Data Dependences and Parallelization
Agenda

- Introduction
- Single Loop
- Nested Loops
- Data Dependence Analysis
Motivation

- DOALL loops: loops whose iterations can execute in parallel

```plaintext
for i = 11, 20
   a[i] = a[i] + 3
```

- New abstraction needed
  - Abstraction used in data flow analysis is inadequate
    - Information of all instances of a statement is combined
Examples

\[
\text{for } i = 11, 20 \\
\quad a[i] = a[i] + 3
\]

Parallel

\[
\text{for } i = 11, 20 \\
\quad a[i] = a[i-1] + 3
\]

Parallel?
Examples

\[
\text{for } i = 11, 20 \\
\text{ } a[i] = a[i] + 3 \quad \text{Parallel}
\]

\[
\text{for } i = 11, 20 \\
\text{ } a[i] = a[i-1] + 3 \quad \text{Not parallel}
\]

\[
\text{for } i = 11, 20 \\
\text{ } a[i] = a[i-10] + 3 \quad \text{Parallel?}
\]
Agenda

- Introduction
- *Single Loop*
- Nested Loops
- Data Dependence Analysis
Data Dependence of Scalar Variables

- True dependence
  \[ a = a = a \]

- Anti-dependence
  \[ a = = a \]

- Output dependence
  \[ a = a = a \]

- Input dependence
  \[ a = = a \]
Array Accesses in a Loop

for i = 2, 5
a[i] = a[i] + 3

Stanford University
CS243 Winter 2006
Array Anti-dependence

for i = 2, 5
a[i-2] = a[i] + 3
Array True-dependence

for $i = 2, 5$

$a[i] = a[i-2] + 3$
Dynamic Data Dependence

- Let $o$ and $o'$ be two (dynamic) operations
- Data dependence exists from $o$ to $o'$, iff
  - either $o$ or $o'$ is a write operation
  - $o$ and $o'$ may refer to the same location
  - $o$ executes before $o'$
Static Data Dependence

- Let a and a’ be two static array accesses (not necessarily distinct)
- Data dependence exists from a to a’, iff
  - either a or a’ is a write operation
  - There exists a dynamic instance of a (o) and a dynamic instance of a’ (o’) such that
    - o and o’ may refer to the same location
    - o executes before o’
Recognizing DOALL Loops

- Find data dependences in loop
- Definition: a dependence is loop-carried if it crosses an iteration boundary
- If there are no loop-carried dependences, then loop is parallelizable
Compute Dependence

for i = 2, 5
a[i-2] = a[i] + 3

- There is a dependence between a[i] and a[i-2] if
  - There exist two iterations \( i_r \) and \( i_w \) within the loop bounds such that iterations \( i_r \) and \( i_w \) read and write the same array element, respectively
  - There exist \( i_r, i_w, 2 \leq i_r, i_w \leq 5, i_r = i_w - 2 \)
There exists a dependence between $a[i-2]$ and $a[i]$ if

- There exist two iterations $i_v$ and $i_w$ within the loop bounds such that iterations $i_v$ and $i_w$ write the same array element, respectively.
- There exist $i_v$, $i_w$, $2 \leq i_v, i_w \leq 5$, $i_v - 2 = i_w - 2$.
Parallelization

for i= 2, 5
\[ a[i-2] = a[i] + 3 \]

- Is there a loop-carried dependence between a[i] and a[i-2]?
- Is there a loop-carried dependence between a[i-2] and a[i-2]?
Agenda

- Introduction
- Single Loops
- Nested Loops
- Data Dependence Analysis
Nested Loops

Which loop(s) are parallel?

\[
\begin{align*}
\text{for } i_1 &= 0, 5 \\
\text{for } i_2 &= 0, 3 \\
a[i_1,i_2] &= a[i_1-2,i_2-1] + 3
\end{align*}
\]
Iteration Space

- An abstraction for loops
  ```
  for i1 = 0, 5
  for i2 = 0, 3
  a[i1,i2] = 3
  ```

- Iteration is represented as coordinates in iteration space.
Execution Order

- Sequential execution order of iterations: Lexicographic order [0,0], [0,1], …[0,3], [1,0], [1,1], …[1,3], [2,0]…
- Let \( I = (i_1, i_2, \ldots, i_n) \). \( I \) is lexicographically less than \( I' \), \( I < I' \), iff there exists \( k \) such that \( (i_1, \ldots, i_{k-1}) = (i'_1, \ldots, i'_{k-1}) \) and \( i_k < i'_k \)
Is there a data dependence between $a[i1,i2]$ and $a[i1-2,i2-1]$?

- There exist $i1_r$, $i2_r$, $i1_w$, $i2_w$, such that
- $0 \leq i1_r, i1_w \leq 5,$
- $0 \leq i2_r, i2_w \leq 3,$
- $i1_r - 2 = i1_w$
- $i2_r - 1 = i2_w$
Loop-carried Dependence

- If there are no loop-carried dependences, then loop is parallelizable.
- Dependence carried by outer loop:
  - $i_1^r \neq i_1^w$
- Dependence carried by inner loop:
  - $i_1^r = i_1^w$
  - $i_2^r \neq i_2^w$
- This can naturally be extended to dependence carried by loop level $k$. 
Nested Loops

Which loop carries the dependence?

for i₁ = 0, 5
for i₂ = 0, 3
a[i₁,i₂] = a[i₁-2,i₂-1] + 3
Agenda

- Introduction
- Single Loop
- Nested Loops
- *Data Dependence Analysis*
Solving Data Dependence Problems

- Memory disambiguation is un-decidable at compile-time.

```plaintext
read(n)
for i = 0, 3
    a[i] = a[n] + 3
```
Domain of Data Dependence Analysis

- Only use loop bounds and array indices which are integer linear functions of variables.

\[
\text{for } i_1 = 1, n \\
\text{for } i_2 = 2 \cdot i_1, 100 \\
\text{for } a[i_1+2*i_2+3][4*i_1+2*i_2][i_1*i_1] = \ldots \\
\ldots = a[1][2*i_1+1][i_2] + 3
\]
Equations

- There is a data dependence, if
  - There exist \( i_1, i_2, i_1, i_2 \), such that
  - \( 0 \leq i_1, i_1 \leq n, 2i_1 \leq i_2 \leq 100, 2i_1 \leq i_2 \leq 100 \)
  - \( i_1 + 2i_2 + 3 = 1, 4i_1 + 2i_2 = 2i_1 + 1 \)
- Note: ignoring non-affine relations

\[
\text{for } i_1 = 1, n \\
\text{for } i_2 = 2i_1, 100 \\
a[i_1+2i_2+3][4i_1+2i_2][i_1i_1] = \ldots \\
\ldots = a[1][2i_1+1][i_2] + 3
\]
Solutions

- There is a data dependence, if
  - There exist \(i_{1r}, i_{2r}, i_{1w}, i_{2w}\), such that
  - \(0 \leq i_{1r}, i_{1w} \leq n, 2*i_{1w} \leq i_{2w} \leq 100, 2*i_{1w} \leq i_{2w} \leq 100\),
  - \(i_{1w} + 2*i_{2w} + 3 = 1, 4*i_{1w} + 2*i_{2w} - 1 = i_{1r} + 1\)

- No solution \(\rightarrow\) No data dependence

- Solution \(\rightarrow\) there may be a dependence
Form of Data Dependence Analysis

- Data dependence problems originally contains equalities and equalities
- Eliminate inequalities in the problem statement:
  - Replace $a \neq b$ with two sub-problems: $a > b$ or $a < b$
  - We get

$$\exists \text{int } i, A_1 i \leq \bar{b}_1, A_2 i = \bar{b}_2$$
Form of Data Dependence Analysis

- Eliminate equalities in the problem statement:
  - Replace $a = b$ with two sub-problems: $a \leq b$ and $b \leq a$
  - We get
    \[
    \exists \text{int } \tilde{i}, \ A\tilde{i} \leq \tilde{b}
    \]

- Integer programming is NP-complete, i.e. expensive
Techniques: Inexact Tests

- Examples: GCD test, Banerjee’s test
- 2 outcomes
  - No $\rightarrow$ no dependence
  - Don’t know $\rightarrow$ assume there is a solution $\rightarrow$ dependence
- Extra data dependence constraints
- Sacrifice parallelism for compiler efficiency
GCD Test

- Is there any dependence?

\[
\text{for } i = 1, 100 \\
a[2*i] = \ldots \\
\ldots = a[2*i+1] + 3
\]

- Solve a linear Diophantine equation
  - \(2i_w = 2i_r + 1\)
GCD

- The greatest common divisor (GCD) of integers \(a_1, a_2, \ldots, a_n\), denoted \(\gcd(a_1, a_2, \ldots, a_n)\), is the largest integer that evenly divides all these integers.

- Theorem: The linear Diophantine equation

\[
a_1x_1 + a_2x_2 + \ldots + a_nx_n = c
\]

has an integer solution \(x_1, x_2, \ldots, x_n\) iff \(\gcd(a_1, a_2, \ldots, a_n)\) divides \(c\)
Examples

- **Example 1**: $\gcd(2, -2) = 2$. No solutions
  
  \[ 2x_1 - 2x_2 = 1 \]

- **Example 2**: $\gcd(24, 36, 54) = 6$. Many solutions
  
  \[ 24x + 36y + 54z = 30 \]
Multiple Equalities

Equation 1: \( \gcd(1, -2, 1) = 1. \) Many solutions

Equation 2: \( \gcd(3, 2, 1) = 1. \) Many solutions

Is there any solution satisfying both equations?

\[
\begin{align*}
x - 2y + z &= 0 \\
3x + 2y + z &= 5
\end{align*}
\]
The Euclidean Algorithm

- Assume $a$ and $b$ are positive integers, and $a > b$.
- Let $c$ be the remainder of $a/b$.
  - If $c=0$, then $\gcd(a,b) = b$.
  - Otherwise, $\gcd(a,b) = \gcd(b,c)$.
- $\gcd(a_1, a_2, ..., a_n) = \gcd(\gcd(a_1, a_2), a_3 ..., a_n)$
Exact Analysis

- Most memory disambiguations are simple integer programs.
- Approach: Solve exactly – yes, or no solution
  - Solve exactly with Fourier-Motzkin + branch and bound
  - Omega package from University of Maryland
Incremental Analysis

- Use a series of simple tests to solve simple programs (based on properties of inequalities rather than array access patterns)
- Solve exactly with Fourier-Motzkin + branch and bound
- Memoization
  - Many identical integer programs solved for each program
  - Save the results so it need not be recomputed
State of the Art

- Multiprocessors need large outer parallel loops
- Many inter-procedural optimizations are needed
  - Interprocedural scalar optimizations
    - Dependence
    - Privatization
    - Reduction recognition
  - Interprocedural array analysis
    - Array section analysis
Summary

- DOALL loops
- Iteration Space
- Data dependence analysis