Lecture 5

Data Distributions
Goal:

- distribute arrays across local memories of parallel machine so that data elements can be accessed in parallel
- Standard distributions for dense arrays: (HPF, Scalapack)
  - block
  - cyclic
  - block cyclic(b)
- Block cyclic distribution subsumes other two
DISTRIBUTE A(BLOCK)

DISTRIBUTE A(BLOCK(4))

A(i) is mapped to processor \[\frac{i}{b}\] if distribution is BLOCK(b)
Cyclic/Block Cyclic:

\[
\begin{array}{cccc}
P0 & P1 & P2 & P3 \\
\bullet & \bullet & \bullet & \bullet \\
0 & 9
\end{array}
\]

\[A(i) \text{ is mapped to processor } P \left\lfloor \frac{i}{b} \right\rfloor \mod P\]
if distribution is CYCLIC(b)
Common use of cyclic distribution:

Matrix factorization codes

- BLOCK distribution: small number of processors end up with all the work after a while

- CYCLIC distribution: better load balance

- BLOCK-CYCLIC: lower communication costs than CYCLIC
Distributions for 2-D Arrays:

Each dimension can be distributed by
- block
- cyclic
- *: dimension not distributed

A (4,8)
Distributing both dimensions:

- # of array distribution dimensions
  = # of dimensions of processor grid

- 2-D processor grid

A (4,8)

DISTRIBUTE A (BLOCK, BLOCK)  
DISTRIBUTE A (BLOCK, CYCLIC)  
DISTRIBUTE A (CYCLIC, CYCLIC)
Let us re-examine matrix-vector product:

We looked at three different versions of matrix-vector product:
- self-scheduled master/slave version from MPI doc
- 1-D alignment: each processor gets a contiguous set of rows
- 2-D alignment: each processor gets a block

At present, compiler technology is not adequate to determine which version is best for a given application.

HPF position: you tell us data distribution, we generate code.

```plaintext
DISTRIBUTE A(BLOCK,*), X(BLOCK), Y(BLOCK)

DO 10 I = 1, N
   DO 10 J = 1, N
   10 Y(I) = Y(I) + A(I,J)*X(J)
```

Detail: HPF requires both alignment and distribution directives

HPF does not support dynamic data distributions such as those for m/s version
Lecture 5 (contd)

Integer Linear Programming Problems in Restructuring Compilers
Two problems:

Given a system of linear inequalities $A x \leq b$

where $A$ is a $m \times n$ matrix of integers,
$b$ is an $m$ vector of integers,
$x$ is an $n$ vector of unknowns,

(i) Are there integer solutions?
(ii) Enumerate all integer solutions.

These problems are at the heart of dependence analysis, loop transformations and code generation in compiling programs with dense matrix computations.
Is this a parallel loop or are there dependences between iterations?

- Dependences arise if we write a location in some iteration \( w \) and read that location in a later iteration \( r \).

- Mathematically, we have the following constraints:

\[
\begin{align*}
1 & \leq w < r \leq 100 \\
2w + 1 & = 2r \\
w, r & \text{ are integers}
\end{align*}
\]

\[
\begin{bmatrix}
-1 & 0 \\
1 & -1 \\
0 & 1 \\
2 & -2 \\
-2 & 2
\end{bmatrix}
\begin{bmatrix}
w \\
r
\end{bmatrix}
\leq
\begin{bmatrix}
-1 \\
-1 \\
100 \\
-1 \\
1
\end{bmatrix}
\]

\( w, r \) are integers

**Constraints**

**Canonical Form**

Are there integer solutions that satisfy the constraints?

No => we have a DO-ALL loop.
Loop transformations need enumeration of all integer points in region

\[
\begin{array}{c}
\text{DO 10 } I = 1, N \\
\text{DO 10 } J = 1, M \\
10 \ Y(I) = Y(I) + A(I,J) \ast X(J) \\
\text{DO 10 } K = 1, M \\
\text{DO 10 } L = 1, N \\
10 \ Y(L) = Y(L) + A(L,K) \ast X(K)
\end{array}
\]

Loop interchange

Bounds on \( K \) & \( L \)?

\[
\begin{pmatrix}
-1 & 0 \\
1 & 0 \\
0 & -1 \\
0 & 1
\end{pmatrix}
\begin{pmatrix}
I \\
J
\end{pmatrix}
\leq
\begin{pmatrix}
-1 \\
N \\
-1 \\
M
\end{pmatrix}
\]

\[
\begin{pmatrix}
-1 & 0 \\
1 & 0 \\
0 & -1 \\
0 & 1
\end{pmatrix}
\begin{pmatrix}
0 \\
1 \\
1 \\
0
\end{pmatrix}
\begin{pmatrix}
K \\
L
\end{pmatrix}
\leq
\begin{pmatrix}
-1 \\
N \\
-1 \\
M
\end{pmatrix}
\]
Presentation sequence:

- one equation, several variables
  \[2x + 3y = 5\]
- several equations, several variables
  \[2x + 3y + 5z = 5\]
  \[3x + 4y = 3\]
- equations & inequalities
  \[2x + 3y = 5\]
  \[x > 1\]
  \[y < 3\]
Presentation sequence:

- one equation, several variables
  \[2x + 3y = 5\]

- several equations, several variables
  \[2x + 3y + 5z = 5\]
  \[3x + 4y = 3\]

- equations & inequalities
  \[2x + 3y = 5\]
  \[x > 1\]
  \[y < 3\]

Diophantine Equations:
Variations of
Gaussian Elimination

Fourier-Motzin
Elimination
One equation, many variables:

Thm: The linear Diophantine equation \( a_1 x_1 + a_2 x_2 + \ldots + a_n x_n = c \) has integer solutions iff \( \gcd(a_1,a_2,\ldots,a_n) \) divides \( c \).

Examples:

1. \( 2x = 3 \)  No solutions
2. \( 2x = 6 \)  One solution: \( x = 3 \)
3. \( 2x + y = 3 \)
   \[ \gcd(2,1) = 1 \text{ which divides } 3. \]
   Solutions: \( x = t, \quad y = (3 - 2t) \)
4. \( 2x + 3y = 3 \)
   \[ \gcd(2,3) = 1 \text{ which divides } 3. \]
   Let \( z = x + \lfloor 3/2 \rfloor \)
   \[ y = x + y \]
   Rewrite equation as \( 2z + y = 3 \)
   Solutions: \( z = t \quad \Rightarrow \quad x = (3t - 3) \)
   \[ y = (3 - 2t) \quad y = (3 - 2t) \]

Intuition: Think of underdetermined systems of eqns over reals.
Caution: Integer constraint \( \Rightarrow \) Diophantine system may have no solns
Thm: The linear Diophantine equation \( a_1 x_1 + a_2 x_2 + \ldots + a_n x_n = c \) has integer solutions iff \( \gcd(a_1, a_2, \ldots, a_n) \) divides \( c \).

Proof: WLOG, assume that all coefficients \( a_1, a_2, \ldots, a_n \) are positive.

We prove only the IF case by induction, the proof in the other direction is trivial.

The induction is on the integer pair \([\# \text{ of variables}, \text{smallest coefficient}]\), where

\[[a, b] < [c, d] \text{ if } (a < c) \text{ and } (b \leq d) \text{ or } ((a \leq c) \text{ and } (b < d)) .\]

Base case:

If \((\# \text{ of variables} = 1)\), then equation is \( a_1 x_1 = c \) which has integer solutions if \( a_1 \) divides \( c \).

If \((\text{smallest coefficient} = 1)\), then \( \gcd(a_1, a_2, \ldots, a_n) = 1 \) which divides \( c \).

Wlog, assume that \( a_1 = 1 \), and observe that the equation has solutions of the form \((c - a_2 t_2 - a_3 t_3 - \ldots - a_n t_n, t_2, t_3, \ldots t_n)\).

Inductive case:

Suppose smallest coefficient is \( a_1 \), and let

\[ t = x_1 + \floor{a_2/a_1} x_2 + \ldots + \floor{a_n/a_1} x_n \]

In terms of this variable, the equation can be rewritten as

\[ (a_1) t + (a_2 \mod a_1) x_2 + \ldots + (a_n \mod a_1) x_n = c \quad (1) \]

where we assume that all terms with zero coefficient have been deleted.

Observe that \((1)\) has integer solutions iff original equation does too.

Now \( \gcd(a, b) = \gcd(a \mod b, b) \Rightarrow \gcd(a_1, a_2, \ldots, a_n) = \gcd(a_1, (a_2 \mod a_1), \ldots, (a_n \mod a_1)) \Rightarrow \gcd(a_1, (a_2 \mod a_1), \ldots, (a_n \mod a_1)) \) divides \( c \).

Furthermore, since \((a_i \mod a_1)\) is less than or equal to \(a_1\), it is easy to see that the quantity \([\# \text{ of variables}, \text{smallest coefficient}]\) has decreased in going from the original equation to Equation \((1)\). □
Summary:

Eqn: \[ a_1 x_1 + a_2 x_2 + \ldots + a_n x_n = c \]

- Does this have integer solutions?
  \[ \text{Does } \text{gcd}(a_1,a_2,\ldots,a_n) \text{ divide } c \]?

- Enumerating all solutions:

  Theorem shows how to find all solutions
  Actual implementation: special case of systems of equations
  which we discuss next
Systems of Diophantine Equations:

Key idea: use integer Gaussian elimination

Example:

\[
\begin{align*}
2x + 3y + 4z &= 5 \\
x - y + 2z &= 5
\end{align*}
\]

\[
\begin{bmatrix}
2 & 3 & 4 \\
1 & -1 & 2
\end{bmatrix}
\begin{bmatrix}
x \\
y \\
z
\end{bmatrix}
= 
\begin{bmatrix}
5 \\
5
\end{bmatrix}
\]

It is not easy to determine if this Diophantine system has solutions.

Easy special case: lower triangular matrix

\[
\begin{bmatrix}
1 & 0 & 0 \\
-2 & 5 & 0
\end{bmatrix}
\begin{bmatrix}
x \\
y \\
z
\end{bmatrix}
= 
\begin{bmatrix}
5 \\
5
\end{bmatrix}
\Rightarrow
\begin{align*}
x &= 5 \\
y &= 3 \\
z &= \text{arbitrary integer}
\end{align*}
\]

Question: Can we convert general integer matrix into equivalent lower triangular system?
**Integer Gaussian Elimination:**

- Row/column operations to get matrix into triangular form
- For us, column operations are important because we usually have more unknowns than equations.

**Overall strategy:** Given $A \mathbf{x} = \mathbf{b}$

Find matrices $U_1, U_2, \ldots, U_k$ such that

$$A * U_1 * U_2 * \ldots * U_k \text{ is lower triangular (say) } L$$

Solve $L \mathbf{x}' = \mathbf{b}$ (easy)

Compute $\mathbf{x} = (U_1 * U_2 * \ldots * U_k) x'$

**Proof:**

$$(A * U_1 * U_2 * \ldots * U_k) x' = \mathbf{b}$$

$$\Rightarrow A (U_1 * U_2 * \ldots U_k x') = \mathbf{b} \Rightarrow \mathbf{x} = (U_1 * U_2 * \ldots U_k) x'$$
Caution: Not all column operations preserve integer solutions.

\[
\begin{bmatrix}
2 & 3 \\
6 & 7
\end{bmatrix}
\begin{bmatrix}
x \\
y
\end{bmatrix}
= 
\begin{bmatrix}
5 \\
1
\end{bmatrix}
\]

Solution: \( x = -8, \ y = 7 \)

\[
\begin{bmatrix}
1 & -3 \\
0 & 2
\end{bmatrix}
\]

\[
\begin{bmatrix}
2 & 0 \\
6 & -4
\end{bmatrix}
\begin{bmatrix}
x' \\
y'
\end{bmatrix}
= 
\begin{bmatrix}
5 \\
1
\end{bmatrix}
\]

which has no integer solutions!

Intuition: With some column operations, recovering solution of original system requires solving lower triangular system using rationals.

Question: Can we stay purely in the integer domain?

One solution: Use only unimodular column operations
Unimodular Column Operations:

(a) Interchange two columns

\[
\begin{bmatrix}
2 & 3 \\
6 & 7 \\
\end{bmatrix}
\rightarrow
\begin{bmatrix}
3 & 2 \\
7 & 6 \\
\end{bmatrix}
\]

Check

Let \( x, y \) satisfy first eqn.
Let \( x', y' \) satisfy second eqn.
\[ x' = y, \quad y' = x \]

(b) Negate a column

\[
\begin{bmatrix}
2 & 3 \\
6 & 7 \\
\end{bmatrix}
\rightarrow
\begin{bmatrix}
2 & -3 \\
6 & -7 \\
\end{bmatrix}
\]

Check

\[ x' = x, \quad y' = -y \]

(c) Add an integer multiple of one column to another

\[
\begin{bmatrix}
2 & 3 \\
6 & 7 \\
\end{bmatrix}
\rightarrow
\begin{bmatrix}
2 & 1 \\
6 & 1 \\
\end{bmatrix}
\]

Check

\[ x = x' + n y', \quad y = y' \]

n = -1
Facts:

1. The three unimodular column operations
   - interchanging two columns
   - negating a column
   - adding an integer multiple of one column to another
   on the matrix $A$ of the system $A \mathbf{x} = \mathbf{b}$
   preserve integer solutions, as do sequences of these operations.

2. Unimodular column operations can be used to reduce
   a matrix $A$ into lower triangular form.

3. A unimodular matrix has integer entries and a determinant
   of $+1$ or $-1$.

4. The product of two unimodular matrices is also unimodular.
**Algorithm:** Given a system of Diophantine equations $Ax = b$

1. Use unimodular column operations to reduce matrix $A$ to lower triangular form $L$.
2. If $Lx' = b$ has integer solutions, so does the original system.
3. If explicit form of solutions is desired, let $U$ be the product of unimodular matrices corresponding to the column operations. Then, $x = Ux'$ where $x'$ is the solution of the system $Lx' = b$.

Detail: Instead of lower triangular matrix, it is better to compute 'column echelon form' of matrix.

**Column echelon form:** Let $r_j$ be the row containing the first non-zero in column $j$.

(i) $r_{(j+1)} > r_j$ if column $j$ is not entirely zero.
(ii) column $(j+1)$ is zero if column $j$ is.

\[
\begin{bmatrix}
x & 0 & 0 \\
x & 0 & 0 \\
x & x & x \\
\end{bmatrix}
\]
is lower triangular but not column echelon.