An Efficient Data Dependence Analysis for Parallelizing Compilers

Alok Jadhav - 05305041
John Chaitanya Kati-05305046

11th November, 2005
Agenda

- Background
- Basic Concepts
- Algorithm
- Examples
- Conclusion
- References
Approaches in Dependence Testing

- Based on Numerical Methods, Inaccurate
  - Multiple Dimensions simultaneously, Time Consuming
  - $\lambda$ test, Efficient and Accurate
Approaches in Dependence Testing

- Based on Numerical Methods, Inaccurate
- Multiple Dimensions simultaneously, Time Consuming
- \( \lambda \) test, Efficient and Accurate
Approaches in Dependence Testing

- Based on Numerical Methods, Inaccurate
- Multiple Dimensions simultaneously, Time Consuming
- \( \lambda \) test, Efficient and Accurate
Numerical Methods

- Based on solving Diophantine Equations and the Bounds on real functions
  - Data areas accessed by two array references are examined Dimension by Dimension
  - Inaccurate
Numerical Methods

- Based on solving Diophantine Equations and the Bounds on real functions
- Data areas accessed by two array references are examined Dimension by Dimension
- Inaccurate
Numerical Methods

- Based on solving Diophantine Equations and the Bounds on real functions
- Data areas accessed by two array references are examined Dimension by Dimension
- Inaccurate
Example

\[ N = 50 \]
\[ DO \ I = 1, N \]
\[ DO \ J = 2, N \]
\[ S1 : \quad A[2 \ast I + 3 \ast J + N, 3 \ast I + J + N - 1] = ....(r1) \]
\[ S2 : \quad .... = A[I - J + N + 1, 2 \ast I - J + N - 2] \quad (r2) \]

\textit{ENDO}
\textit{ENDDO}

- Let \( x_1 = I \) and \( x_2 = J \) for reference \( r1 \)
- Let \( x_3 = I \) and \( x_4 = J \) for reference \( r2 \)

- Diaphontine Equations are
  \[ 2x_1 + 3x_2 - x_3 + x_4 = 1 \ldots . . . . . . (1) \]
  \[ 3x_1 + x_2 - 2x_3 + x_4 = -1 \ldots . . . . . . (2) \]
  \textit{where} \( 1 \leq x_1, x_3 \leq 50, 2 \leq x_2, x_4 \leq 50 \)
Example (Cont...)

Solution by previous Numerical Methods :-

- Each Dimension is treated separately
- Any equation has no solution within loop bounds, no dependence
- If each equation has solution independently, dependence has to be assumed

- \((x_1, x_2, x_3, x_4) = (1,2,9,2)\) - solution to (1)
- \((x_1, x_2, x_3, x_4) = (1,2,4,2)\) - solution to (2)

- So we assume dependence exists
Example (Cont...)

- Actual solution \((1) \times 3 - (2) \times 2\)
- Reduced Equation \(7x_2 + x_3 + x_4 = 5\)
- No solution
- Previous Numerical Methods failed because of presence of coupled subscripts
Effect of *coupled subscripts* on determination of *dependence directions*

\[
DO \ i = 1, 100 \\
DO \ j = 1, 100 \\
S1 : \quad A[i, j] = \ldots \ldots (3) \\
S2 : \quad A[j, i] = \ldots \ldots (4) \\
ENDO \\
ENDO
\]

\[\downarrow\]
\[i_1 = j_2 \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots (5)\]
\[j_1 = i_2 \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots (6)\]
\[1 \leq i_1, i_2, j_1, j_2 \leq 100 \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots (7)\]
\[i_1 = i_2, j_1 < j_2 \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots (8)\]
Effect of *coupled subscripts* on determination of dependence directions

\[ DO \ i = 1, 100 \]
\[ DO \ j = 1, 100 \]
\[ S1: \quad A[i, j] = \ldots \quad (3) \]
\[ S2: \quad A[j, i] = \ldots \quad (4) \]

\[ ENDO \]
\[ ENDO \]

\[ \downarrow \]
\[ i_1 = j_2 \quad \ldots \quad (5) \]
\[ j_1 = i_2 \quad \ldots \quad (6) \]
\[ 1 \leq i_1, i_2, j_1, j_2 \leq 100 \quad (7) \]
\[ i_1 = i_2, j_1 < j_2 \quad \ldots \quad (8) \]
Effect of *coupled subscripts* on determination of dependence directions (Cont...)

- Previous Methods treat each dimension separately
- (5), (7), (8) is considered, solution is $i_1 = j_2 = 100$
- (6), (7), (8) is considered, solution is $j_1 = i_2 = 1$
- Actually there is no solution, if both equations are considered simultaneously
- Previous Methods failed because of presence of *coupled subscripts*
Notations used

1. Reference pair \((r1, r2)\)

2. \(r1\) is \(A(f_1(i_1, i_2, \ldots, i_{l_1}), f_2(i_1, i_2, \ldots, i_{l_1}), \ldots, f_m(i_1, i_2, \ldots, i_{l_1}))\)

3. \(r2\) is \(A(g_1(j_1, j_2, \ldots, j_{l_2}), g_2(j_1, j_2, \ldots, j_{l_2}), \ldots, g_m(j_1, j_2, \ldots, j_{l_2}))\)
Basic Concepts

\[ f_1(i_1, i_2, \ldots, i_{l_1}) = g_1(j_1, j_2, \ldots, j_{l_2}) \]
\[ f_2(i_1, i_2, \ldots, i_{l_1}) = g_2(j_1, j_2, \ldots, j_{l_2}) \]
\[ \vdots \]
\[ f_m(i_1, i_2, \ldots, i_{l_1}) = g_2(j_1, j_2, \ldots, j_{l_2}) \]

\[ a^{(1)}_1 v^{(1)} + a^{(2)}_1 v^{(2)} + \cdots + a^{(n)}_1 v^{(n)} + c_1 = 0 \]
\[ a^{(1)}_2 v^{(1)} + a^{(2)}_2 v^{(2)} + \cdots + a^{(n)}_2 v^{(n)} + c_2 = 0 \]
\[ \vdots \]
\[ a^{(1)}_m v^{(1)} + a^{(2)}_m v^{(2)} + \cdots + a^{(n)}_m v^{(n)} + c_m = 0 \]
Basic Concepts

\[
f_1(i_1, i_2, \ldots, i_{l_1}) = g_1(j_1, j_2, \ldots, j_{l_2})
\]
\[
f_2(i_1, i_2, \ldots, i_{l_1}) = g_2(j_1, j_2, \ldots, j_{l_2})
\]
\[\vdots\]
\[
f_m(i_1, i_2, \ldots, i_{l_1}) = g_2(j_1, j_2, \ldots, j_{l_2})
\]

\[
a^{(1)}_1 v^{(1)} + a^{(2)}_1 v^{(2)} + \cdots + a^{(n)}_1 v^{(n)} + c_1 = 0
\]
\[
a^{(1)}_2 v^{(1)} + a^{(2)}_2 v^{(2)} + \cdots + a^{(n)}_2 v^{(n)} + c_2 = 0
\]
\[\vdots\]
\[
a^{(1)}_m v^{(1)} + a^{(2)}_m v^{(2)} + \cdots + a^{(n)}_m v^{(n)} + c_m = 0
\]
Geometrical illustrations

Figure: Geometrical illustration

Notations

Basic Concepts
Case for Two-dimensional array reference

Alok Jadhav - 05305041 John Chaitanya Kati-05305046
An Efficient Data Dependence Analysis for Parallelizing Compilers
Our equations are: \( f_1 = 0 \) and \( f_2 = 0 \)

where \( f_i = a_i^{(1)} v^{(1)} + a_i^{(2)} v^{(2)} + \cdots + a_i^{(n)} v^{(n)} + c_i \)

Linear combination is: \( f_{\lambda_1,\lambda_2} = \lambda_1 f_1 + \lambda_2 f_2 \)

in expanded form

\[
f_{\lambda_1,\lambda_2} = (\lambda_1 a_1^{(1)} + \lambda_2 a_2^{(1)}) v^{(1)} + (\lambda_1 a_1^{(2)} + \lambda_2 a_2^{(2)}) v^{(2)} + \cdots + (\lambda_1 a_1^{(n)} + \lambda_2 a_2^{(n)}) v^{(n)}
\]
Our equations are: \( f_1 = 0 \) and \( f_2 = 0 \)
where \( f_i = a_i^{(1)}v^{(1)} + a_i^{(2)}v^{(2)} + \cdots + a_i^{(n)}v^{(n)} + c_i \)

Linear combination is: \( f_{\lambda_1, \lambda_2} = \lambda_1 f_1 + \lambda_2 f_2 \)
in expanded form
\[
f_{\lambda_1, \lambda_2} = (\lambda_1 a_1^{(1)} + \lambda_2 a_2^{(1)})v^{(1)} + (\lambda_1 a_1^{(2)} + \lambda_2 a_2^{(2)})v^{(2)} + \cdots + (\lambda_1 a_1^{(n)} + \lambda_2 a_2^{(n)})v^{(n)}
\]
Case for Two-dimensional array reference

Our equations are: \( f_1 = 0 \) and \( f_2 = 0 \)
where \( f_i = a_i^{(1)} v^{(1)} + a_i^{(2)} v^{(2)} + \cdots + a_i^{(n)} v^{(n)} + c_i \)

Linear combination is: \( f_{\lambda_1,\lambda_2} = \lambda_1 f_1 + \lambda_2 f_2 \)
in expanded form
\[
f_{\lambda_1,\lambda_2} = (\lambda_1 a_1^{(1)} + \lambda_2 a_2^{(1)}) v^{(1)} + (\lambda_1 a_1^{(2)} + \lambda_2 a_2^{(2)}) v^{(2)} + \cdots + (\lambda_1 a_1^{(n)} + \lambda_2 a_2^{(n)}) v^{(n)}
\]
Definition

\[ \psi^{(i)} = \lambda_1 a_1^{(i)} + \lambda_2 a_2^{(i)} \]

\[ \phi^{(i,j)} = \psi^{(i)} + \psi^{(j)} \]

where \( \nu^{(i)} \) and \( \nu^{(j)} \) are related by a dependance direction
Canonical Solution

\[ a\lambda_1 + b\lambda_2 = 0 \]

\[ (\lambda_1, \lambda_2) = (1, 0), \quad \text{if} \quad a = 0 \]

\[ (\lambda_1, \lambda_2) = (0, 1), \quad \text{if} \quad b = 0 \]

\[ (\lambda_1, \lambda_2) = (b, -a), \quad \text{if} \quad a, b \neq 0 \text{ and } b > 0 \]

\[ (\lambda_1, \lambda_2) = (-b, a), \quad \text{if} \quad a, b \neq 0 \text{ and } b < 0 \]
A Set: Set of all canonical solutions to the $\psi$ equations and $\phi$ equations.

Every canonical solution determines a $\lambda$ plane.

Theorem: $S$ intersects $V$ if and only if every $\lambda$ plane intersects $V$. 

Algorithm

- Determine the $\psi$ equations and $\phi$ equations
- Determine the $\Lambda$ set
- Each element of $\Lambda$ set determines a $\lambda$ plane
- Each $\lambda$ plane is tested to see if intersects $V$ by checking its max and min values
- If any one of the $\lambda$ planes does not intersect $V$ we stop
\[ f_{\lambda_1, \lambda_2} = (2\lambda_1 + 3\lambda_2)x_1 + (3\lambda_1 + \lambda_2)x_2 \\
+(-\lambda_1 - 2\lambda_2)x_3 + (\lambda_1 + \lambda_2)x_4 = 0 \]

\( \psi \) Equations:
\[ 2\lambda_1 + 3\lambda_2 = 0 \]
\[ 3\lambda_1 + \lambda_2 = 0 \]
\[ -\lambda_1 - 2\lambda_2 = 0 \]
\[ \lambda_1 + \lambda_2 = 0 \]
\( \Lambda = (3, -2), (1, -3), (2, -1), (1, -1) \)

Consider the loop bounds:
\[
1 \leq x_1, x_3 \leq 50, \quad 2 \leq x_2, x_4 \leq 50
\]

Here, \((3, -2)\) shows the absence of data dependence and hence we stop and do not consider the next \(\lambda\) planes
\[ i_1 = j_2 \]
\[ j_1 = i_2 \]
\[ 1 \leq i_1, i_2, j_1, j_2 \leq 100 \]
\[ i_1 = i_2, \; j_1 < j_2 \]
\[ f_{\lambda_1, \lambda_2} = (\lambda_1 + 0\lambda_2)i_1 + (0\lambda_1 + \lambda_2)j_1 + (-\lambda_1 + 0\lambda_2)j_2 + (0\lambda_1 - \lambda_2)i_2 = 0 \]

\[ \psi \quad \text{Equations:} \]
\[ \lambda_1 + 0\lambda_2 = 0 \]
\[ 0\lambda_1 + \lambda_2 = 0 \]
\[ -\lambda_1 + 0\lambda_2 = 0 \]
\[ 0\lambda_1 - \lambda_2 = 0 \]
\[ \lambda = \textit{test} \]

\[
\begin{align*}
\phi \quad \text{Equations:} \\
\lambda_1 - \lambda_2 &= 0 \\
-\lambda_1 + \lambda_2 &= 0
\end{align*}
\]

\[ \Lambda = (1, 0), (0, 1), (1, 1) \]

Consider the loop bounds:

\[ 1 \leq x_1, x_3 \leq 50, \quad 2 \leq x_2, x_4 \leq 50 \]

Here, \((1, 1)\) shows the absence of data dependence and hence we stop
Discussed λ test and examined that it is better than previous Numerical methods
[1] An Efficient Data Dependence Analysis for Parallelizing Compilers, Zhiyuan Li and Pen-Chung Yew and Chuan-Qi Zhu 
IEEE Transactions on Parallel and Distributed Systems, 1990