Compiling for Parallelism & Locality

Assignments
- Deadline for project 4 extended to Dec 1

Last time
- Data dependences and loops

Today
- Finish data dependence analysis for loops

Data Dependence Analysis

Dependence Testing in General

General code
\[
\text{do } i_1 = l_1, h_1 \\
\vdots \\
\text{do } i_n = l_n, h_n \\
\text{A}(f(i_1, \ldots, i_n)) = \ldots \text{A}(g(i_1, \ldots, i_n)) \\
\text{enddo} \\
\vdots \\
\text{enddo}
\]

There exists a dependence between iterations \(I=(i_1, \ldots, i_n)\) and \(J=(j_1, \ldots, j_n)\) when
- \(f(I) = g(J)\)
- \((l_1, \ldots, l_n) < I, J < (h_1, \ldots, h_n)\)

Algorithms for Solving the Dependence Problem

Heuristics
- GCD test (Banerjee76, Towle76): determines whether integer solution is possible, no bounds checking
- Banerjee test (Banerjee 79): checks real bounds
- I-Test (Kong et al. 90): integer solution in real bounds
- Lambda test (Li et al. 90): all dimensions simultaneously
- Delta test (Goff et al. 91): pattern matches for efficiency
- Power test (Wolfe et al. 92): extended GCD and Fourier Motzkin combination

Use some form of Fourier-Motzkin elimination for integers, exponential worst-case
- Parametric Integer Programming (Feautrier91)
- Omega test (Pugh92)

Dependence Testing

Consider the following code...
\[
\text{do } i = 1, 5 \\
\text{A}(3*i+2) = \text{A}(2*i+1) + 1 \\
\text{enddo}
\]

Question
- How do we determine whether one array reference depends on another across iterations of an iteration space?
Dependence Testing: Simple Case

Sample code

do i = l, h
   A(a*i+c) = ... A(a*i+c)
enddo

Dependence?
- a*i_1+c_1 = a*i_2+c_2, or
- a*i_1 - a*i_2 = c_2-c_1
- Solution exists if a divides c_2-c_1

GCD Test

Idea
- Generalize test to linear functions of iterators

Code

do i = l, h
   do j = l, h
      A(a_1*i + a_2*j + a_3) = ... A(b_1*i + b_2*j + b_3) ... 
   enddo
enddo

Again
- a_1*i_1 - b_1*i_1 + a_2*j_1 - b_2*j_1 = b_3 - a_3
- Solution exists if gcd(a_1, a_2, b_1, b_2) divides b_3 - a_3

Example

Code

do i = l, h
   A(2*i+2) = A(2*i-2)+1
enddo

Dependence?
- 2*i_1 - 2*i_2 = -2 - 2 = -4
  (yes, 2 divides -4)

Kind of dependence?
- Anti? i_2 + d = i_1  ⇒  d = -2
- Flow? i_1 + d = i_2  ⇒  d = 2

GCD Test

Code

do i = l, h
   do j = l, h
      A(4*i + 2*j + 1) = ... A(6*i + 2*j + 4) ... 
   enddo
enddo

Again
- 4*i_1 - 6*i_1 + 2*j_1 - 2*j_2 = b_3 - a_3
- Solution exists if gcd(4, 6, 2, 2) divides b_3 - a_3

Example

Code

do i = l, h
   do j = l, h
      A(4*i + 2*j) = A(6*i + 2*j + 4) ... 
   enddo
enddo

gcd(4, 6, 2, 2) = 2

Does 2 divide 4-1?
Banerjee Test

```c
for (i=L; i<=U; i++) {
    x[a_0 + a_1*i] = ...
    ...
    = x[b_0 + b_1*i]
}
```

Does \(a_0 + a_1*i = b_0 + b_1*i\) for some integer \(i\) and \(i'\)?

If so then 
\[(a_1*i - b_1*i') = (b_0 - a_0)\]

Determine upper and lower bounds on \((a_1*i - b_1*i')\)

```c
for (i=1; i<=5; i++) {
    x[i+5] = x[i];
}
```

upper bound = \(a_1*\text{max}(i) - b_1 * \text{min}(i')\) = 4
lower bound = \(a_1*\text{min}(i) - b_1*\text{max}(i')\) = -4

Distance Vectors: Legality

Definition

– A dependence vector, \(v\), is **lexicographically nonnegative** when the left-most entry in \(v\) is positive or all elements of \(v\) are zero

Yes: \((0,0,0), (0,1), (0,2,-2)\)

No: \((-1), (0,-2), (0,1,1)\)

– A dependence vector is **legal** when it is lexicographically nonnegative (assuming that indices increase as we iterate)

Why are lexicographically negative distance vectors illegal?

What are legal direction vectors?

Direction Vector

**Definition**

– A direction vector serves the same purpose as a distance vector when less precision is required or available

– Element \(i\) of a direction vector is \(<, >, =\) based on whether the source of the dependence precedes, follows or is in the same iteration as the target in loop \(i\)

**Example**

```c
do i = 1,6
    do j = 1,5
        A(i,j) = A(i-1,j-1)+1
    enddo
enddo
```

**Direction vector:** \((<,<)\)

**Distance vector:** \((1,1)\)

Loop-Carried Dependences

**Definition**

– A dependence \(D=(d_1,...,d_n)\) is **carried** at loop level \(i\) if \(d_i\) is the first nonzero element of \(D\)

**Example**

```c
do i = 1,6
    do j = 1,6
        A(i,j) = B(i-1,j) + 1
    enddo
enddo
```

**Distance vectors:** 
- \((0,1)\) for accesses to \(A\)
- \((1,0)\) for accesses to \(B\)

**Loop-carried dependences**

– The \(j\) loop carries dependence due to \(A\)
– The \(i\) loop carries dependence due to \(B\)
Parallelization

Idea

- Each iteration of a loop may be executed in parallel if it carries no dependences

Example

\[
\begin{align*}
& \text{do } i = 1, 6 \\
& \quad \text{do } j = 1, 5 \\
& \quad \quad A(i, j) = B(i-1, j-1) + 1 \\
& \quad \quad B(i, j) = A(i, j-1) \times 2 \\
& \quad \text{enddo} \\
& \text{enddo}
\end{align*}
\]

Parallelize i loop?

Parallelization

Idea

- Each iteration of a loop may be executed in parallel if it carries no dependences

Example

\[
\begin{align*}
& \text{do } i = 1, 6 \\
& \quad \text{do } j = 1, 5 \\
& \quad \quad A(i, j) = B(i-1, j-1) + 1 \\
& \quad \quad B(i, j) = A(i, j-1) \times 2 \\
& \quad \text{enddo} \\
& \text{enddo}
\end{align*}
\]

Parallelize j loop?

Example 2: Parallelization (reprise)

Why can’t this loop be parallelized?

\[
\begin{align*}
& \text{do } i = 1, 100 \\
& \quad A(i) = A(i-1) + 1 \\
& \text{enddo}
\end{align*}
\]

Why can this loop be parallelized?

\[
\begin{align*}
& \text{do } i = 1, 100 \\
& \quad A(i) = A(i) + 1 \\
& \text{enddo}
\end{align*}
\]

Distance Vector: (1)

Example 1: Loop Permutation (reprise)

Sample code

\[
\begin{align*}
& \text{do } j = 1, 6 \\
& \quad \text{do } i = 1, 5 \\
& \quad \quad A(j, i) = A(j, i) + 1 \\
& \quad \text{enddo} \\
& \text{enddo}
\end{align*}
\]

Why is this legal?

- No loop-carried dependences, so we can arbitrarily change order of iteration execution
**Concepts**

Improve performance by ...
- improving data locality
- parallelizing the computation

**Data Dependences**
- iteration space
- distance vectors and direction vectors
- loop carried

**Transformation legality**
- must respect data dependences
- scalar expansion as a technique to remove anti and output dependences

**Data Dependence Testing**
- general formulation of the problem
- GCD test

---

**Next Time**

**Lecture**
- Loop transformations for parallelism and locality