Why tile?

High communication startup costs in DMCs \(\implies\) frequent communication is expensive.

Cluster iterations of a nested loop in order to:

- exploit locality of reference
- incur smaller execution overhead from high message startup costs: send fewer but larger messages

Issues:

- Cluster determination: legal tiles derived from a linear programming formulation.
- Communication minimal tiling
- Code generation
- Non-unimodular loop transformations
- Allocation and scheduling of tiles
- Grain size optimization of tiles
Tiling research

- Irigoin and Triolet (1988)
- Ancourt and Irigoin (1991)
- Ramanujam and Sadayappan (1990, 1991)
- Wolf and Lam (1990)
- Schreiber and Dongarra (1990)
- King and Ni (1990)
- Wolf and Lam (1992)
- Boulet et al. (1994)
Iteration Space Blocking

- A tile in an \( n \)-dimensional iteration space is an \( n \)-dimensional subset of the iteration space.
- A tile is defined by a set of boundaries regularly spaced apart.
- Each tile boundary is an \( (n - 1) \)-dimensional plane.

\[
\begin{array}{l}
\text{for } i = 2 \text{ to } N \text{ do} \\
\quad \text{for } j = 2 \text{ to } N \text{ do} \\
\quad \quad A(i,j) = A(i-1,j) + A(i,j-1) \\
\quad \text{endfor} \\
\text{endfor}
\end{array}
\]
Convex Partitions

- A, B, C and D are four tasks

Convex partitions $\implies$ no deadlock in atomic execution

Convexity condition: Sarkar (1987)
Tiling

Cluster iterations into atomic execution units: tiles

For atomic execution, i.e., once started, a tile must run to completion without any synchronization or communication interspersed with execution of iterations in a tile ⇒ no dependence cycles among tiles

all dependences that cross a tile boundary must all do so in the same direction (from one tile to the other): Sufficient condition to avoid cycles

\[ \vec{H}_i \cdot \vec{d}_j \geq 0 \quad \forall \vec{d}_j \]

or \[ \vec{H}_i \cdot \vec{d}_j \leq 0 \quad \forall \vec{d}_j \]

This must be true for all boundaries! Condition from Irigoin & Triolet (1988)
Deadlock

for i = 2 to N do
    for j = 2 to N do
        A(i, j) = A(i-1, j+1) + A(i-1, j) + A(i, j-1)
    endfor
endfor

\[ D = \begin{bmatrix} 1 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix} \]

Iteration Space
Deadlock - continued

Tile Space with $(1, 0)$ and $(0, 1)$

Cycle(s) among tiles violates atomicity requirement

Tile Space with $(1, -1)$ and $(0, 1)$
You can make the tile small enough

Dependence vectors cross in opposite direction but tile size is fixed, in order to avoid deadlock

Focus on free-size tiles (FST's)
Legal Tiles

- Tiles in $n$-dimensions require $n$ tile boundaries
- Each tile boundary is defined by a vector normal to it. Let $\vec{H}_i$ be the vector normal to the $i$th tile boundary
- Condition on tiling implies
  \[ \vec{H}_i \cdot \vec{d}_j \geq 0 \quad i = 1, \ldots, n \quad j = 1, \ldots, r \]
- Let $H$ = matrix whose rows are $\vec{H}_i$
  Let $D$ = dependence matrix whose columns distance (dependence) vectors
  Hence, $D^+ = HD$ has all entries non-negative
- For $\{\vec{H}_i\}$ to define $n$-dimensional tiles, the matrix $H$ must be non-singular, i.e., the rows of $H$ must be linearly independent $\implies H^{-1}$ exists
Tile Boundaries and Extreme Vectors

- $D_{ij}^+ = \vec{H}_i \cdot \vec{d}_j$
- $D = H^{-1} D^+$
- Since $D_{ij}^+ \geq 0$, every dependence vector (column in $D$) can be expressed as non-negative linear combinations of the columns (vectors) of the matrix $H^{-1} = E$.
- Such a collection of vectors $\{\vec{E}_i\}$ is called a set of extreme vectors.

\[
\begin{bmatrix}
\vec{d}_1 & \ldots & \vec{d}_j & \ldots & \vec{d}_r
\end{bmatrix}
= \\
\begin{bmatrix}
\vec{e}_1 & \ldots & \vec{e}_j & \ldots & \vec{e}_n
\end{bmatrix}
\begin{bmatrix}
\vec{d}_1^+ & \ldots & \vec{d}_j^+ & \ldots & \vec{d}_r^+
\end{bmatrix}
\]

Legal tiling hyperplanes $\{\vec{H}_i\}$ can be found if and only if a set of spanning extreme vectors can be found for the $\{\vec{d}_j\}$.
Extreme vectors

- The columns of $H^{-1}$ generate a cone that contains the dependence cone.
- Finding the canonical (fully permutable) form of a nest is equivalent to finding its dependence cone.
- A set of vectors $\{H^{-1}\}$ or $\{E\}$ of the following form can always be found for a given $D$ (where $e_i \geq 0$)

$$
E = \begin{bmatrix}
1 & 0 & 0 & \cdots & 0 & 0 \\
-e_1 & 1 & 0 & \cdots & 0 & 0 \\
0 & -e_2 & 1 & \cdots & 0 & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
0 & 0 & \cdots & \cdots & 1 & 0 \\
0 & 0 & \cdots & \cdots & -e_{n-1} & 1 \\
\end{bmatrix}
$$

- In this case, $E^{-1} = H$ is lower triangular with unit diagonal:

$$
E_{i,j}^{-1} = \begin{cases} 
0 & \text{if } j > i \\
1 & \text{if } j = i \\
\prod_{k=j}^{i-1} e_k & \text{if } j < i 
\end{cases}
$$
**Extreme Vectors in 2-D**

A *subset* of the dependence vectors are the extreme vectors in 2-D iteration spaces *i.e.*, from among $\vec{d}_i$, there exist 2 vectors say $\vec{d}_1$ and $\vec{d}_2$ such that

$$\vec{d}_i = a_i \vec{d}_1 + b_i \vec{d}_2 \quad i = 1, \ldots, r \quad a_i \geq 0 \quad b_i \geq 0$$

$\vec{d}_1$ and $\vec{d}_2$ are the extreme vectors. Not true for $n$-D iteration spaces ($n > 2$).

Distance vectors in 2-D are $(0, \oplus)$ or $(+, \ast)$:
Extreme Vectors in 2-D: continued

- Distance vectors belong to the I and IV quadrant except for \((i = 0, j < 0)\)
- Extreme vectors are the ones that together have the largest span
- Find \(r_i = \frac{d_{i2}}{d_{i1}}\)
- Vectors with the smallest and largest \(r_i\) are the extreme vectors
- \(O(r)\) algorithm \(r = \text{no. of distance vectors}\)

Example:

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

\((0,1)\) \(\rightarrow\) (1,-2) \(\rightarrow\) (1,-1) \(\rightarrow\) (1,0)
Finding Legal Tiles

- Any lower triangular integer matrix $H$ with unit diagonal is invertible. Its inverse is also a lower triangular integer matrix with a unit diagonal.
- In general, the determinant of $H \neq 0$ for invertibility. The determinant of an arbitrary matrix $H$ is not a linear function of its entries, $h_{ij}$
- Focus on lower triangular, unit diagonal $H$.
- Based on results so far, finding legal tiling planes can be formulated as:

\[
\sum_{l=1}^{n} H_{i,l} D_{l,j} \geq 0
\]

subject to the constraints

\[
\begin{align*}
H_{i,i} & = 1 \\
H_{i,l} & = 0 \text{ if } l > i
\end{align*}
\]
Communication

Constraints to minimize communication can be incorporated into the LP formulation \( \Rightarrow \)

A non-zero entry in \( D^+ \) say \( D^+_{ij} = \vec{H}_i \cdot \vec{d}_j \) implies that communication is incurred due to the \( j \)th dependence vector poking the \( i \)th tile boundary

- \( D^+_{ij} = 0 \implies \vec{d}_j \) lies on the \( i \)th tile boundary
- \( D^+_{ij} > 0 \implies \vec{d}_j \) pokes the \( i \)th tile boundary
Tiling for Minimal Communication

- Assume tiles of size \((L \times L \times \ldots \times L)\) in an \(n\)-dimensional iteration space
- For a lower triangular \(H\) with unit diagonal (unimodular \(H\))

\[
\text{Communication volume } = L^{n-1} \sum_{i=1}^{n} \sum_{j=1}^{r} \sum_{l=1}^{n} H_{i,l}D_{l,j}
\]

- Hence, finding legal tiling planes that minimizing communication can be formulated as the linear programming problem:

Find \(H\) such that

\[
\sum_{i=1}^{n} \sum_{j=1}^{r} \sum_{l=1}^{n} H_{i,l}D_{l,j}
\]

is minimized subject to

\[
\begin{align*}
H_{i,l} &= 0 \text{ if } l > i \\
H_{i,i} &= 1 \\
\sum_{l=1}^{n} H_{i,l}d_{l,j} &\geq 0
\end{align*}
\]
Example

Consider

\[ D = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & -1 \end{bmatrix} \]

Note: Any vector in this case can not be expressed as a non-negative linear combination of the other three.

The problem is to find \( H \)

\[ H = \begin{bmatrix} 1 & 0 & 0 \\ a & 1 & 0 \\ b & c & 1 \end{bmatrix} \]

such that

\[ D^+ = HD = \begin{bmatrix} 1 & 0 & 0 & 1 \\ a & 1 & 0 & a + 1 \\ b & c & 1 & b + c - 1 \end{bmatrix} \]

has nonnegative entries whose sum is minimized.
Example - continued

Thus, the problem is:
Minimize $2a + 2b + 2c + 4$ subject to:

\[
\begin{align*}
a & \geq 0 \\
a + 1 & \geq 0 \\
b & \geq 0 \\
c & \geq 0 \\
b + c - 1 & \geq 0
\end{align*}
\]

There are two solutions; the first is $a = 0, b = 1, c = 0$ for which the transformation $H$ is

\[
H = \begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
1 & 0 & 1
\end{bmatrix}.
\]

The extreme vectors are columns of $H^{-1}$ given by:

\[
H^{-1} = \begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
-1 & 0 & 1
\end{bmatrix}.
\]
Example - continued

The second solution is $a = 0, b = 0, c = 1$ for which the transformation $H$ is

$$H = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}$$

for which the extreme vectors are columns of $H^{-1}$ given by:

$$H^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -1 & 0 \end{bmatrix}$$

In either case, we note that two of the three columns of $H^{-1}$ are dependence vectors.
Code Generation

- Apply the transformation $H$ to the loop nest to render it tileable
- Transformation $H$ (as defined previously) can be implemented as a sequence of loop skewing transformations:

```c
/* maximum loop nest = n */

for $i = n$ to 2 by $-1$ do
    for $j = 1$ to $i - 1$ by 1 do
        if $(h_{ij} \neq 0)$ then
            skew loop $i$ with respect to loop $j$
            by a factor $h_{ij}$
        endif
    endfor
endfor
```
**Code generation – Example**

Example: loop nest = \((i_1, i_2, i_3)\)

\[
H = \begin{bmatrix}
1 & 0 & 0 \\
3 & 1 & 0 \\
8 & 3 & 1
\end{bmatrix}
\]

Let the optimal transformation \(H\) be:

\[
H = \begin{bmatrix}
1 & 0 & 0 \\
3 & 1 & 0 \\
0 & 0 & 1
\end{bmatrix} \begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 3 & 1
\end{bmatrix} \begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
8 & 0 & 1
\end{bmatrix}
\]

The three steps are (in the following order):

1. skew \(i_3\) with respect to \(i_1\) by 8
2. skew \(i_3\) with respect to \(i_2\) by 3
3. skew \(i_2\) with respect to \(i_1\) by 3
Need for non-unimodular transformations

Example:

\[ D = \begin{bmatrix} 0 & 2 \\ 1 & -1 \end{bmatrix}. \]

We need to find a non-singular matrix \( H \)

\[ H = \begin{bmatrix} 1 & h_1 \\ h_2 & 1 \end{bmatrix} \] such that \( HD = \begin{bmatrix} h_1 & 2 - h_1 \\ 1 & 2h_2 - 1 \end{bmatrix} \]

has nonnegative entries and their sum is minimum.

- LP extrema occur at corners of feasible regions ⇒ examine the invertibility of \( H \), i.e., \( 1 - h_1h_2 \neq 0 \) at each corner separately
- Corners: \( h_1 = 0, h_2 = 1/2 \) and \( h_1 = 2, h_2 = 1/2 \): \( h_1 = 2, h_2 = 1/2 \) cannot be used for valid \( H \).
- The solution is \( h_1 = 0, h_2 = 1/2 \) corresponds to a non-unimodular transformation.
- Code generation for non-unimodular matrix is an important but difficult problem which has been solved recently.
Communication with non-unimodular $H$

$H = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$

$H = \begin{bmatrix} 1 & 0 \\ 1 & 2 \end{bmatrix}$

$D = \begin{bmatrix} 2 & 0 \\ -1 & 1 \end{bmatrix}$
Scheduling tiles

- Dependences among tiles captured in the tile space graph (TSG)
- For large enough tiles (FST’s), dependences only between neighboring tiles
- Tile space dependence vectors \( \{ \mathbf{d}_j^t \} \) are \( n \)-tuples; each tuple entry \( \in \{0, 1\} \)
- In \( n \)-dimensional iteration spaces, each tile can depend on at most \( 2^n - 1 \) other tiles.
- Let \( \mathbf{T}^* = (t_1, \ldots, t_n) \) represent the scheduling hyperplane in TSG \( \Rightarrow \mathbf{T}^* \cdot \mathbf{d}_j^t > 0 \quad \forall \ j. \)
- Hence, optimal scheduling hyperplane in the TSG is the wavefront (realized through skewing in tile space)

\[
\mathbf{T}^* = (1, 1, \ldots, 1)
\]
Allocation of tiles to processors

- Most tile space dependences include the orthogonal unit vectors

  \[(0, \ldots, 0, 1), (0, \ldots, 0, 1, 0), \ldots, (1, 0, \ldots, 0)\]

- Allocating tiles along any one of these direction to the same processor ⇒ internalizes communication

- In addition, tiles in any of these direction will have to be executed sequentially anyway (because of dependences)

- Mapping an \(n\)-D tile space onto an \(n - 1\) dimensional torus

- If an \(n - 1\) dimensional torus can be embedded in the processor interconnection network, we get good performance.
Tile Size Optimization

- Assume \( n \)-dimensional iteration space of size \( N \times N \times \ldots \times N \)
- \( n \)-dimensional tiles assumed to be of size \( l \times l \times \ldots \times l \)
- Communication cost model

\[
t_{\text{comm}} = s + w \cdot length
\]

\( s = \) message startup cost; \( w = \) per word cost. All costs normalized with respect to time to execute one iteration

- Cost of transferring \( c \) values per iteration across a tile boundary \((n - 1)\) dimensional surface

\[
= s + cw l^{n-1}
\]

- Execution cost per tile,

\[
T \approx \frac{nN}{l} \left( l^n + (n - 1)cw l^{n-1} + (n - 1)s \right)
\]
Tile Size Optimization - continued

- Assuming $s \neq 0$, setting $\frac{dT}{dl} = 0$, we get

$$\frac{Kn}{l} \left[ Kl^{K-1} + wc(K - 1)^2l^{K-2} \right]$$

$$= \frac{Kn}{l^2} \left[ l^K + (K - 1)wcK^{K-1} + (K - 1)s \right]$$

- If $w$ is negligible (per word cost $\approx 0$),

$$l^n = s \quad \text{or} \quad l_{opt} = \sqrt[n]{s}$$

Balanced computation and communication

- If $s = 0$, then $l_{opt} = 1$
- For $n = 3$, with $w \approx 0$, setting $\frac{dT}{dl} = 0$,

$$l^3 + l^2wc = s$$

- This equation can be solved analytically if $s \geq \frac{4w^3c^3}{27}$
- For other cases, $l_{opt}$ can be obtained numerically