CSC2/458 Parallel and Distributed Systems Automated Parallelization in Software (Contd.)

Sreepathi Pai

February 1, 2018

URCS

Outline

Characterizing loop dependences

Identifying Loop Dependences

Current Loop Optimizations

Outline

Characterizing loop dependences

Identifying Loop Dependences

Current Loop Optimizations

Why characterize dependences?

- The definition of dependence that we have used so far:
  - Two statements have a dependence if:
  - Both access same location (memory or register)
  - And one of the accesses is a write
- This is not sufficient to reason about dependences in loops
- We will extend this definition of dependences to *loop dependence*
  - Study additional characteristics of dependences

Already encountered characteristics of dependences

- True dependence
  - $S_1 \delta S_2$
  - S_1 writes, S_2 reads
- Anti-dependence
  - $S_1 \delta^{-1} S_2$
  - S_1 reads, S_2 writes
- Output dependence
  - $S_1 \delta^o S_2$
  - Both S_1 and S_2 write

Loop-independent dependence

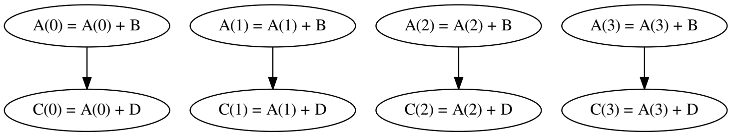
- What are the dependences in the loop body below?
- Can you change the order of the statements in the loop body?

```
DO I = 1, 10
  A(I) = A(I) + B
  C(I) = A(I) + D
ENDDO
```

- Can you change the (execution) order of loop iterations?

Note: FORTRAN uses parentheses in array references: e.g., `A(I)`

Loop-independent dependences visualized



NOTE: Only dependences from first four iterations visualized.

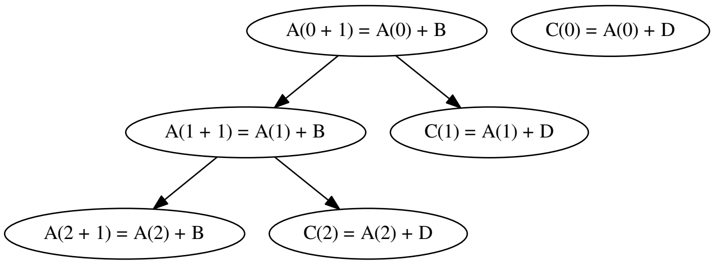
Loop-carried dependences

- What are the dependences in the loop body below?
- Can you change the order of the statements in the loop body?

```
DO I = 1, 10
  A(I + 1) = A(I) + B
  C(I) = A(I) + D
ENDDO
```

- Can you change the (execution) order of loop iterations?

Loop-carried dependences visualized



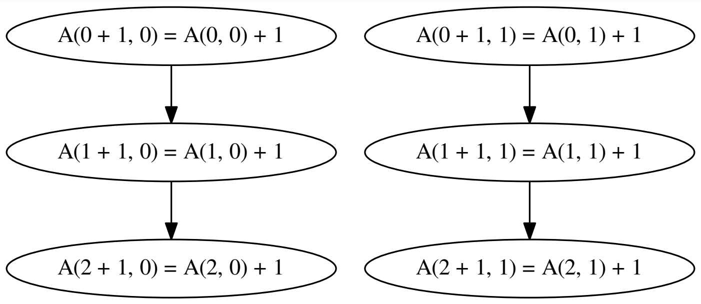
NOTE: Only dependences from first three iterations visualized.

Dependence Level for Loop-Carried Dependences

```
DO I = 1, 10
   DO J = 1, 2
      A(I + 1, J) = A(I, J) + 1
   ENDDO
ENDDO
```

- Can you change the order of inner loop?
- Can you change the order of the outer loop?

Dependences Visualized



NOTE: Only dependences from first three iterations visualized.

Loop Dependences

- Loop-independent dependence
  - In same iteration, independent of loops
- Loop-carried dependence
  - Across different iterations of atleast one loop
- Dependence Level of a Loop-carried Dependence
  - The nesting level k of loop that carries the dependence
  - $S_1 \delta_k S_2$

Iteration Spaces

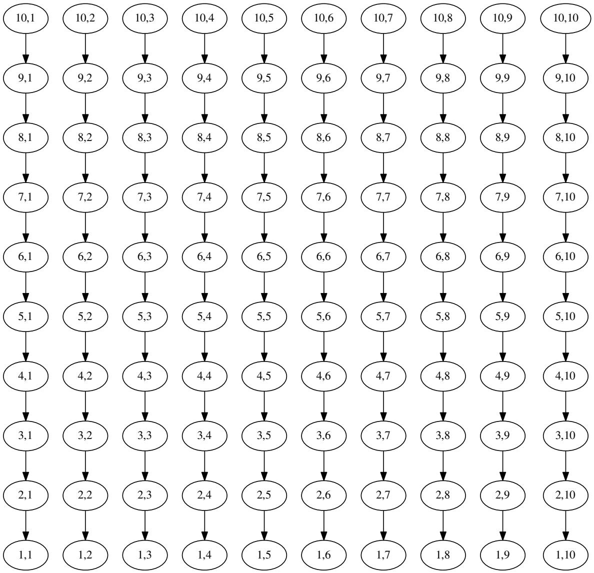
```
DO I = 1, 2
   DO J = 1, 2
      S
   ENDDO
ENDDO
```

- `S` has four instances (I, J): $(1,1), (1,2), (2,1), (2,2)$
- Each of these values represents an *iteration vector*
    - Particular values of loop indices from outermost loop to innermost loop

Iteration Space Example

```
DO J = 1, 10
  DO I = 1, 10
   A(I+1, J) = A(I, J) + X
  ENDDO
ENDDO
```

Iteration Space Figure



Iteration Vector Ordering

For two vectors i and j, each containing n elements, $i < j$ is defined as:

```
def lessthan(i, j, n):
    if n == 1:
        return i[0] < j[0]

    # test prefix for elementwise-equality
    if i[0:n-1] == j[0:n-1]:
        return i[n-1] < j[n-1]
    else:
        return lessthan(i, j, n-1)
```

Can similarly define other order relations.

Loop dependence

Statement S1 (source) depends on statement S2 (sink) if:

- There exist iteration vectors i and j such that $i < j$ or $i = j$
- There is a path from S1 to S2 in the loop
- S1 accesses memory M in iteration i
- S2 accesses memory M in iteration j
- and one of the accesses is a write

Distance Vectors

$$d(i,j)_k = j_k - i_k$$

- Where $i, j, d(i,j)$ are n-element vectors
- i_k indicates k-th element of i

Example distance vector: $(0, 1)$

Direction Vectors

$D(i, j)_k =$

- $"<"$, if $d(i, j)_k > 0$
- $"="$, if $d(i, j)_k = 0$
- $">"$, if $d(i, j)_k < 0$

Example direction vector for $(0, 1)$: $(=, <)$

Information we need to track

For every pair of memory references:

- Iteration Vectors i and j which have a dependence, or
- Unique Distance Vectors $d(i, j)$, or
- Unique Direction Vectors $D(i, j)$

Test

- Which of these indicates a loop-independent dependence?
  - $(=,=)$
  - $(=,<)$
- Of the loop-carried dependence in example above, what level is the loop-carried dependence?

Theorems

WARNING: Informal language

- Direction Vector Transform (Theorem 2.3 in AK)
  - If a transformation reorders loop iterations, and preserves the leftmost non-"$=$" component as "$<$", all dependences are preserved.
- Theorem 2.4 in AK
  - If a level-k dependence exists, and a transformation reorders loop iterations while not reordering the level-k loop
  - And does not move loops inside k outside the loop and vice versa
  - It preserves all level-k dependences.
- Iteration Reordering (Theorem 2.6 in AK)
  - Iterations of a level k loop can be reordered if there is no level k dependence.

Outline

Characterizing loop dependences

Identifying Loop Dependences

Current Loop Optimizations

Generalizing Loop Indices

```
DO I_1 = ...
   DO I_2 = ...
      ...
      DO I_N = ...
         A(f1, f2, f3, ..., fM) = ...
         ... = A(g1, g2, g3, ..., gM)
      ENDDO
   ENDDO
ENDDO
```

where A is M-dimensional array, and fX and gX are *index functions* of the form

- fX(I_1, I_2, ..., I_N)
- gX(I_1, I_2, ..., I_N)
- $1 <= X <= M$

Dependence using Iteration Vectors

Let α and β be iteration vectors:

- $\alpha = (i_1, i_2, i_3, ..., i_N)$
- $\beta = (i'_1, i'_2, i'_3, ..., i'_N)$

Then a dependence exists if:

- (vectors) $\alpha < \beta$
- $fX(\alpha) = gX(\beta)$, for $1 <= X <= M$

Example

```
DO J = 1, 10
  DO I = 1, 10
   A(I+1, J) = A(I, J) + X
  ENDDO
ENDDO
```

- $f1(J, I) = I + 1$, $f2(J, I) = J$
- $g1(J, I) = I$, $g2(J, I) = J$
- For $\alpha = (0, 0)$ (i.e. $J = 0, I = 0$) and $\beta = (0, 1)$ (i.e. $J = 0, I = 1$):
  - $f1(\alpha) = g1(\beta)$, i.e. $1 = 1$
  - $f2(\alpha) = g2(\beta)$, i.e. $0 = 0$
  - Many other values of α and β also satisfy these equations.

Dependence Testing

Do iteration vectors α and β exist such that:

- (vectors) $\alpha < \beta$
- $fX(\alpha) = gX(\beta)$, for $1 <= X <= M$

How can we find α and β if they exist?

Restrictions on Index functions

- fX and gX must be decidable
- fX and gX must be "analyzable"
  - to avoid brute force search
- fX and gX must be a linear functions of loop indices:
  - i.e. for $fX(i_1, i_2, i_3, ..., i_N)$
  - $fX = a_1 i_1 + a_2 i_2 + ... + a_n i_n + e$
  - e is optional loop invariant calculation

Dependence Testing on Restricted Index Functions

- Given that *fX* and *gX* are linear functions of loop indices
- Do iteration vectors α and β exist such that:

- (vectors) $\alpha < \beta$
- $fX(\alpha) = gX(\beta)$, for $1 <= X <= M$

How can we find α and β if they exist?

What is this problem better known as?

Dependence Testing

- Integer Linear Programming is NP-complete
- Lots of heuristics invented
  - Profitable to know if no solution exists since it implies no dependence!
  - See Chapter 3 of AK (we will not cover this in this course, take CSC 2/455)

Outline

Current Focus in Compilers

- GCC begin supporting vectorization for C around 4.9
  - `-ftree-vectorize` or `-O3`
  - Can get it to tell you why vectorization failed.
- LLVM also supports vectorization
  - See "Polly" at `http://polly.llvm.org`

More focus on optimization by loop transformation

- More emphasis on *Scheduling*
    - Which iteration of loop executes where
- Classical loop transformations
    - Loop tiling
    - Loop fusion
    - etc.
- Unifying theory and infrastructure
    - polyhedral.info