(i1,…,in))
        ENDDO
      …
    ENDDO
ENDDO
```

- A dependence exists from S1 to S2 if and only if there exist values of a and b such that

  (1) a is lexicographically less than or equal to b and

  (2) the system of dependence equations is satisfied:
  $f_i(a) = g_i(b)$ for all i, $1 \le i \le m$
- Direct application of Loop Dependence Theorem

Summary

- Introducing data dependence
  - What is the meaning of S2 depends on S1?
  - What is the meaning of $S_1 \delta S_2$, $S_1 \delta^{-1} S_2$, $S_1 \delta^0 S_2$?
  - What is the safety constraint of reordering transformations?
- Loop dependence
  - What is the meaning of iteration vector (3,5,7)?
  - What is the iteration space of a loop nest?
  - What is the meaning of iteration vector I < J?
  - What is the distance/direction vector of a loop dependence?
  - What is the relation between dependence distance and direction?
  - What is the safety constraint of loop reordering transformations?
- Level of loop dependence and transformations
  - What is the meaning of loop carried/independent dependences?
  - What is the level of a loop dependence or loop transformation?
  - What is the safety constraint of loop parallelization?
- Dependence testing theory